AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Informatyki Elektroniki i Telekomunikacji

Katedra Informatyki

Podstawy Baz Danych

Projekt – „Konferencje”

Dokumentacja

Autorzy:
Filip Dej
Mateusz Szpyrka

Spis treści

1 Opis zadania

Zadanie miało polegać na zaprojektowaniu i zaimplementowaniu bazy danych dla firmy organizującej konferencje. Dodatkowo po zakończeniu implementacji mieliśmy także wygenerować dane spełniające założenia zadania, a następnie wypełnić nimi naszą bazę w stopniu odpowiadającym trzyletniej działalności firmy.

2 Wymagania funkcjonalne

Zgodnie ze specyfikacją bazy opisaną w treści zadania określiliśmy funkcje, jakie powinna realizować gotowa baza danych, aby prawidłowo spełniać postawione wymagania:

- Do bazy może zostać dodana konferencja. Pełny opis konferencji składa się z opisu każdego dnia konferencji, a także opisów powiązanych z konkretnym dniem warsztatów. Dodatkowo organizator konferencji może określić różne ceny uczestnictwa w danym dniu w zależności od terminu złożenia rezerwacji (dla każdego dnia konferencji określone są progi cenowe obowiązujące przy składaniu rezerwacji w ustalonych przedziałach czasowych). Ceny na warsztaty powiązane z dniami konferencji są natomiast stałe. Do wybranych dni konferencji organizator może także ustalić zniżkę cenową dla studentów;

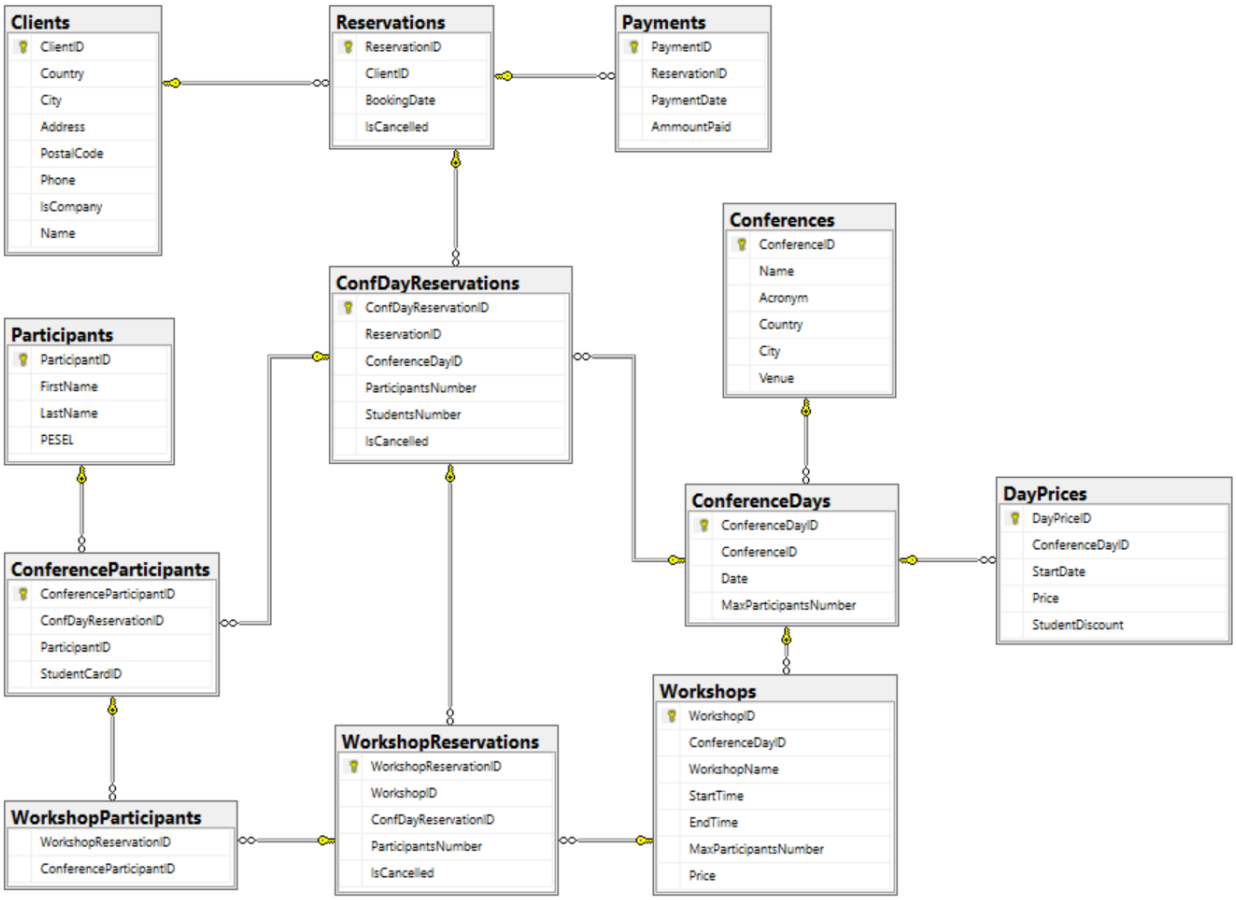
- Do bazy danych może zostać dodany nowy klient, może to być zarówno firma, jak i osoba prywatna (tylko zarejestrowani w bazie klienci mogą składać rezerwacje na konferencje);

- Klient może złożyć rezerwację na wybrane dni konferencji oraz warsztaty. Opis rezerwacji składa się z informacji odnośnie liczby rezerwowanych miejsc dla wybranych dni konferencji oraz warsztatów (w przypadku dni konferencji klient podaje także liczbę miejsc studenckich). Po złożeniu rezerwacji klient ma możliwość podania list uczestników rejestrowanych na poszczególne zarezerwowane wydarzenia (jeśli dane osobowe uczestników nie zostaną podane do tygodnia przed datą rozpoczęcia konferencji, firma kontaktuje się z klientem);

- Swoją rezerwację klient może opłacać w dowolnych ratach, jednak w przypadku braku jakichkolwiek wpłat na tydzień po złożeniu rezerwacji - zajęte miejsca na poszczególne wydarzenia zostają zwolnione, a rezerwacja jest anulowana (zakładamy, że w przypadku opłacenia tylko części należności po okresie tygodnia firma kontaktuje się z klientem);

- Organizatorzy konferencji mają dostęp do danych dotyczących organizowanych przez siebie wydarzeń - list uczestników na poszczególne dni konferencji oraz warsztaty, płatności klientów, a także wybranych danych statystycznych (np. najchętniej rezerwowanych warsztatów lub dni konferencji);

3 Projekt bazy danych



4 Definicje tabel

4.1 Clients

Tabela odpowiadająca za przechowywanie danych kontaktowych oraz podstawowych informacji dotyczących klientów

- ClientID (Klucz główny) - Identyfikuje klienta
- Country - Państwo, z którego pochodzi klient
- City - Miasto, z którego pochodzi klient
- Address - Adres klienta
- PostalCode - Kod pocztowy klienta

- Phone - Numer telefonu klienta
- IsCompany - Pole przechowujące informację, czy klient jest firmą
- Name - Nazwa klienta

```
create table Clients
(
  ClientID int identity
    constraint PK__Payments__9B556A58A8DBBA58
      primary key,
  Country nvarchar(40),
  City nvarchar(40),
  Address nvarchar(120),
  PostalCode nvarchar(10),
  Phone varchar(15),
  IsCompany bit,
  Name nvarchar(120) not null
)
go

create unique index Clients_ClientID_uindex
  on Clients (ClientID)
go
```

4.2 ConfDayReservations

Przechowuje informacje o rezerwacjach na konkretny dzień konferencji

- ConfDayReservationID (Klucz główny) - Identyfikuje rezerwację na dany dzień konferencji
- ReservationID (Klucz obcy) - Identyfikuje rezerwację, do której dana rezerwacja dnia konferencji należy
- ConferenceDayID (Klucz obcy) - Identyfikuje dzień konferencji, z którym jest związana dana rezerwacja
- ParticipantsNumber - Określa całkowitą liczbę uczestników w rezerwacji na dany dzień konferencji
- StudentsNumber - Określa liczbę studentów w rezerwacji na dany dzień konferencji
- IsCancelled - Przechowuje informację, czy dana rezerwacja na dzień konferencji została anulowana

```
create table ConfDayReservations
(
  ConfDayReservationID int identity
    constraint PK__ConfDayR__61523CE5B6016876
      primary key,
  ReservationID int not null
    constraint ReservationID_ReservationID__fk
      references Reservations,
  ConferenceDayID int not null
```

```
    constraint ConfDayReservations_ConferenceDays__fk
      references ConferenceDays,
  ParticipantsNumber int not null
    constraint CK_ConfDayReservations_PositiveParticipantsNumber
      check ([ParticipantsNumber]>0),
  StudentsNumber int not null,
  IsCancelled bit default 0 not null,
  constraint CK_ConfDayReservations_ProperStudentsNumber
    check ([StudentsNumber]>=0 AND [StudentsNumber]<=[ParticipantsNumber])
)
go
```

4.3 ConferenceDays

Przechowuje informacje o dniach konferencji

- DonferenceDayID (Klucz główny) - Identyfikuje dzień konferencji
- ConferenceID (Klucz obcy) - Identyfikuje konferencję, z którą powiązany jest dzień
- Date - Data dzienna dnia konferencji
- MaxParticipantNumber - Maksymalna liczba uczestników danego dnia konferencji

```
create table ConferenceDays
(
  ConferenceDayID int identity
    constraint PK__Conferen__E57A64628B520B05
      primary key,
  ConferenceID int not null
    constraint ConferenceDays_Conferences__fk
      references Conferences,
  Date date not null,
  MaxParticipantsNumber int not null
    constraint CK_ConferenceDays_PositiveParticipantsNumber
      check ([MaxParticipantsNumber]>0)
)
go
```

4.4 ConferenceParticipants

Odpowiada za przyporządkowanie uczestników rezerwacjom na dni konferencji

- ConferenceParticipantID (Klucz główny)-Identyfikuje uczestnika przypisanego do rezerwacji dnia konferencji
- ConfDayReservationID (Klucz obcy) - Identyfikuje rezerwację konferencji
- ParticipantID (Klucz obcy) - Identyfikuje uczestnika
- StudentCardid - Numer legitymacji studenckiej

```
create table ConferenceParticipants
(
  ConferenceParticipantID int identity
    constraint PK__Conferen__2B5E41B0B91EDBD1
      primary key,
  ConfDayReservationID int not null
    constraint ConferenceParticipants_ConfDayReservations__fk
      references ConfDayReservations,
  ParticipantID int
    constraint ConferenceParticipants_Participants__fk
      references Participants,
  StudentCardID char(5)
)
go
```

4.5 Conferences

Przechowuje informacje dotyczące organizowanych konferencji

- ConferenceID (Klucz główny)- Identyfikuje konferencję
- Name - Pełna nazwa konferencji
- Acronym - Skrótowa nazwa konferencji
- Country - Państwo, w którym odbywa się konferencja
- City - Miasto, w którym odbywa się konferencja
- Venue - Dokładna lokalizacja, w której odbywa się konferencja

```
create table Conferences
(
  ConferenceID int identity
    constraint PK__Conferen__4A95EA939368B092
      primary key,
  Name nvarchar(120) not null,
  Acronym nvarchar(15) not null,
  Country nvarchar(40) not null,
  City nvarchar(40) not null,
  Venue nvarchar(120) not null
)
go
```

4.6 DayPrices

Przechowuje informacje o progach cenowych konferencji

- DayPriceID (Klucz Główny)- Identyfikuje próg cenowy
- ConferenceDayID (Klucz obcy) - Identyfikuje dzień konferencji, którego dotyczy dany próg cenowy
- StartDate - Data, od której próg zaczyna obowiązywać

- Price - Cena na danym progu
- StudentDiscount - Procentowa zniżka studencka

```
create table DayPrices
(
  DayPriceID int identity
    constraint PK__DayPrice__ED8F267FEE031823
      primary key,
  ConferenceDayID int not null
    constraint DayPrices_ConferenceDays__fk
      references ConferenceDays,
  StartDate date not null,
  Price money
    constraint CK_DayPrices_NonnegativePrice
      check ([Price]>=0),
  StudentDiscount decimal(3,2)
    constraint CK_DayPrices_ProperStudentDiscount
      check ([StudentDiscount]>=0 AND [StudentDiscount]<=1)
)
go
```

4.7 Participants

Przechowuje podstawowe informacje dotyczące uczestników

- ParticipantID (Klucz główny) - Identyfikuje uczestnika
- FirstName - Imię uczestnika
- LastName - Nazwisko uczestnika
- PESEL - Nr PESEL uczestnika

```
create table Participants
(
  ParticipantID int identity
    constraint PK__Particip__7227997E8BB43A7E
      primary key,
  FirstName nvarchar(40) not null,
  LastName nvarchar(40) not null,
  PESEL char(11) not null
)
go

create unique index UQ_Participants_PESEL
  on Participants (PESEL)
go
```

4.8 Payments

Tabela odpowiadająca za przechowywanie płatności klientów za zarezerwowane miejsca

- PaymentID (Klucz główny)- Identyfikuje płatność

- ReservationID (Klucz obcy) - Identyfikuje Rezerwację, z którą związana jest płatność
- PaymentDate - Data płatności
- AmmountPaid - Wpłacona kwota

```
create table Payments
(
  PaymentID int identity
    constraint PK__Particip__7227997E314D17B4
      primary key,
  ReservationID int not null
    constraint Payments_Reservations__fk
      references Reservations,
  PaymentDate date not null,
  AmmountPaid money not null
    constraint CK_Payments_PositiveAmmountPaid
      check ([AmmountPaid]>0)
)
go
```

4.9 Reservations

Przechowuje informacje o złożonych rezerwacjach

- ReservationID (Klucz główny) - Identyfikuje rezerwację
- ClientID (Klucz obcy) - Identyfikuje klienta, który złożył rezerwację
- BookingDate - Data złożenia rezerwacji
- IsCancelled - Przechowuje informację, czy rezerwacja została anulowana

```
create table Reservations
(
  ReservationID int identity
    constraint PK__Reservat__B7EE5F04600864FE
      primary key,
  ClientID int not null
    constraint Reservations_Clients__fk
      references Clients,
  BookingDate date not null
    constraint CK_Reservations_ValidBookingDate
      check ([BookingDate]<=getdate()),
  IsCancelled bit default 0 not null
)
go

create unique index Reservations_ReservationID_uindex
  on Reservations (ReservationID)
go
```

4.10 WorkshopParticipants

Odpowiada za przyporządkowanie rezerwacji warsztatów uczestnikom przypisanym do rezerwacji danego dnia konferencji

- WorkshopReservationID (Klucz główny) - Identyfikuje Rezerwację warsztatu
- ConferenceParticipantID (Klucz główny) - Identyfikuje uczestnika przypisanego do rezerwacji dnia konferencji

```
create table WorkshopParticipants
(
  WorkshopReservationID int not null
    constraint WorkshopParticipants_WorkshopReservations__fk
      references WorkshopReservations,
  ConferenceParticipantID int not null
    constraint WorkshopParticipants_ConferenceParticipants__fk
      references ConferenceParticipants
)
go
```

4.11 WorkshopReservations

Przechowuje informacje o rezerwacjach na warsztaty

- WorkshopReservationID (Klucz główny) - Identyfikuje rezerwację warsztatów
- WorkshopID (Klucz obcy) - Identyfikuje warsztaty, których dotyczy dana rezerwacja
- ConfDayReservationID (Klucz obcy) - Identyfikuje rezerwację dnia konferencji, w którym odbywają się warsztaty
- ParticipantsNumber - Określa rezerwowaną liczbę miejsc na warsztatach
- IsCancelled - Określa, czy dana rezerwacja warsztatów została anulowana

```
create table WorkshopReservations
(
  WorkshopReservationID int identity
    constraint PK__Workshop__38E529656C927FC5
      primary key,
  WorkshopID int not null
    constraint WorkshopReservations_Workshops__fk
      references Workshops,
  ConfDayReservationID int not null
    constraint WorkshopReservations_ConfDayReservations__fk
      references ConfDayReservations,
  ParticipantsNumber int not null
    constraint CK_WorkshopReservations_PositiveParticipantsNumber
      check ([ParticipantsNumber]>0),
  IsCancelled bit default 0 not null
)
go
```

4.12 Workshops

Przechowuje informacje o warsztatach

- WorkshopID (Klucz główny) - Identyfikuje warsztaty
- ConferenceDayID (Klucz obcy) - Identyfikuje dzień konferencji, w którym odbywają się dane warsztaty
- WorkshopName - Nazwa warsztatów
- StartTime - Godzina początku warsztatów
- EndTime - Godzina końca warsztatów
- MaxParticipantNumber - Maksymalna liczba uczestników mogących wziąć udział w warsztatach
- Price - Cana warsztatów

```
create table Workshops
(
  WorkshopID int identity
    constraint PK__Workshop__7A008C2A97FD2FD6
      primary key,
  ConferenceDayID int not null
    constraint Workshops_ConferenceDays__fk
      references ConferenceDays,
  WorkshopName nvarchar(120) not null,
  StartTime time not null,
  EndTime time not null,
  MaxParticipantsNumber int not null
    constraint CK_Workshops_PositiveParticipantsNumber
      check ([MaxParticipantsNumber]>0),
  Price money not null
    constraint CK_Workshops_NonnegativePrice
      check ([Price]>=0),
  constraint CK_Workshops_ProperTimeRange
    check ([EndTime]>[StartTime])
)
go
```

5 Widoki

5.1 Dotyczące miejsc

5.1.1 AvailableConferences

Wyświetla nadchodzące konferencje na które zostały wolne miejsca

```
CREATE VIEW dbo.VI_AvailableConferences
AS
  SELECT
    *,
    dbo.FN_GetConferenceStartDate(c.ConferenceID) AS ConferenceStartDate,
```

```
    dbo.FN_GetConfDayFreePlaces(c.ConferenceID) AS FreePlaces
  FROM Conferences c
  WHERE c.ConferenceID IN(SELECT ConferenceID FROM VI_AvailableConfDays)
GO
```

5.1.2 AvailableConfDays

Wyświetla nadchodzące dni konferencji na które zostały jeszcze wolne miejsca

```
CREATE VIEW dbo.VI_AvailableConfDays
AS
  SELECT c.*
  FROM dbo.ConferenceDays c
  WHERE dbo.FN_GetConfDayFreePlaces(c.ConferenceDayID)>0
        AND (SELECT Date
             FROM ConferenceDays
             WHERE c.ConferenceDayID = ConferenceDayID) > GETDATE()
GO
```

5.1.3 AvailableWorkshops

Wyświetla nadchodzące warsztaty na które zostały wolne miejsca

```
CREATE VIEW dbo.VI_AvailableWorkshops
AS
  SELECT w.*
  FROM dbo.Workshops w
  WHERE dbo.FN_GetWorkshopFreePlaces(w.WorkshopID)>0
        AND w.ConferenceDayID IN(SELECT ConferenceDayID FROM VI_AvailableConfDays)
go
```

5.1.4 ConferenceDayPlaces

Wyświetla dni konferencji wraz ze statystykami dotyczącymi zajętości miejsc

```
CREATE VIEW dbo.VI_ConferenceDayPlaces
AS
  SELECT c.ConferenceID,c.ConferenceDayID,c.Date,
    c.MaxParticipantsNumber AS NumberOfPlaces,
    dbo.FN_GetConfDayBookedPlaces(c.ConferenceDayID) AS NumberOfBookedPlaces,
    dbo.FN_GetConfDayFreePlaces(c.ConferenceDayID) AS NumberOfFreePlaces
  FROM dbo.ConferenceDays c
GO
```

5.1.5 WorkshopPlaces

Wyświetla warsztaty wraz ze statystykami dotyczącymi zajętości miejsc

```
CREATE VIEW dbo.VI_WorkshopPlaces
AS
  SELECT w.WorkshopID,w.StartTime,w.EndTime,
    w.MaxParticipantsNumber AS NumberOfPlaces,
    dbo.FN_GetWorkshopBookedPlaces(w.WorkshopID) AS NumberOfBookedPlaces,
    dbo.FN_GetWorkshopFreePlaces(w.WorkshopID) AS NumberOfFreePlaces
  FROM dbo.Workshops w
GO
```

5.2 Dotyczące rezerwacji

5.2.1 CancelledReservations

Wyświetla anulowane rezerwacje

```
CREATE VIEW dbo.VI_CancelledReservations
AS
  SELECT r.*
  FROM Reservations r
  WHERE r.IsCancelled=1
GO
```

5.2.2 FullyPaidReservations

Wyświetla w pełni opłacone rezerwacje

```
CREATE VIEW dbo.VI_FullyPaidReservations
AS
  SELECT r.*
  FROM Reservations r
  WHERE dbo.FN_GetReservationTotalPrice(ReservationID)=
        (SELECT ISNULL(SUM(p.AmmountPaid) ,0)
         FROM Payments p
         WHERE p.ReservationID=r.ReservationID)
GO
```

5.2.3 OverpaidReservations

Wyświetla nadpłacone rezerwacje

```
CREATE VIEW dbo.VI_OverpaidReservations
AS
  SELECT r.*
  FROM Reservations r
  WHERE dbo.FN_GetReservationTotalPrice(ReservationID)<
        (SELECT ISNULL(SUM(p.AmmountPaid),0)
         FROM Payments p
         WHERE p.ReservationID=r.ReservationID)
GO
```

5.2.4 UnderpaidReservations

Wyświetla niedopłacone rezerwacje

```
CREATE VIEW dbo.VI_UnderpaidReservations
AS
  SELECT r.*
  FROM Reservations r
  WHERE dbo.FN_GetReservationTotalPrice(r.ReservationID)>
        (SELECT ISNULL(SUM(p.AmmountPaid),0)
         FROM Payments p
         WHERE p.ReservationID=r.ReservationID)
GO
```

5.2.5 UncompletedReservations

Wyświetla nie zakończone rezerwacje wraz z powodem

```
CREATE VIEW dbo.VI_UncompletedReservations
AS
  SELECT r.*,'ConferenceDay' AS Cause
  FROM Reservations r
  WHERE r.ReservationID IN(SELECT uc.ReservationID
                           FROM VI_UncompletedConfDayReservations uc)
  UNION
  SELECT r.*,'Workshop' AS Cause
  FROM Reservations r
  WHERE r.ReservationID IN(SELECT cdr.ReservationID
                           FROM  ConfDayReservations cdr
                           WHERE cdr.ConfDayReservationID
                                 IN(SELECT uw.ConfDayReservationID
                                    FROM VI_UncompletedWorkshopReservations uw))
GO
```

5.2.6 UncompletedConfDayReservations

Wyświetla nie wypełnione rezerwacje dni konferencji będące mniej niż 2 tygodnie od startu konferencji

```
CREATE VIEW dbo.VI_UncompletedConfDayReservations
AS
  SELECT r.*, dbo.FN_GetConfDayReservationSignedParticipantsNumber(r.ConfDayReservationID)
              AS NumberOfSignedParticipants
  FROM ConfDayReservations r

  JOIN ConferenceDays cd
      ON cd.ConferenceDayID=r.ConferenceDayID
      AND dbo.FN_GetConfDayReservationSignedParticipantsNumber(r.ConfDayReservationID)
                    < r.ParticipantsNumber
      AND GETDATE()>DATEADD(WEEK,-2,dbo.FN_GetConferenceStartDate(cd.ConferenceID))
GO
```

5.2.7 UncompletedWorkshopReservations

Wyświetla nie wypełnione rezerwacje warsztatów będące mniej niż 2 tygodnie od startu konferencji

```
CREATE VIEW dbo.VI_UncompletedWorkshopReservations
AS
  SELECT r.*, r.ParticipantsNumber -
          dbo.FN_GetWorkshopReservationSignedParticipantsNumber(r.WorkshopReservationID)
                AS MissingParticipants
  FROM WorkshopReservations r
  JOIN ConfDayReservations cdr
      ON r.ConfDayReservationID=cdr.ConfDayReservationID
         AND dbo.FN_GetWorkshopReservationSignedParticipantsNumber(r.WorkshopReservationID)
                   < r.ParticipantsNumber
  JOIN ConferenceDays cd
      ON cdr.ConferenceDayID=cd.ConferenceDayID AND
          GETDATE()>DATEADD(WEEK,-2,dbo.FN_GetConferenceStartDate(cd.ConferenceID))
go
```

5.3 Pozostałe

5.3.1 ClientDetails

Wyświetla szczegółowe informacje o klientach

```
CREATE VIEW dbo.VI_ClientDetails
AS
  SELECT
    c.ClientID,
    c.Name,
    c.Country,
    c.City,
    c.PostalCode,
    c.Address,
    c.Phone,
    IIF(c.IsCompany=1,'Company','Individual') Type
  From Clients c
GO
```

5.3.2 ConfDayRanking

Wyświetla dni konferencji o największym procencie zainteresowania

```
CREATE VIEW dbo.VI_ConfDayRanking
AS
  SELECT c.*, (dbo.FN_GetConfDayBookedPlaces(c.ConferenceDayID)*100/c.MaxParticipantsNumber)
                              AS Interest
  FROM dbo.ConferenceDays c
GO
```

5.3.3 WorkshopRanking

Wyświetla warsztaty o największym procencie zainteresowania

```
CREATE VIEW dbo.VI_WorkshopRanking
AS
  SELECT w.*, (dbo.FN_GetWorkshopBookedPlaces(w.WorkshopID)*100/w.MaxParticipantsNumber)
                              AS Interest
  FROM dbo.Workshops w
go
```

6 Funkcje

6.1 Funkcje dotyczące organizowanych wydarzeń

6.1.1 GenerateConferenceDayList

Generuje listę uczestników danego dnia konferencji.

```
CREATE FUNCTION FN_GenerateConferenceDayList(@ConferenceDayID INT)
  RETURNS TABLE
AS
  RETURN (
  SELECT p.FirstName, p.LastName
  FROM Participants p
  WHERE p.ParticipantID
        IN(
```

```
          SELECT cp.ParticipantID
          FROM ConferenceParticipants cp
          WHERE cp.ConfDayReservationID
                IN(
                  SELECT cdr.ConfDayReservationID
                  FROM ConfDayReservations cdr
                  WHERE IsCancelled=0 AND ConferenceDayID=@ConferenceDayID
                )
        )
  )
```

6.1.2 GenerateWorkshopsList

Generuje listę uczestników danego warsztatu.

```
CREATE FUNCTION FN_GenerateWorkshopsList(@WorkshopID INT)
  RETURNS TABLE
AS
  RETURN (
  SELECT p.FirstName, p.LastName
  FROM Participants p
  WHERE p.ParticipantID
        IN(
          SELECT cp.ParticipantID
          FROM ConferenceParticipants cp
          WHERE cp.ConferenceParticipantID
                IN(
                  SELECT cdr.ConferenceParticipantID
                  FROM WorkshopParticipants cdr
                  WHERE WorkshopID=@WorkshopID
                )
        )
  )
```

6.1.3 GetConfDayBookedPlaces

Zwraca liczbę zarezerwowanych miejsc na dany dzień konferencji.

```
CREATE FUNCTION FN_GetConfDayBookedPlaces(@ConferenceDayID INT)
  RETURNS INT
AS BEGIN

  RETURN isnull((
    SELECT sum(ParticipantsNumber)
    FROM ConfDayReservations
    WHERE @ConferenceDayID = ConferenceDayID AND IsCancelled = 0
  ), 0)

END
```

6.1.4 GetConfDayFreePlaces

Zwraca liczbę wolnych miejsc na dany dzień konferencji.

```
CREATE FUNCTION dbo.FN_GetConfDayFreePlaces(@ConferenceDayID INT)
  RETURNS INT
AS BEGIN

  DECLARE @BookedPlaces INT = isnull(dbo.FN_GetConfDayBookedPlaces(@ConferenceDayID), 0)
```

```
  RETURN (
    SELECT MaxParticipantsNumber
    FROM ConferenceDays
    WHERE @ConferenceDayID = ConferenceDayID
  ) - @BookedPlaces

END
```

6.1.5 GetConferenceStartDate

Zwraca początek danej konferencji na podstawie dat dni konferencji (wybiera najwcześniejszą datę).

```
CREATE FUNCTION dbo.FN_GetConferenceStartDate(@ConferenceID INT)
  RETURNS DATE
AS BEGIN
  RETURN (
    SELECT min(cd.Date)
    FROM ConferenceDays AS cd
    WHERE cd.ConferenceID = @ConferenceID
    GROUP BY cd.ConferenceID
  )
END
```

6.1.6 GetWorkshopBookedPlaces

Zwraca liczbę zarezerwowanych miejsc na dany warsztat.

```
CREATE FUNCTION dbo.FN_GetWorkshopBookedPlaces(@WorkshopID INT)
  RETURNS INT
AS BEGIN

  RETURN isnull((
    SELECT sum(ParticipantsNumber)
    FROM WorkshopReservations
    WHERE @WorkshopID = WorkshopID AND IsCancelled = 0
  ), 0)

END
```

6.1.7 GetWorkshopFreePlaces

Zwraca liczbę wolnych miejsc na dany warsztat.

```
CREATE FUNCTION dbo.FN_GetWorkshopFreePlaces(@WorkshopID INT)
  RETURNS INT
AS BEGIN

  DECLARE @BookedPlaces INT = isnull(dbo.FN_GetWorkshopBookedPlaces(@WorkshopID), 0)

  RETURN (
    SELECT MaxParticipantsNumber
    FROM Workshops
    WHERE @WorkshopID = WorkshopID
  ) - @BookedPlaces

END
```

6.2 Funkcje dotyczące składanych rezerwacji

6.2.1 GetConfDayReservationPrice

Zwraca sumaryczny koszt danej rezerwacji pojedynczego dnia konferencji.

```
CREATE FUNCTION dbo.FN_GetConfDayReservationPrice(@ConfDayReservationID INT)
  RETURNS INT
AS BEGIN

  DECLARE @ConferenceDayID INT, @BookingDate DATE

  SET @ConferenceDayID = (
    SELECT ConferenceDayID
    FROM ConfDayReservations
    WHERE @ConfDayReservationID = ConfDayReservationID
  )

  SET @BookingDate = (
    SELECT BookingDate
    FROM Reservations AS R
    JOIN ConfDayReservations AS CDR
      ON R.ReservationID = CDR.ReservationID
    WHERE @ConfDayReservationID = ConfDayReservationID
  )

  DECLARE @NormalPrice INT, @StudentPrice INT

  SET @NormalPrice = (
    SELECT Price
    FROM DayPrices
    WHERE DayPriceID = dbo.FN_GetProperDayPriceID(@BookingDate, @ConferenceDayID)
  )

  SET @StudentPrice = (
    SELECT Price * StudentDiscount
    FROM DayPrices
    WHERE DayPriceID = dbo.FN_GetProperDayPriceID(@BookingDate, @ConferenceDayID)
  )

  RETURN (
    SELECT @NormalPrice * (ParticipantsNumber - ConfDayReservations.StudentsNumber) +
        @StudentPrice * StudentsNumber
    FROM ConfDayReservations
    WHERE @ConfDayReservationID = ConfDayReservationID
  )
END
```

6.2.2 GetConfDayReservationSignedParticipantsNumber

Zwraca liczbę zarejestrowanych osób w obrębie danej rezerwacji na dzień konferencji.

```
CREATE FUNCTION FN_GetConfDayReservationSignedParticipantsNumber(@ConfDayReservationID INT)
  RETURNS INT
AS BEGIN

  RETURN (
    SELECT count(*)
    FROM (
      SELECT *
      FROM ConferenceParticipants
      WHERE @ConfDayReservationID = ConfDayReservationID
    ) t
  )
```

```
END
```

6.2.3 GetConfDayReservationSignedStudentsNumber

Zwraca liczbę zarejestrowanych studentów w obrębie danej rezerwacji na dzień konferencji.

```
CREATE FUNCTION FN_GetConfDayReservationSignedStudentsNumber(@ConfDayReservationID INT)
  RETURNS INT
AS BEGIN

  RETURN (
    SELECT count(*)
    FROM (
      SELECT *
      FROM ConferenceParticipants
      WHERE @ConfDayReservationID = ConfDayReservationID AND StudentCardID IS NOT NULL
    ) t
  )

END
```

6.2.4 GetReservationAlreadyPaidAmmount

Zwraca sumę wpłat złożonych za daną rezerwację.

```
CREATE FUNCTION dbo.FN_GetReservationAlreadyPaidAmmount(@ReservationID INT)
  RETURNS INT
AS
BEGIN

  RETURN isnull((
    SELECT sum(AmmountPaid)
    FROM Payments
    WHERE @ReservationID = ReservationID
  ), 0)

END
```

6.2.5 GetReservationTotalPrice

Zwraca sumaryczny koszt całej rezerwacji.

```
CREATE FUNCTION dbo.FN_GetReservationTotalPrice(@ReservationID INT)
  RETURNS INT
AS BEGIN

  DECLARE @TotalConfDayReservationsPrice INT, @TotalWorkshopReservationsPrice INT

  SET @TotalConfDayReservationsPrice = (

    SELECT sum(ConfDayReservationPrice)
    FROM (
      SELECT dbo.FN_GetConfDayReservationPrice(ConfDayReservationID) AS ConfDayReservationPrice
      FROM ConfDayReservations
      WHERE ReservationID = @ReservationID
    ) t
  )

  SET @TotalWorkshopReservationsPrice = (
```

```
    SELECT sum(WorkshopReservationPrice)
    FROM (
      SELECT dbo.FN_GetWorkshopReservationPrice(WorkshopReservationID) AS WorkshopReservationPrice
      FROM WorkshopReservations
      WHERE ConfDayReservationID IN (
        SELECT ConfDayReservationID
        FROM ConfDayReservations
        WHERE @ReservationID = ReservationID
      )
    ) t
  )

  RETURN isnull(@TotalConfDayReservationsPrice, 0) + isnull(@TotalWorkshopReservationsPrice, 0)
END
```

6.2.6 GetWorkshopReservationPrice

Zwraca sumaryczny koszt danej rezerwacji na pojedynczy warsztat.

```
CREATE FUNCTION dbo.FN_GetWorkshopReservationPrice(@WorkshopReservationID INT)
  RETURNS INT
AS BEGIN

  RETURN (
    SELECT WR.ParticipantsNumber * W.Price
    FROM WorkshopReservations AS WR
    JOIN Workshops AS W
      ON WR.WorkshopID = W.WorkshopID
    WHERE @WorkshopReservationID = WR.WorkshopReservationID
  )
END
```

6.2.7 GetWorkshopReservationSignedParticipantsNumber

Zwraca liczbę zarejestrowanych uczestników w obrębie danej rezerwacji na warsztat.

```
CREATE FUNCTION dbo.FN_GetWorkshopReservationSignedParticipantsNumber(@WorkshopReservationID INT)
  RETURNS INT
AS BEGIN
  RETURN (
    SELECT count(*)
    FROM dbo.WorkshopReservations AS WR
    JOIN WorkshopParticipants AS WP
      ON WR.WorkshopReservationID = WP.WorkshopReservationID
    WHERE @WorkshopReservationID = WR.WorkshopReservationID
  )
END
```

6.3 Funkcje pomocnicze

6.3.1 AnyWorkshopsColliding

Sprawdza czy jakikolwiek uczestnik w bazie danych jest zapisany na warsztaty, które ze sobą kolidują.

```
CREATE FUNCTION FN_AnyWorkshopsColliding()
  RETURNS BIT
AS
BEGIN
```

```
  IF EXISTS(
    SELECT *
    FROM WorkshopParticipants AS wp1
    JOIN ConferenceParticipants AS cp1
      ON wp1.ConferenceParticipantID = cp1.ConferenceParticipantID
    JOIN ConferenceParticipants AS cp2
      ON cp1.ConferenceParticipantID <> cp2.ConferenceParticipantID
        AND cp1.ParticipantID = cp2.ParticipantID
    JOIN WorkshopParticipants AS wp2
      ON cp2.ConferenceParticipantID = wp2.ConferenceParticipantID
    JOIN WorkshopReservations AS wr1
      ON wr1.WorkshopReservationID = wp1.WorkshopReservationID
    JOIN WorkshopReservations AS wr2
      ON wr2.WorkshopReservationID = wp2.WorkshopReservationID
    WHERE dbo.FN_WorkshopsColliding (wp1.WorkshopReservationID, wp2.WorkshopReservationID) = 1
  )
  BEGIN
    RETURN 1
  END

  RETURN 0

END
```

6.3.2 ParticipantWorkshopsColliding

Zwraca informację, czy którekolwiek warsztaty podanego uczestnika kolidują ze sobą.

```
CREATE FUNCTION dbo.FN_ParticipantWorkshopsColliding(@ParticipantID INT)
  RETURNS BIT
AS
BEGIN

  IF EXISTS(
    SELECT * FROM Participants AS p
    JOIN ConferenceParticipants AS cp1
      ON p.ParticipantID = cp1.ParticipantID AND p.ParticipantID = @ParticipantID
    JOIN ConferenceParticipants AS cp2
      ON p.ParticipantID = cp2.ParticipantID
         AND cp1.ConferenceParticipantID <> cp2.ConferenceParticipantID
    JOIN WorkshopParticipants AS wp1
      ON cp1.ConferenceParticipantID = wp1.ConferenceParticipantID
    JOIN WorkshopParticipants AS wp2
      ON cp2.ConferenceParticipantID = wp2.ConferenceParticipantID
    JOIN WorkshopReservations AS wr1
      ON wp1.WorkshopReservationID = wr1.WorkshopReservationID
    JOIN WorkshopReservations AS wr2
      ON wp2.WorkshopReservationID = wr2.WorkshopReservationID
    WHERE dbo.FN_WorkshopsColliding (wr1.WorkshopID, wr2.WorkshopID) = 1
  )
  RETURN 1

  RETURN 0

END
```

6.3.3 GetProperDayPriceID

Dla danego dnia konferencji zwraca ID progu cenowego obowiązującego podczas podanej daty złożenia rezerwacji.

```
CREATE FUNCTION dbo.FN_GetProperDayPriceID(@Date DATE, @ConferenceDayID INT)
  RETURNS INT
AS
BEGIN
  RETURN (
    SELECT TOP 1 DayPriceID
    FROM DayPrices
    WHERE @Date > StartDate AND @ConferenceDayID = ConferenceDayID
    ORDER BY StartDate DESC
  )
END
```

6.3.4 WorkshopsColliding

Sprawdza czy dwa podane warsztaty ze sobą kolidują.

```
CREATE FUNCTION dbo.FN_WorkshopsColliding(@workshop1ID INT, @workshop2ID INT)
  RETURNS BIT
AS
BEGIN

  DECLARE @date1 DATE = (
    SELECT Date
    FROM ConferenceDays
    WHERE ConferenceDayID = (
      SELECT ConferenceDayID
      FROM Workshops
      WHERE WorkshopID = @workshop1ID
    )
  )

  DECLARE @date2 DATE = (
    SELECT Date
    FROM ConferenceDays
    WHERE ConferenceDayID = (
      SELECT ConferenceDayID
      FROM Workshops
      WHERE WorkshopID = @workshop2ID
    )
  )

  DECLARE @startTime1 TIME = (
    SELECT StartTime
    FROM Workshops
    WHERE @workshop1ID = WorkshopID
  )

  DECLARE @startTime2 TIME = (
    SELECT StartTime
    FROM Workshops
    WHERE @workshop2ID = WorkshopID
  )

  DECLARE @endTime1 TIME = (
    SELECT EndTime
    FROM Workshops
    WHERE @workshop1ID = WorkshopID
  )

  DECLARE @endTime2 TIME = (
    SELECT EndTime
    FROM Workshops
    WHERE @workshop2ID = WorkshopID
  )
```

```
  IF (
    @date1 = @date2
    AND (@startTime1 >= @startTime2 AND @startTime1 < @endTime2)
    OR (@startTime2 >= @startTime1 AND @startTime2 < @endTime1)
  )
  BEGIN
    RETURN 1
  END

  RETURN 0
END
```

7 Procedury

7.1 Dodawanie rekordów

7.1.1 AddClient

Dodaje nowego klienta do bazy.

```
CREATE PROCEDURE dbo.PR_AddClient
  @Country NVARCHAR(40),
  @City NVARCHAR(40),
  @Address NVARCHAR(120),
  @PostalCode NVARCHAR(10),
  @Phone VARCHAR(15),
  @Name NVARCHAR(120),
  @IsCompany BIT
AS
BEGIN

    BEGIN TRY
      INSERT INTO Clients (Country, City, Address, PostalCode, Phone, IsCompany, Name)
        VALUES (@Country, @City, @Address, @PostalCode, @Phone, @IsCompany, @Name)
    END TRY
    BEGIN CATCH
      DECLARE @errormsg NVARCHAR(2048)
      = 'An error ocurred while adding Client. Error message: ' + ERROR_MESSAGE();
      ;THROW 52000, @errormsg, 1
    END CATCH
END
```

7.1.2 AddConfDayReservation

Dodaje do bazy nową rezerwację na dzień konferencji.

```
CREATE PROCEDURE dbo.PR_AddConfDayReservation
  @ReservationID INT,
  @ConferenceDayID INT,
  @ParticipantsNumber INT,
  @StudentsNumber INT
AS
BEGIN

    BEGIN TRY
      IF NOT EXISTS(
        SELECT *
        FROM Reservations
        WHERE @ReservationID = ReservationID
```

```
      )
      BEGIN
        ;THROW 60000, 'Given ReservationID does not exist', 1
      END

      IF NOT EXISTS(
        SELECT *
        FROM ConferenceDays
        WHERE @ConferenceDayID = ConferenceDayID
      )
        BEGIN
        ;THROW 60000, 'Given ConferenceDayID does not exist', 1
      END

      INSERT INTO dbo.ConfDayReservations(ReservationID, ConferenceDayID,
                          ParticipantsNumber, StudentsNumber)
        VALUES (@ReservationID, @ConferenceDayID, @ParticipantsNumber, @StudentsNumber)
    END TRY
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) = 'An error occurred while adding confDayReservation: '
                          + ERROR_MESSAGE ();
      THROW 60000, @errorMsg, 1
    END CATCH
END
```

7.1.3 AddConference

Dodaje do bazy nową konferencję.

```
CREATE PROCEDURE dbo.PR_AddConference
  @Name NVARCHAR(120),
  @Acronym NVARCHAR(15),
  @Country NVARCHAR(40),
  @City NVARCHAR(40),
  @Venue NVARCHAR(120)
AS
BEGIN

    BEGIN TRY
      INSERT INTO Conferences (Name, Acronym, Country, City, Venue)
        VALUES (@Name, @Acronym, @Country, @City, @Venue)
    END TRY
    BEGIN CATCH
      DECLARE @errormsg NVARCHAR(2048)
      = 'An error ocurred while adding Conference. Error message: ' + ERROR_MESSAGE();
      ;THROW 52000, @errormsg, 1
    END CATCH
END
```

7.1.4 AddConferenceDay

Dodaje do bazy nowy dzień konferencji.

```
CREATE PROCEDURE dbo.PR_AddConferenceDay
  @ConferenceID INT,
  @Date DATE,
  @MaxParticipantsNumber INT
AS
BEGIN

  BEGIN TRY
```

```
  IF NOT EXISTS(
    SELECT *
    FROM Conferences
    WHERE @ConferenceID = ConferenceID
  )
  BEGIN
    ;THROW 60000, 'Given ConferenceID does not exist', 1
  END

  INSERT INTO dbo.ConferenceDays(ConferenceID, Date, MaxParticipantsNumber)
  VALUES (@ConferenceID, @Date, @MaxParticipantsNumber)
  END TRY
  BEGIN CATCH

  DECLARE @errorMsg nvarchar (2048) = 'An error occurred while adding ConferenceDay: '
                  + ERROR_MESSAGE ();
  THROW 60000, @errorMsg, 1
  END CATCH
END
```

7.1.5 AddConferenceParticipant

Dodaje do bazy nowego uczestnika dnia konferencji.

```
CREATE PROCEDURE dbo.PR_AddConferenceParticipant
  @ConfDayReservationID INT,
  @ParticipantID INT,
  @StudentCardID CHAR(6)
AS
BEGIN

    BEGIN TRY
      IF NOT EXISTS(
        SELECT * FROM ConfDayReservations WHERE @ConfDayReservationID=ConfDayReservationID
      ) BEGIN
          ;THROW 54321, 'Given ConfDayReservationID does not exist', 1
        END
      IF NOT EXISTS(
        SELECT * FROM Participants WHERE @ParticipantID=ParticipantID
      ) BEGIN
          ;THROW 54321, 'Given ParticipantID does not exist', 1
        END
      INSERT INTO ConferenceParticipants (ConfDayReservationID, ParticipantID, StudentCardID)
        VALUES (@ConfDayReservationID, @ParticipantID, @StudentCardID)
    END TRY
    BEGIN CATCH
      DECLARE @errormsg NVARCHAR(2048)
      = 'An error ocurred while adding ConferenceParticipant. Error message: ' + ERROR_MESSAGE();
      ;THROW 52000, @errormsg, 1
    END CATCH
END
```

7.1.6 AddDayPrice

Dodaje do bazy nowy próg cenowy dla dnia konferencji.

```
CREATE PROCEDURE dbo.PR_AddDayPrice
  @ConferenceDayID INT,
  @StartDate DATE,
  @Price MONEY,
  @StudentDiscount DECIMAL(3, 2)
AS
```

```
BEGIN

  BEGIN TRY

  IF NOT EXISTS(
    SELECT *
    FROM ConferenceDays
    WHERE @ConferenceDayID = ConferenceDayID
  )
  BEGIN
    ;THROW 60000, 'Given ConferenceDayID does not exist', 1
  END

  INSERT INTO dbo.DayPrices(ConferenceDayID, StartDate, Price, StudentDiscount)
  VALUES (@ConferenceDayID, @StartDate, @Price, @StudentDiscount)
  END TRY
  BEGIN CATCH
  DECLARE @errorMsg nvarchar (2048) = 'An error occurred while adding DayPrice: '
                                                    + ERROR_MESSAGE ();
  THROW 60000, @errorMsg, 1
  END CATCH
END
```

7.1.7 AddParticipant

Dodaje do bazy dane osobowe nowego uczestnika.

```
CREATE PROCEDURE dbo.PR_AddParticipant
  @FirstName NVARCHAR(40),
  @LastName NVARCHAR(40),
  @PESEL CHAR(11)
AS
BEGIN

    BEGIN TRY
      INSERT INTO Participants (FirstName, LastName, PESEL)
        VALUES (@FirstName, @LastName, @PESEL)
    END TRY
    BEGIN CATCH
      DECLARE @errormsg NVARCHAR(2048)
      = 'An error ocurred while adding Participant. Error message: ' + ERROR_MESSAGE();
      ;THROW 52000, @errormsg, 1
    END CATCH
END
```

7.1.8 AddPayments

Dodaje do bazy dane na temat nowej wpłaty.

```
CREATE PROCEDURE dbo.PR_AddPayments
  @ReservationID INT,
  @PaymentDate DATE,
  @AmmountPaid MONEY
AS
BEGIN

  BEGIN TRY
  IF NOT EXISTS(
    SELECT * FROM Reservations WHERE ReservationID=@ReservationID
  )
  BEGIN
    ;THROW 54321, 'Given ReservationID does not exist', 1
```

```
  END
  INSERT INTO Payments (ReservationID, PaymentDate, AmmountPaid)
  VALUES (@ReservationID, @PaymentDate, @AmmountPaid)
  END TRY
  BEGIN CATCH
  DECLARE @errormsg NVARCHAR(2048)
    = 'An error occurred while adding Payment: ' + ERROR_MESSAGE();
  ;THROW 52000, @errormsg, 1
  END CATCH
END
```

7.1.9 AddReservation

Dodaje do bazy nową rezerwację.

```
CREATE PROCEDURE dbo.PR_AddReservation
  @ClientID INT,
  @BookingDate DATE
AS
BEGIN

  BEGIN TRY
  IF NOT EXISTS(
    SELECT *
    FROM Clients
    WHERE @ClientID = ClientID
  )
  BEGIN
    ;THROW 60000, 'Given ClientID does not exist', 1
  END

  INSERT INTO dbo.Reservations(ClientID, BookingDate)
  VALUES (@ClientID, @BookingDate)
  END TRY
  BEGIN CATCH
  DECLARE @errorMsg nvarchar (2048) = 'An error occurred while adding Reservation: '
                    + ERROR_MESSAGE ();
  THROW 60000, @errorMsg, 1
  END CATCH
END
```

7.1.10 AddWorkshop

Dodaje do bazy nowy warsztat.

```
CREATE PROCEDURE dbo.PR_AddWorkshop
  @ConferenceDayID INT,
  @WorkshopName NVARCHAR(120),
  @StartTime TIME,
  @EndTime TIME,
  @MaxParticipantsNumber INT,
  @Price MONEY
AS
BEGIN

    BEGIN TRY

      IF NOT EXISTS(
        SELECT *
        FROM ConferenceDays
        WHERE @ConferenceDayID = ConferenceDayID
      )
```

```
        BEGIN
        ;THROW 60000, 'Given ConferenceDayID does not exist', 1
      END

      INSERT INTO dbo.Workshops(ConferenceDayID, WorkshopName,
                                StartTime, EndTime, MaxParticipantsNumber, Price)
        VALUES (@ConferenceDayID, @WorkshopName,
                                  @StartTime, @EndTime, @MaxParticipantsNumber, @Price)
    END TRY
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) = 'An error occurred while adding Workshop: '
                                                                          + ERROR_MESSAGE ();
      THROW 60000, @errorMsg, 1
    END CATCH
END
```

7.1.11 AddWorkshopParticipant

Dodaje do bazy nowego uczestnika warsztatów.

```
CREATE PROCEDURE dbo.PR_AddWorkshopParticipant
  @WorkshopID INT,
  @ConferenceParticipantID INT
AS
BEGIN

  BEGIN TRY

  IF NOT EXISTS(
    SELECT *
    FROM Workshops
    WHERE @WorkshopID = WorkshopID
  )
  BEGIN
    ;THROW 60000, 'Given WorkshopID does not exist', 1
  END

  IF NOT EXISTS(
    SELECT *
    FROM ConferenceParticipants
    WHERE @ConferenceParticipantID = ConferenceParticipantID
  )
  BEGIN
    ;THROW 60000, 'Given ConferenceParticipantID does not exist', 1
  END

  INSERT INTO dbo.WorkshopParticipants(WorkshopID, ConferenceParticipantID)
  VALUES (@WorkshopID, @ConferenceParticipantID)
  END TRY
  BEGIN CATCH
  DECLARE @errorMsg nvarchar (2048) = 'An error occurred while adding WorkshopParticipant: '
                      + ERROR_MESSAGE ();
  THROW 60000, @errorMsg, 1
  END CATCH
END
```

7.1.12 AddWorkshopReservation

Dodaje do bazy nową rezerwację na warsztat.

```
CREATE PROCEDURE dbo.PR_AddWorkshopReservation
  @WorkshopID INT,
```

```
  @ConfDayReservationID INT,
  @ParticipantsNumber INT,
  @StudentsNumber INT
AS
BEGIN

  BEGIN TRY
  IF NOT EXISTS(
    SELECT *
    FROM ConfDayReservations
    WHERE @ConfDayReservationID = ConfDayReservationID
  )
  BEGIN
    ;THROW 60000, 'Given ConfDayReservationID does not exist', 1
  END

  IF NOT EXISTS(
    SELECT *
    FROM Workshops
    WHERE @WorkshopID = WorkshopID
  )
  BEGIN
    ;THROW 60000, 'Given WorkshopID does not exist', 1
  END

  INSERT INTO dbo.WorkshopReservations(WorkshopID, ConfDayReservationID,
                                       ParticipantsNumber, StudentsNumber)
  VALUES (@WorkshopID, @ConfDayReservationID, @ParticipantsNumber, @StudentsNumber)
  END TRY
  BEGIN CATCH
  DECLARE @errorMsg nvarchar (2048) = 'An error occurred while adding WorkshopReservation: '
                                    + ERROR_MESSAGE ();
  THROW 60000, @errorMsg, 1
  END CATCH
END
```

7.2 Modyfikowanie rekordów

7.2.1 CancelConfDayReservation

Anuluje rezerwację na dany dzień konferencji.

```
CREATE PROCEDURE dbo.PR_CancelConfDayReservation
  @ConfDayReservationID INT
AS
BEGIN
  BEGIN TRY
    IF NOT EXISTS(
    SELECT *
    FROM ConfDayReservations
    WHERE ConfDayReservationID=@ConfDayReservationID
  )
  BEGIN
    ;THROW 61001, 'Given ConfDayReservationID does not exist', 1
  END
  IF @ConfDayReservationID IS NOT NULL
  BEGIN
    UPDATE  ConfDayReservations
    SET IsCancelled=1 WHERE ConfDayReservationID=@ConfDayReservationID
  END
  END TRY
  BEGIN CATCH
  DECLARE @errorMsg nvarchar (2048) = 'An error occurred while cancelling ConfDayReservation:
                                    + ERROR_MESSAGE ();
```

```
  THROW 61000, @errorMsg, 1
  END CATCH
END
```

7.2.2 CancelReservation

Anuluje całą rezerwację. Wywołanie powoduje anulowanie wszystkich rezerwacji na dni konferencji oraz warsztaty związane z daną rezerwacją.

```
CREATE PROCEDURE dbo.PR_CancelReservation
  @ReservationID INT
AS
BEGIN
  BEGIN TRY
  IF NOT EXISTS(
    SELECT *
    FROM Reservations
    WHERE ReservationID=@ReservationID
  )
  BEGIN
    ;THROW 61001, 'Given ReservationID does not exist', 1
  END
  IF @ReservationID IS NOT NULL
  BEGIN
    UPDATE Reservations
    SET IsCancelled=1 WHERE ReservationID=@ReservationID

    UPDATE ConfDayReservations
    SET IsCancelled=1
    WHERE @ReservationID = ReservationID

    UPDATE WorkshopReservations
    SET IsCancelled=1
    WHERE ConfDayReservationID IN (
    SELECT ConfDayReservationID
    FROM ConfDayReservations
    WHERE ReservationID = @ReservationID
    )

  END
  END TRY
  BEGIN CATCH
  DECLARE @errorMsg nvarchar (2048) = 'An error occurred while cancelling Reservation: '
                                    + ERROR_MESSAGE ();
  THROW 61000, @errorMsg, 1
  END CATCH
END
```

7.2.3 CancelWorkshopReservation

Anuluje rezerwację na warsztat.

```
CREATE PROCEDURE dbo.PR_CancelWorkshopReservation
  @WorkshopReservationID INT
AS
BEGIN
  BEGIN TRY
  IF NOT EXISTS(
    SELECT *
    FROM WorkshopReservations
    WHERE WorkshopReservationID=@WorkshopReservationID
  )
```

```
  BEGIN
    ;THROW 61001, 'Given WorkshopReservationID does not exist', 1
  END
  IF @WorkshopReservationID IS NOT NULL
  BEGIN
    UPDATE WorkshopReservations
    SET IsCancelled=1 WHERE WorkshopReservationID=@WorkshopReservationID
  END
  END TRY
  BEGIN CATCH
  DECLARE @errorMsg nvarchar (2048) = 'An error occurred while cancelling WorkshopReservation: '
                                    + ERROR_MESSAGE ();
  THROW 61000, @errorMsg, 1
  END CATCH
END
```

7.2.4 UpdateClientDetails

Pozwala na zmodyfikowanie danych dotyczących klienta.

```
CREATE PROCEDURE dbo.PR_UpdateClientDetails
  @ClientID INT,
  @Country NVARCHAR(40),
  @City NVARCHAR(40),
  @Address NVARCHAR(120),
  @PostalCode NVARCHAR(10),
  @Phone VARCHAR(15),
  @IsCompany BIT
AS
BEGIN

    BEGIN TRY

      IF NOT EXISTS(
        SELECT *
        FROM Clients
        WHERE ClientID=@ClientID
      )
        BEGIN
        ;THROW 60000, 'Given ClientID does not exist', 1
      END
      IF @Country IS NOT NULL
        BEGIN
          UPDATE Clients
            SET Country = @Country WHERE ClientID=@ClientID
        END
      IF @City IS NOT NULL
        BEGIN
          UPDATE Clients
            SET City = @City WHERE ClientID=@ClientID
        END
      IF @Address IS NOT NULL
        BEGIN
          UPDATE Clients
            SET Address = @Address WHERE ClientID=@ClientID
        END
      IF @PostalCode IS NOT NULL
        BEGIN
          UPDATE Clients
            SET PostalCode = @PostalCode WHERE ClientID=@ClientID
        END
      IF @Phone IS NOT NULL
        BEGIN
          UPDATE Clients
```

```
            SET Phone = @Phone WHERE ClientID=@ClientID
        END
      IF @IsCompany IS NOT NULL
        BEGIN
          UPDATE Clients
            SET IsCompany = @IsCompany WHERE ClientID=@ClientID
        END
    END TRY
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) = 'An error occurred while changing client details : '
                                + ERROR_MESSAGE ();
      THROW 60000, @errorMsg, 1
    END CATCH
END
```

7.2.5 UpdateConferenceDayDate

Pozwala na zmodyfikowanie daty dnia konferencji.

```
CREATE PROCEDURE dbo.PR_UpdateConferenceDayDate
  @ConferenceDayID INT,
  @Date DATE
AS
BEGIN

    BEGIN TRY

      IF NOT EXISTS(
        SELECT *
        FROM ConferenceDays
        WHERE ConferenceDayID=@ConferenceDayID
      )
        BEGIN
        ;THROW 60000, 'Given ConferenceDayID does not exist', 1
      END
      UPDATE ConferenceDays
        SET Date=@Date WHERE ConferenceDayID=@ConferenceDayID
    END TRY
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) = 'An error occurred while changing ConferenceDay date : '
                                        + ERROR_MESSAGE ();
      THROW 60000, @errorMsg, 1
    END CATCH
END
```

7.2.6 UpdateConferenceDayMaxParticipantsNumber

Pozwala na zmodyfikowanie maksymalnej liczby uczestników danego dnia konferencji.

```
CREATE PROCEDURE dbo.PR_UpdateConferenceDayMaxParticipantsNumber
  @ConferenceDayID INT,
  @MaxParticipantsNumber INT
AS
BEGIN

    BEGIN TRY

      IF NOT EXISTS(
        SELECT *
        FROM ConferenceDays
        WHERE ConferenceDayID=@ConferenceDayID
      )
```

```
        BEGIN
        ;THROW 60000, 'Given ConferenceDayID does not exist', 1
      END
      UPDATE ConferenceDays
        SET MaxParticipantsNumber=@MaxParticipantsNumber WHERE ConferenceDayID=@ConferenceDayID
    END TRY
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) =
                  'An error occurred while changing ConferenceDay maxParticipantsNumber : '
                   + ERROR_MESSAGE ();
      THROW 60000, @errorMsg, 1
    END CATCH
END
```

7.2.7 UpdateConferenceDetails

Pozwala na zmodyfikowanie danych dotyczących konferencji.

```
CREATE PROCEDURE dbo.PR_UpdateConferenceDetails
  @ConferenceID INT,
  @Name NVARCHAR(120),
  @Acronym NVARCHAR(15),
  @Country NVARCHAR(40),
  @City NVARCHAR(40),
  @Venue NVARCHAR(120)
AS
BEGIN

    BEGIN TRY

      IF NOT EXISTS(
        SELECT *
        FROM Conferences
        WHERE ConferenceID=@ConferenceID
      )
        BEGIN
        ;THROW 60000, 'Given ConferenceID does not exist', 1
      END
      IF @Name IS NOT NULL
        BEGIN
          UPDATE Conferences
            SET Name = @Name WHERE ConferenceID=@ConferenceID
        END
      IF @Acronym IS NOT NULL
        BEGIN
          UPDATE Conferences
            SET Acronym = @Acronym WHERE ConferenceID=@ConferenceID
        END
      IF @Country IS NOT NULL
        BEGIN
          UPDATE Conferences
            SET Country = @Country WHERE ConferenceID=@ConferenceID
        END
      IF @City IS NOT NULL
        BEGIN
          UPDATE Conferences
            SET City = @City WHERE ConferenceID=@ConferenceID
        END
      IF @Venue IS NOT NULL
        BEGIN
          UPDATE Conferences
            SET Venue = @Venue WHERE ConferenceID=@ConferenceID
        END
    END TRY
```

```
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) = 'An error occurred while changing Conference details : '
                                + ERROR_MESSAGE ();
      THROW 60000, @errorMsg, 1
    END CATCH
END
```

7.2.8 UpdateDayPriceDetails

Pozwala na zmodyfikowanie danych dotyczących progu cenowego dla danego dnia konferencji.

```
CREATE PROCEDURE dbo.PR_UpdateDayPriceDetails
  @DayPriceID INT,
  @StartDate DATE,
  @Price MONEY,
  @StudentDiscount DECIMAL(3,2)
AS
BEGIN

    BEGIN TRY

      IF NOT EXISTS(
        SELECT *
        FROM DayPrices
        WHERE DayPriceID=@DayPriceID
      )
        BEGIN
        ;THROW 60000, 'Given DayPriceID does not exist', 1
      END
      IF @StartDate IS NOT NULL
        BEGIN
          UPDATE DayPrices
            SET StartDate = @StartDate WHERE DayPriceID=@DayPriceID
        END
      IF @Price IS NOT NULL
        BEGIN
          UPDATE DayPrices
            SET Price = @Price WHERE DayPriceID=@DayPriceID
        END
      IF @StudentDiscount IS NOT NULL
        BEGIN
          UPDATE DayPrices
            SET StudentDiscount = @StudentDiscount WHERE DayPriceID=@DayPriceID
        END
    END TRY
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) = 'An error occurred while changing DayPrice details: '
                                + ERROR_MESSAGE ();
      THROW 60000, @errorMsg, 1
    END CATCH
END
```

7.2.9 UpdateWorkshopMaxParticipantsNumber

Pozwala na zmodyfikowanie maksymalnej liczby uczestników danego warsztatu.

```
CREATE PROCEDURE dbo.PR_UpdateWorkshopMaxParticipantsNumber
  @WorkshopID INT,
  @MaxParticipantsNumber INT
AS
BEGIN
```

```
    BEGIN TRY

      IF NOT EXISTS(
        SELECT *
        FROM Workshops
        WHERE @WorkshopID = WorkshopID
      )
        BEGIN
        ;THROW 60000, 'Given WorkshopID does not exist', 1
      END
      UPDATE Workshops
        SET MaxParticipantsNumber=@MaxParticipantsNumber WHERE WorkshopID=@WorkshopID
    END TRY
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) =
              'An error occurred while changing Workshop maxParticipantsNumber : ' + ERROR_MESSAGE ();
      THROW 60000, @errorMsg, 1
    END CATCH
END
```

7.2.10 UpdateWorkshopPrice

Pozwala na zmodyfikowanie ceny warsztatu.

```
CREATE PROCEDURE dbo.PR_UpdateWorkshopPrice
  @WorkshopID INT,
  @Price INT
AS
BEGIN

    BEGIN TRY

      IF NOT EXISTS(
        SELECT *
        FROM Workshops
        WHERE @WorkshopID = WorkshopID
      )
        BEGIN
        ;THROW 60000, 'Given WorkshopID does not exist', 1
      END
      UPDATE Workshops
        SET Price=@Price WHERE WorkshopID=@WorkshopID
    END TRY
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) = 'An error occurred while changing Workshop price : '
                      + ERROR_MESSAGE ();
      THROW 60000, @errorMsg, 1
    END CATCH
END
```

7.2.11 UpdateWorkshopTime

Pozwala na zmodyfikowanie godzin rozpoczęcia oraz zakończenia danego warsztatu.

```
CREATE PROCEDURE dbo.PR_UpdateWorkshopTime
  @WorkshopID INT,
  @StartTime TIME,
  @EndTime TIME
AS
BEGIN

    BEGIN TRY
```

```
      IF NOT EXISTS(
        SELECT *
        FROM Workshops
        WHERE @WorkshopID = WorkshopID
      )
        BEGIN
        ;THROW 60000, 'Given WorkshopID does not exist', 1
      END
      UPDATE Workshops
        SET StartTime=@StartTime, EndTime=@EndTime WHERE WorkshopID=@WorkshopID
    END TRY
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) = 'An error occurred while changing Workshop time : '
                          + ERROR_MESSAGE ();
      THROW 60000, @errorMsg, 1
    END CATCH
END
```

7.3 Usuwanie rekordów

7.3.1 RemoveConfDayReservation

Usuwa z bazy rezerwację na dzień konferencji.

```
CREATE PROCEDURE dbo.PR_RemoveConfDayReservation
  @ConfDayReservationID INT
AS
BEGIN
    BEGIN TRY
      IF NOT EXISTS(
        SELECT *
        FROM ConfDayReservations
        WHERE ConfDayReservationID=@ConfDayReservationID
      )
      BEGIN
        ;THROW 61001, 'Given ConfDayReservationID does not exist', 1
      END
      IF @ConfDayReservationID IS NOT NULL
        BEGIN
          DELETE  FROM ConfDayReservations
                  WHERE ConfDayReservationID=@ConfDayReservationID
        END
    END TRY
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) = 'An error occurred while deleting ConfDayReservation: '
                          + ERROR_MESSAGE ();
      THROW 61000, @errorMsg, 1
    END CATCH
END
```

7.3.2 RemoveConferenceParticipant

Usuwa z bazy uczestnika dnia konferencji.

```
CREATE PROCEDURE dbo.PR_RemoveConferenceParticipant
  @ConferenceParticipantID INT
AS
BEGIN
    BEGIN TRY
      IF NOT EXISTS(
        SELECT *
```

```
        FROM ConferenceParticipants
        WHERE ConferenceParticipantID=@ConferenceParticipantID
      )
      BEGIN
        ;THROW 61001, 'Given ConferenceParticipantID does not exist', 1
      END
      IF @ConferenceParticipantID IS NOT NULL
        BEGIN
          DELETE  FROM ConferenceParticipants
                  WHERE ConferenceParticipantID=@ConferenceParticipantID
        END
    END TRY
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) = 'An error occurred while deleting ConferenceParticipant: '
                          + ERROR_MESSAGE ();
      THROW 61000, @errorMsg, 1
    END CATCH
END
```

7.3.3 RemoveReservation

Usuwa z bazy rezerwację.

```
CREATE PROCEDURE dbo.PR_RemoveReservation
  @ReservationID INT
AS
BEGIN
    BEGIN TRY
      IF NOT EXISTS(
        SELECT *
        FROM Reservations
        WHERE ReservationID=@ReservationID
      )
      BEGIN
        ;THROW 61001, 'Given ReservationID does not exist', 1
      END
      IF @ReservationID IS NOT NULL
        BEGIN
          DELETE  FROM Reservations
                  WHERE ReservationID=@ReservationID
        END
    END TRY
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) = 'An error occurred while deleting Reservation: '
                          + ERROR_MESSAGE ();
      THROW 61000, @errorMsg, 1
    END CATCH
END
```

7.3.4 RemoveWorkshopParticipant

Usuwa z bazy uczestnika warsztatu.

```
CREATE PROCEDURE dbo.PR_RemoveWorkshopParticipant
  @ConferenceParticipantID INT,
  @WorkshopID INT
AS
BEGIN
    BEGIN TRY
      IF NOT EXISTS(
        SELECT *
        FROM WorkshopParticipants
```

```
      WHERE ConferenceParticipantID=@ConferenceParticipantID OR WorkshopID=@WorkshopID
    )
    BEGIN
      ;THROW 61001, 'Given ConferenceParticipantID and WorkshopID does not exist', 1
    END
    IF @ConferenceParticipantID IS NOT NULL
      BEGIN
        DELETE  FROM WorkshopParticipants
                WHERE ConferenceParticipantID=@ConferenceParticipantID
      END
    IF @WorkshopID IS NOT NULL
      BEGIN
        DELETE  FROM WorkshopParticipants
                WHERE WorkshopID=@WorkshopID
      END
  END TRY
  BEGIN CATCH
    DECLARE @errorMsg nvarchar (2048) = 'An error occurred while deleting WorkshopParticipant: '
                          + ERROR_MESSAGE ();
    THROW 61000, @errorMsg, 1
  END CATCH
END
```

7.3.5 RemoveWorkshopReservation

Usuwa z bazy rezerwację na warsztat.

```
CREATE PROCEDURE dbo.PR_RemoveWorkshopReservation
  @WorkshopReservationID INT
AS
BEGIN
    BEGIN TRY
      IF NOT EXISTS(
        SELECT *
        FROM WorkshopReservations
        WHERE WorkshopReservationID=@WorkshopReservationID
      )
      BEGIN
        ;THROW 61001, 'Given WorkshopReservationID does not exist', 1
      END
      IF @WorkshopReservationID IS NOT NULL
        BEGIN
          DELETE  FROM WorkshopReservations
                  WHERE WorkshopReservationID=@WorkshopReservationID
        END
    END TRY
    BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048) = 'An error occurred while deleting WorkshopReservation: '
                          + ERROR_MESSAGE ();
      THROW 61000, @errorMsg, 1
    END CATCH
END
```

8 Triggery

8.1 Na tabeli ConfDayReservations

8.1.1 NotEnoughPlaces

Blokuje dodanie rezerwacji na dzień konferencji, jeśli nie ma już wystarczająco dużo wolnych miejsc.

```
CREATE TRIGGER TR_ConfDayReservations_NotEnoughPlaces
ON dbo.ConfDayReservations
  AFTER INSERT, UPDATE
AS
BEGIN
  IF EXISTS(
    SELECT *
    FROM ConfDayReservations
    WHERE dbo.FN_GetConfDayFreePlaces(ConferenceDayID) < 0
  )
  BEGIN
    ROLLBACK
    ;THROW 60000, 'Not enough free places for given ConferenceDayID', 1
  END
END
```

8.2 Na tabeli ConferenceDays

8.2.1 CannotDecreasePlaces

Blokuje zmniejszenie ilości miejsc na dzień konferencji, jeśli suma zarezerwowanych miejsc przekracza nowy limit.

```
CREATE TRIGGER TR_ConferenceDays_CannotDecreasePlaces
  ON dbo.ConferenceDays
  AFTER UPDATE
AS
BEGIN

  IF EXISTS(
    SELECT *
    FROM ConferenceDays
    WHERE dbo.FN_GetConfDayFreePlaces (ConferenceDayID) < 0
  )
  BEGIN
    ROLLBACK
    ;THROW 60000,
            'Registered participants number exceeds new places limit for updated ConferenceDay', 1
  END

END
```

8.2.2 WorkshopsCollidingAfterUpdate

Blokuje zmianę daty dnia konferencji jeśli powoduje to kolidowanie warsztatów jakiegokolwiek uczestnika w bazie.

```
CREATE TRIGGER dbo.TR_ConferenceDays_WorkshopsCollidingAfterUpdate
  ON dbo.ConferenceDays
  AFTER UPDATE
```

```
AS
BEGIN

  IF dbo.FN_AnyWorkshopsColliding() = 1
  BEGIN
    ROLLBACK
    ;THROW 6000,
            'Changing given ConferenceDay.Date will cause some participant''s workshops collision', 1
  END

END
```

8.3 Na tabeli ConferenceParticipants

8.3.1 AlreadyRegistered

Blokuje dodanie uczestnika dnia konferencji, jeśli taki uczestnik jest już na ten dzień zarejestrowany.

```
CREATE TRIGGER TR_ConferenceParticipants_AlreadyRegistered
ON dbo.ConferenceParticipants
  AFTER INSERT, UPDATE
AS
BEGIN
  IF EXISTS(
    SELECT *
    FROM inserted
    JOIN ConferenceParticipants AS cp
      ON inserted.ParticipantID = cp.ParticipantID
         AND inserted.ConfDayReservationID = cp.ConfDayReservationID
         AND inserted.ConferenceParticipantID <> cp.ConferenceParticipantID
  )
  BEGIN

    ROLLBACK
    ;THROW 60000, 'Participant already registered for given ConfDayReservationID', 1

  END
END
```

8.3.2 CancelledReservation

Blokuje dodanie uczestnika do anulowanej rezerwacji na dzień konferencji.

```
CREATE TRIGGER dbo.TR_ConferenceParticipants_CancelledReservation
  ON dbo.ConferenceParticipants
  AFTER INSERT, UPDATE
AS
BEGIN

  IF EXISTS(
    SELECT *
    FROM inserted
    JOIN ConfDayReservations AS cdr
      ON inserted.ConfDayReservationID = cdr.ConfDayReservationID
    WHERE cdr.IsCancelled = 1
  )
  BEGIN

    ROLLBACK
    ;THROW 60000, 'Cannot add new participant to cancelled ConfDayReservation', 1
```

```
  END

END
```

8.3.3 NoMoreFreePlaces

Blokuje dodanie uczestnika dnia konferencji, jeśli limit miejsc (lub studentów) określony w rezerwacji został już osiągnięty.

```
CREATE TRIGGER dbo.TR_ConferenceParticipants_NoMoreFreePlaces
  ON dbo.ConferenceParticipants
  AFTER INSERT, UPDATE
AS
BEGIN

    IF EXISTS(
      SELECT *
      FROM inserted AS i
      JOIN ConfDayReservations AS CDR
      ON i.ConfDayReservationID = CDR.ConfDayReservationID
      WHERE dbo.FN_GetConfDayReservationSignedParticipantsNumber(i.ConfDayReservationID) >
                          CDR.ParticipantsNumber
    )
    BEGIN
      ROLLBACK
      ;THROW 60000, 'No more free places for given ConfDayReservationID', 1
    END

    IF EXISTS(
      SELECT *
      FROM inserted AS i
      JOIN ConfDayReservations AS CDR
      ON i.ConfDayReservationID = CDR.ConfDayReservationID
      WHERE dbo.FN_GetConfDayReservationSignedStudentsNumber(i.ConfDayReservationID) >
                          CDR.StudentsNumber
    )
    BEGIN
      ROLLBACK
      ;THROW 60000, 'No more free student places for given ConfDayReservationID', 1
    END
END
```

8.4 Na tabeli DayPrices

8.4.1 IllegalStartDate

Blokuje dodanie progu cenowego, jeśli data początku jego obowiązywania wypada po dacie dnia konferencji.

```
CREATE TRIGGER dbo.TR_DayPrices_IllegalStartDate
ON dbo.DayPrices
  AFTER INSERT, UPDATE
AS
BEGIN

  IF EXISTS(
    SELECT *
    FROM inserted
    JOIN ConferenceDays AS cd
      ON cd.ConferenceDayID = inserted.ConferenceDayID
```

```
    WHERE cd.Date < inserted.StartDate
  )
  BEGIN
    ROLLBACK
    ;THROW 60000, 'Cannot add DayPrice with StartDate later than ConferenceDay.Date', 1
  END

END
```

8.4.2 RepeatedStartDate

Blokuje dodanie progu cenowego do dnia konferencji, jeśli data jego początku pokrywa się z datą innego progu dla tego samego dnia.

```
CREATE TRIGGER TR_DayPrices_RepeatedStartDate
ON dbo.DayPrices
  AFTER INSERT, UPDATE
AS
BEGIN
  IF EXISTS(
    SELECT count(*)
    FROM DayPrices
    GROUP BY StartDate, ConferenceDayID
    HAVING count(*) > 1
  )
  BEGIN
    ROLLBACK
    ;THROW 60000, 'Given StartDate already exists in DayPrices', 1
  END
END
```

8.5 Na tabeli WorkshopParticipants

8.5.1 CancelledReservation

Blokuje dodanie uczestnika warsztatu do anulowanej rezerwacji.

```
CREATE TRIGGER dbo.TR_WorkshopParticipants_CancelledReservation
  ON dbo.WorkshopParticipants
  AFTER INSERT, UPDATE
AS
BEGIN

  IF EXISTS(
    SELECT *
    FROM inserted
    JOIN WorkshopReservations AS wr
      ON inserted.WorkshopReservationID = wr.WorkshopReservationID
    WHERE wr.IsCancelled = 1
  )
  BEGIN

    ROLLBACK
    ;THROW 60000, 'Cannot add new participant to cancelled WorkshopReservation', 1

  END

END
```

8.5.2 CollidingWorkshops

Blokuje dodanie uczestnika warsztatu, jeśli nowy warsztat koliduje z którymkolwiek innym warsztatem, na który uczestnik jest zarejestrowany.

```
CREATE TRIGGER TR_WorkshopParticipants_CollidingWorkshops
ON dbo.WorkshopParticipants
  AFTER INSERT, UPDATE
AS
BEGIN

  IF EXISTS(
    SELECT *
    FROM inserted
    JOIN ConferenceParticipants AS cp
      ON inserted.ConferenceParticipantID = cp.ConferenceParticipantID
    WHERE dbo.FN_ParticipantWorkshopsColliding (cp.ParticipantID) = 1
  )
  BEGIN

    ROLLBACK
    ;THROW 60000, 'Cannot add workshopParticipant because of colliding workshops', 1

  END

END
```

8.5.3 NoMoreFreePlaces

Blokuje dodanie uczestnika warsztatu jeśli limit miejsc określony w rezerwacji na warsztat został osiągnięty.

```
CREATE TRIGGER TR_WorkshopParticipants_NoMoreFreePlaces
ON dbo.WorkshopParticipants
  AFTER INSERT, UPDATE
AS
BEGIN

  IF EXISTS(
    SELECT *
    FROM WorkshopReservations
    WHERE ParticipantsNumber <
            dbo.FN_GetWorkshopReservationSignedParticipantsNumber (WorkshopReservationID)
  )
  BEGIN

    ROLLBACK
    ;THROW 60000, 'No free places left for given WorkshopReservationID', 1

  END

END
```

8.5.4 NotSignedForConfDay

Blokuje dodanie uczestnika do warsztatu, jeśli nie jest on zarejestrowany na powiązany z warsztatem dzień konferencji.

```
CREATE TRIGGER TR_WorkshopParticipants_NotSignedForConfDay
ON dbo.WorkshopParticipants
  AFTER INSERT, UPDATE
```

```
AS
BEGIN

  IF EXISTS(
    SELECT *
    FROM WorkshopParticipants AS wp
    JOIN ConferenceParticipants AS cp
      ON wp.ConferenceParticipantID = cp.ConferenceParticipantID
    JOIN ConfDayReservations AS cdr
      ON cp.ConfDayReservationID = cdr.ConfDayReservationID
    JOIN WorkshopReservations AS wr
      ON wp.WorkshopReservationID = wr.WorkshopReservationID
    WHERE wr.ConfDayReservationID <> cdr.ConfDayReservationID
  )
  BEGIN

    ROLLBACK
    ;THROW 60000, 'Tried do add WorkshopParticipants to WorkshopReservation' +
    '       who is not signed for corelated ConfDayReservation', 1

  END

END
```

8.6 Na tabeli WorkshopReservations

8.6.1 IllegalParticipantsNumber

Blokuje dodanie rezerwacji na warsztat o większej liczbie miejsc niż określona w powiązanej rezerwacji na dzień konferencji.

```
CREATE TRIGGER dbo.TR_WorkshopReservations_IllegalParticipantsNumber
  ON dbo.WorkshopReservations
  AFTER INSERT, UPDATE
AS
BEGIN

  IF EXISTS(
    SELECT *
    FROM inserted
    JOIN ConfDayReservations AS cdr
      ON cdr.ConfDayReservationID = inserted.ConfDayReservationID
    WHERE inserted.ParticipantsNumber > cdr.ParticipantsNumber
  )
  BEGIN
    ROLLBACK
    ;THROW 60000, 'Cannot book more places for workshop than for corelated conference day', 1
  END

END
```

8.6.2 NotEnoughPlaces

Blokuje dodanie rezerwacji na warsztat, jeśli nie ma już wystarczająco dużo wolnych miejsc.

```
CREATE TRIGGER dbo.TR_WorkshopReservations_NotEnoughPlaces
ON dbo.WorkshopReservations
  AFTER INSERT, UPDATE
AS
BEGIN
  IF EXISTS(
```

```
    SELECT *
    FROM WorkshopReservations
    WHERE dbo.FN_GetWorkshopFreePlaces(WorkshopID) < 0
  )
  BEGIN
    ROLLBACK
    ;THROW 60000, 'Not enough free places for given WorkshopID', 1
  END
END
```

8.7 Na tabeli Workshops

8.7.1 TooMuchParticipants

Blokuje dodanie warsztatu o liczbie miejsc większej niż określona w powiązanym dniu konferencji.

```
CREATE TRIGGER TR_Workshops_TooMuchParticipants
  ON dbo.Workshops
  AFTER INSERT, UPDATE
AS
BEGIN

  IF EXISTS(
    SELECT *
    FROM ConferenceDays AS cd
    JOIN Workshops AS w
    ON cd.ConferenceDayID = w.ConferenceDayID
    WHERE cd.MaxParticipantsNumber < w.MaxParticipantsNumber
  )
  BEGIN

    ROLLBACK
    ;THROW 60000, 'MaxParticipantsNumber for workshop cannot be bigger than for ConferenceDay', 1
  END
END
```

8.7.2 CannotDecreasePlaces

Blokuje zmniejszenie liczby miejsc dla danego warsztatu, jeśli suma zarezerwowanych miejsc przekracza nowy limit.

```
CREATE TRIGGER TR_Workshops_CannotDecreasePlaces
  ON dbo.Workshops
  AFTER UPDATE
AS
BEGIN

  IF EXISTS(
    SELECT *
    FROM Workshops
    WHERE dbo.FN_GetWorkshopFreePlaces (WorkshopID) < 0
  )
  BEGIN
    ROLLBACK
    ;THROW 60000, 'Registered participants number exceeds new places limit for updated Workshop', 1
  END

END
```

8.7.3 WorkshopsCollidingAfterUpdate

Blokuje zmianę godzin warsztatu, jeśli powodowałoby to kolizję warsztatów dla któregokolwiek uczestnika w bazie.

```
CREATE TRIGGER dbo.TR_Workshops_WorkshopsCollidingAfterUpdate
  ON dbo.Workshops
  AFTER UPDATE
AS
BEGIN

  IF dbo.FN_AnyWorkshopsColliding() = 1
  BEGIN
    ROLLBACK
    ;THROW 6000,
        'Changing given workshop time range will cause some participant''s workshops collision', 1
  END

END
```

9 Generowanie danych

Postanowiliśmy uzupełnić bazę danych przy pomocy własnego generatora napisanego w SQL, aby mieć jak największy wpływ na poprawność generowanych danych oraz odpowiednią liczebność poszczególnych elementów w bazie. Podczas wypełniania naszej bazy w żadnym momencie nie dezaktywowaliśmy żadnych warunków integralnościowych ani triggerów, dzięki czemu wszystkie dane wewnątrz bazy są zgodne z założeniami określonymi w poprzednich sekcjach.

Bazę wypełnialiśmy tabela po tabeli, każdy etap (wraz z kodem SQL) opisany został w oddzielnym podpunkcie.

9.1 Tabela Participants

```
-- Participants

CREATE TABLE GEN_Participants (
        FirstName varchar(20),
        LastName1 varchar(20),
        LastName2 varchar(20),
        PESEL char(11),
      )

INSERT INTO dbo.GEN_Participants (FirstName, LastName1, LastName2, PESEL)
    VALUES
      ('Krzysztof', 'Bo', 'ryczko', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Jacek', 'Długo', 'polski', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Tadeusz', 'Dy', 'duch', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Łukasz', 'Fa', 'ber', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Piotr', 'Fali', 'szewski', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Barbara', 'Dziur', 'dzia', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Dariusz', 'Koś', 'cielnik', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Marek', 'Ga', 'jęcki', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Wacław', 'Fry', 'drych', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Zbigniew', 'Ką', 'kol', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Darin', 'Niko', 'low', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Marcin', 'Kur', 'dziel', dbo.GEN_FN_GetRandomPesel(RAND())),
```

```
      ('Lucjan', 'Pyt', 'lik', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Waldemar', 'To', 'karz', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Jan', 'Woź', 'niak', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Bartosz', 'Ra', 'koczy', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Klaudia', 'Ba', 'łazy', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Irmina', 'Zio', 'ło', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Piotr', 'Jano', 'wski', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Jakub', 'Przy', 'było', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Mariusz', 'Me', 'szka', dbo.GEN_FN_GetRandomPesel(RAND())),
      ('Magdalena', 'Ty', 'niec', dbo.GEN_FN_GetRandomPesel(RAND()))

INSERT INTO Participants (FirstName, LastName, PESEL)
(
  SELECT
    p1.FirstName,
    p2.LastName1 + p3.LastName2 AS LastName,
    substring(p1.PESEL, 1, 4) + substring(p2.PESEL, 5, 4) + substring(p3.PESEL, 9, 3) AS PESEL
  FROM GEN_Participants AS p1
    CROSS JOIN GEN_Participants AS p2
    CROSS JOIN GEN_Participants AS p3
)
```

W celu wypełnienia tabeli uczestników stworzyliśmy najpierw pomocniczą tabelę, do której ręcznie wprowadziliśmy kilka losowych imion i nazwisk, a także losowo wygenerowany ciąg 11 cyfr imitujący numer PESEL przy pomocy własnej funkcji GetRandomPesel. Każde nazwisko podzieliliśmy na dwa człony, a następnie wygenerowaliśmy 10648 różnych kombinacji wprowadzonych danych. Następnie losowo usunęliśmy 9000 uczestników, co w efekcie dało nam 1648 rekordów w tabeli Participants.

9.1.1 Tabela Clients

```
-- Clients

CREATE TABLE GEN_Clients (
  Country NVARCHAR(20),
  City NVARCHAR(20),
  Address NVARCHAR(50),
  PostalCode NVARCHAR(10),
  Phone NVARCHAR(15),
  IsCompany BIT,
  Name NVARCHAR(50)
)

INSERT INTO GEN_Clients (Country, City, Address, PostalCode, Phone, IsCompany, Name)
    VALUES
      ('Polska', 'Kraków', 'Kawiory 15', '30-123',
                substring(dbo.GEN_FN_GetRandomPesel(RAND()), 1, 9), 1, 'Janusz'),
      ('Polska', 'Warszawa', 'Nowy Świat 45', '33-123',
                substring(dbo.GEN_FN_GetRandomPesel(RAND()), 1, 9), 1, 'Cebula'),
      ('Polska', 'Gdańsk', 'Stoczniówka 432', '37-123',
                substring(dbo.GEN_FN_GetRandomPesel(RAND()), 1, 9), 1, 'Bursztyn'),
      ('Polska', 'Radom', 'Chytrówka 12', '32-123',
                substring(dbo.GEN_FN_GetRandomPesel(RAND()), 1, 9), 1, 'Beka'),
      ('Polska', 'Częstochowa', 'Matki Boskiej Licheńskiej 4', '31-123',
                substring(dbo.GEN_FN_GetRandomPesel(RAND()), 1, 9), 1, 'Maria'),
      ('Polska', 'Kraków', 'Sukiennice', '39-123',
                substring(dbo.GEN_FN_GetRandomPesel(RAND()), 1, 9), 1, 'Wawel'),
      ('Spain', 'Madrid', 'La Rambla 123a', '453-233',
                substring(dbo.GEN_FN_GetRandomPesel(RAND()), 1, 9), 1, 'Fernandez'),
      ('Polska', 'Sosnowiec', 'Dziura 8', '30-123',
                substring(dbo.GEN_FN_GetRandomPesel(RAND()), 1, 9), 1, 'Szwagier'),
      ('Germany', 'Berlin', 'Geschwindigkeitsbegrenzung 1', '3440-123',
                substring(dbo.GEN_FN_GetRandomPesel(RAND()), 1, 9), 1, 'Hans'),
      ('Polska', 'Ustki Środkowe', 'Kawiory 15', '45-123',
```

```
                substring(dbo.GEN_FN_GetRandomPesel(RAND()), 1, 9), 1, 'Grażyna')

DECLARE @Suf TABLE (
  Sufix VARCHAR(20)
)

INSERT INTO @Suf (Sufix)
    VALUES
      ('-Stal'),
      ('-Pol'),
      (' Solutions'),
      (' Academy'),
      ('-Ex'),
      (' Dynamics')

INSERT INTO dbo.Clients(Country, City, Address, PostalCode, Phone, IsCompany, Name) (
  SELECT
    Country,
    City,
    Address,
    PostalCode,
    Phone,
    IsCompany,
    Name + Sufix
  FROM GEN_Clients AS t1
    CROSS JOIN @Suf
)
```

Podobnie jak w poprzednim przypadku, tak i tutaj stworzyliśmy ręcznie wypełnioną tabelę pomocniczą. Następnie utworzyliśmy wszystkie możliwe kombinacje klientów dodając do nazwy klienta jeden z pięciu sufiksów imitujących nazwy różnych korporacji. W efekcie uzyskaliśmy 60 rekordów opisujących klientów.

9.1.2 Tabela Conferences

```
-- Conferences

CREATE TABLE GEN_Conferences (
  Name1 VARCHAR(30),
  Name2 VARCHAR(30),
  Country VARCHAR(20),
  City VARCHAR(20),
  Venue VARCHAR(30)
)

INSERT INTO GEN_Conferences (Name1, Name2, Country, City, Venue)
    VALUES
      ('Krakowskie Dni', 'Czytania Dzieciom', 'Polska', 'Kraków',
                'Centrum Kongresowe ICE'),
      ('Bronowicki Tydzień', 'Miłośników Programowania', 'Polska', 'Kraków',
                'Centrum Kongresowe ICE'),
      ('Konferencja', 'Anime Days', 'Spain', 'Świnoujście', 'Wembley'),
      ('Festiwal', 'Studentów Filologii Rumuńskiej', 'Polska', 'Wyspa Wolin',
                'Wolińskie centrum kultury'),
      ('Kongres', 'Graczy w Grę', 'Polska', 'Warszawa', 'Pałac Kultury i Nauki')

DECLARE @yrs TABLE (
  Year CHAR(4)
)

INSERT INTO @yrs (Year)
    VALUES
      ('2016'), ('2017'), ('2018')
```

```
INSERT INTO Conferences (Name, Acronym, Country, City, Venue) (
  SELECT t1.Name1 + ' ' + t2.Name2 + ' ' + t3.Year AS NAME,
  substring(t1.Name1, 1, 2) + substring(t2.Name2, 1, 2) + substring(t3.Year, 3, 2) AS Acronym,
  t1.Country, t1.City, t1.Venue
  FROM GEN_Conferences AS t1
  CROSS JOIN GEN_Conferences AS t2
  CROSS JOIN @yrs AS t3
)
```

Tutaj także ręcznie stworzyliśmy tabelę pomocniczą, każdą nazwę warsztatu podzieliliśmy na dwa człony, a następnie stworzyliśmy wszystkie możliwe kombinacje nazw warsztatów wraz z latami, w których konferencja miałaby się odbywać. Uzyskaliśmy w ten sposób 75 różnych konferencji.

9.1.3 Tabela ConferenceDays

```
-- ConferenceDays

DECLARE @counter INT = (SELECT count(*) FROM Conferences)
DECLARE @date DATE = '2016-01-01'
WHILE (@counter > 0)
  BEGIN

    DECLARE @confID INT = (
      SELECT TOP 1 ConferenceID
      FROM Conferences
      WHERE ConferenceID NOT IN (
        SELECT ConferenceID
        FROM ConferenceDays
      )
      ORDER BY ConferenceID
    )

    DECLARE @daysNumber INT = (convert(INT, RAND() * 10) % 4) + 1

    WHILE(@daysNumber > 0)
      BEGIN
        DECLARE @capacity INT = (convert(INT, RAND() * 1000) % 51 + 30) -- od 30 do 80 uczestnikow
        DECLARE @dayDate DATE = DATEADD(DAY, @daysNumber, @date)

        EXEC dbo.PR_AddConferenceDay @confID, @dayDate, @capacity

        SET @daysNumber = @daysNumber - 1
      END

    SET @counter = @counter - 1
    SET @date = DATEADD(DAY, 15, @date)
  END
```

W tym przypadku dla każdego dnia konferencji wykonywaliśmy określony algorytm. Najpierw losowaliśmy liczbę dni konferencji z zakresu od 1 do 4 (a więc średnio 2,5 dnia dla jednej konferencji). Następnie losowaliśmy liczbę uczestników danego dnia z zakresu od 30 do 80. Początkowo datę w głównej pętli ustawiliśmy na 1.01.2016 i zwiększaliśmy o 15 dni z każdą iteracją. Daty dni jednej konferencji ustawialiśmy jako kolejne dni począwszy od aktualnej daty z pętli głównej. Uzyskaliśmy w ten sposób 177 dni konferencji.

9.1.4 Tabela DayPrices

```
-- DayPrices

DECLARE @counter INT = (SELECT count(*) FROM ConferenceDays)
WHILE (@counter > 0)
```

```
BEGIN

  DECLARE @confDayID INT = (
    SELECT TOP 1 ConferenceDayID
    FROM ConferenceDays
    WHERE ConferenceDayID NOT IN (
      SELECT ConferenceDayID
      FROM DayPrices
    )
    ORDER BY ConferenceDayID
  )

  DECLARE @thresholdsNumber INT = (convert(INT, RAND() * 10) % 4) + 1
                    -- od 1 do 4 progow cenowych
  DECLARE @discount DECIMAL(3, 2) = convert(DECIMAL(3, 2), RAND() / 2 + 0.1)
                    -- od 10 do 60 % znizki
  DECLARE @price MONEY = convert(MONEY, convert(INT, RAND() * 1000) % 501 + 300)
                    -- od 300 do 800 zl pelnej ceny
  DECLARE @date DATE = (
        SELECT Date
        FROM ConferenceDays
        WHERE ConferenceDayID = @confDayID
      )

  WHILE(@thresholdsNumber > 0)
    BEGIN
      DECLARE @startDate DATE = DATEADD(WEEK, (-1) * @thresholdsNumber * 2, @date)
      DECLARE @thPrice MONEY = convert(MONEY, @price / @thresholdsNumber)
      EXEC dbo.PR_AddDayPrice @confDayID, @startDate, @thPrice , @discount
      SET @thresholdsNumber = @thresholdsNumber - 1
    END

  SET @counter = @counter - 1
END
```

Generując dane do tej tabeli także iterowaliśmy się kolejno po wszystkich dniach konferencji. Dla każdego dnia losowaliśmy liczbę progów cenowych z zakresu od 1 do 4. Następnie losowaliśmy zniżkę studencką obowiązującą we wszystkich progach. Na końcu wyznaczaliśmy pełną cenę dnia konferencji i dodawaliśmy odpowiednią liczbę progów do bazy tak, że kolejne progi pojawiały się w dwutygodniowych odstępach, a cena obowiązująca w danym progu była określona wzorem [pełna cena] / [numer progu] (najpóźniejszy próg miał nr 1, poprzedni nr 2 itd.). Powyższy kod wygenerował 422 progi cenowe.

9.1.5 Tabela Workshops

```
-- Workshops


CREATE TABLE GEN_Workshops (
  ID INT IDENTITY (0, 1),
  Name1 VARCHAR(40),
  Name2 VARCHAR(40),
  Name3 VARCHAR(40)
)

INSERT INTO GEN_Workshops (Name1, Name2, Name3)
    VALUES
      ('Implementacja','baz danych', 'dla początkujących'),
      ('Analiza','kodu', '- Poziom Mistrzowski'),
      ('Projektowanie','systemów informatycznych', 'dla nowicjuszy'),
      ('Usuwanie','niechcianych stron', 'dla wytrwałych'),
      ('Przetwarzanie','danych','w kajakarstwie górskim'),
      ('Reorganizacja','przestrzeni dyskowej','w komputerze'),
```

```
      ('Rewitalizacja','monitorów','w łucznictwie olimpijskim'),
      ('Krytyka','przedmiotów','na studiach informatycznych'),
      ('Narzucanie','dobrych nawyków','w środowisku antyszczepionkowym'),
      ('Sprzedaż','postaci','w grach MMORPG')

DECLARE @workshopNames TABLE (
  Name VARCHAR(120)
)

INSERT INTO @workshopNames (Name) (
  SELECT
    p1.Name1 + ' ' + p2.Name2 + ' ' + p3.Name3              AS WorkshopName
  FROM GEN_Workshops AS p1
    CROSS JOIN GEN_Workshops AS p2
    CROSS JOIN GEN_Workshops AS p3
)


DECLARE @counter INT = (SELECT count(*) FROM ConferenceDays)
WHILE (@counter > 0)
  BEGIN

    DECLARE @confDayID INT = (
      SELECT TOP 1 ConferenceDayID
      FROM ConferenceDays
      WHERE ConferenceDayID NOT IN (
        SELECT ConferenceDayID
        FROM Workshops
      )
      ORDER BY ConferenceDayID
    )

    DECLARE @workshopsNumber INT = convert(INT, RAND() * 1000) % 5 + 2
                -- od 2 do 6 warsztatow dla danego dnia

    WHILE(@workshopsNumber > 0)
      BEGIN

        DECLARE @name VARCHAR(120) = (
          SELECT TOP 1 Name
          FROM @workshopNames
          WHERE Name NOT IN (
            SELECT WorkshopName
            FROM Workshops
          )
        )

        DECLARE @tmpDate DATETIME = DATEADD(hour, convert(INT, RAND() * 1000) % 11 + 8, '00:00:00')
        DECLARE @startTime TIME = convert(time(0), @tmpDate)            -- poczatek miedzy 8 a 18
        DECLARE @endTime TIME = convert(time(0),
                            DATEADD(HOUR, convert(INT, RAND() * 100) % 3 + 1, @tmpDate))
                                                                        -- trwa od 1 do 3 h
        DECLARE @price MONEY = convert(MONEY, convert(INT, RAND() * 1000) % 101) -- od 0 do 100 zl

        DECLARE @capacity INT = (
          SELECT MaxParticipantsNumber
          FROM ConferenceDays
          WHERE ConferenceDayID = @confDayID
        ) / (convert(INT, RAND() * 1000) % 5 + 2)    -- od 1/2 do 1/6 miejsc z dnia konferencji

        EXEC dbo.PR_AddWorkshop @confDayID, @name, @startTime, @endTime, @capacity, @price

        SET @workshopsNumber = @workshopsNumber - 1

       END
```

```
    SET @counter = @counter - 1
  END
```

Nazwy warsztatów wygenerowaliśmy poprzez utworzenie wielu kombinacji fragmentów nazw wpisanych ręcznie do tabeli pomocniczej. Następnie dla każdego dnia konferencji losowaliśmy liczbę powiązanych z nim warsztatów (od 2 do 6). Nazwę warsztatów wybieraliśmy z utworzonej tabeli nazw tak, aby nazwy w bazie się nie powtarzały. Dla danych warsztatów losowaliśmy ich cenę z zakresu od 0 do 100 zł. Godzinę rozpoczęcia warsztatów losowaliśmy z zakresu od 8:00 do 18:00, a czas trwania każdych warsztatów przyjmował wartość od 1 do 3 godzin. Finalnie otrzymaliśmy 730 różnych rekordów.

9.1.6 Tabela Reservations

```
-- Reservations

DECLARE @Count INT, @Reservations INT, @Date DATE, @ClientID INT
SET @Count = (Select COUNT(*) FROM Clients)
WHILE @Count > 0
  BEGIN
    SET @ClientID = (SELECT TOP 1 c.ClientID
                     FROM Clients c
                     WHERE ClientID
                     NOT IN(
                     SELECT r.ClientID
                     FROM Reservations r))
    SET @Reservations = CONVERT(INT,RAND() *1000) %3 +1
    IF(@Reservations = 1)
      BEGIN
        SET @Date = (SELECT DATEADD(DAY,CONVERT(INT,RAND()* 2000) % 730,'2016-01-01'))
        EXEC PR_AddReservation @ClientID, @Date
      END
    ELSE IF(@Reservations = 2)
      BEGIN
        SET @Date = (SELECT DATEADD(DAY,CONVERT(INT,RAND()* 1000) % 365,'2016-01-01'))
        EXEC PR_AddReservation @ClientID, @Date
        SET @Date = (SELECT DATEADD(DAY,CONVERT(INT,RAND()* 1000) % 365,
                                                DATEADD(DAY,365,'2016-01-01')))
        EXEC PR_AddReservation @ClientID, @Date
      END
    ELSE
      BEGIN
        SET @Date = (SELECT DATEADD(DAY,CONVERT(INT,RAND()* 1000) % 243,'2016-01-01'))
        EXEC PR_AddReservation @ClientID, @Date
        SET @Date = (SELECT DATEADD(DAY,CONVERT(INT,RAND()* 1000) % 243,
                                                DATEADD(DAY,243,'2016-01-01')))
        EXEC PR_AddReservation @ClientID, @Date
        SET @Date = (SELECT DATEADD(DAY,CONVERT(INT,RAND()* 1000) % 243,
                                                DATEADD(DAY,486,'2016-01-01')))
        EXEC PR_AddReservation @ClientID, @Date
      END
    SET @Count = @Count - 1
  END
```

Dla każdego klienta w bazie wylosowaliśmy z zakresu od 1 do 3 liczbę rezerwacji, które zostały mu przypisane. Następnie ustawiliśmy daty rezerwacji tak, aby były one losowe z granicach pewnych przedziałów czasowych (np. w przypadku wylosowania 2 rezerwacji datę pierwszej z nich losowaliśmy w obrębie roku 2016, a drugiej - 2017). W ten sposób wygenerowaliśmy 122 rezerwacje.

9.1.7 Tabela ConfDayReservations

```
-- ConfDayReservations

DECLARE @counter INT = (
    SELECT count(*)
    FROM Reservations
)
DECLARE @daysTab TABLE (
      ConfDayID INT
    )
WHILE(@counter > 0)
  BEGIN

    DECLARE @reservID INT = (
      SELECT ReservationID
      FROM Reservations
      ORDER BY ReservationID DESC
      OFFSET (@counter-1) ROWS FETCH NEXT 1 ROWS ONLY
    ) -- wybiera rezerwacje

    DECLARE @bookingDate DATE = (
      SELECT BookingDate
        FROM Reservations
        WHERE ReservationID = @reservID
    )

    DECLARE @randRow INT = convert(INT, RAND() * 10000) % (
      SELECT count(*)
      FROM Conferences
      WHERE dbo.FN_GetConferenceStartDate(ConferenceID)
        BETWEEN DATEADD(WEEK, 2, @bookingDate) AND DATEADD(WEEK, 8, @bookingDate)
    )

    DECLARE @confID INT = (
      SELECT ConferenceID
      FROM Conferences
      WHERE dbo.FN_GetConferenceStartDate(ConferenceID)
        BETWEEN DATEADD(WEEK, 2, @bookingDate) AND DATEADD(WEEK, 8, @bookingDate)
      ORDER BY ConferenceID
      OFFSET @randRow ROWS FETCH NEXT 1 ROWS ONLY
    )   -- wybiera losowa konferencje z tych, ktore odbeda sie w najblizszych 2 miesiacach

    DELETE FROM @daysTab

    INSERT INTO @daysTab (ConfDayID) (
      SELECT ConferenceDayID
      FROM ConferenceDays AS cd
      WHERE cd.ConferenceID = @confID
    )   -- wybiera wszystkie dni konferencji

    DECLARE @dayCounter INT = (
      SELECT count(*)
      FROM @daysTab
    )

    WHILE(@dayCounter > 0)
      BEGIN

        IF((convert(INT, RAND() * 10) % 2) = 1)
                            -- losuje czy skladac rezerwacje na dany dzien konferencji
          BEGIN
            DECLARE @confDayID INT = (
              SELECT ConfDayID
              FROM @daysTab
              ORDER BY ConfDayID DESC
              OFFSET (@dayCounter-1) ROWS FETCH NEXT 1 ROWS ONLY
            )
```

```
            IF(dbo.FN_GetConfDayFreePlaces (@confDayID) > 0)
              BEGIN

                DECLARE @participantsNumber INT = convert(INT, RAND() * 1000) % (
                  dbo.FN_GetConfDayFreePlaces (@confDayID)
                ) + 1     -- losuje liczbe rezerwowanych miejsc z zakresu wszystkich wolnych

                DECLARE @studentsNumber INT = @participantsNumber * (
                  convert(INT, RAND() * 10000) % 10 + 1
                ) / 10
                      -- losuje liczbe studentow jako ulamek liczby uczestnikow (od 1/10 do 10/10)

                SET @studentsNumber = @studentsNumber * (convert(INT, RAND() * 1000) % 2)
                           -- losuje czy beda jacykolwiek studenci

                EXEC dbo.PR_AddConfDayReservation @reservID, @confDayID,
                                          @participantsNumber, @studentsNumber
              END

          END
        SET @dayCounter = @dayCounter - 1
      END

    SET @counter = @counter - 1
  END
```

Dla każdej rezerwacji z bazy wylosowaliśmy jedną konferencję, która zaczyna się od 2 tygodni do 2 miesięcy od daty złożenia rezerwacji. Następnie dla każdego dnia konferencji losowaliśmy, czy w ogóle składać rezerwację na ten dzień (średnio na co drugi dzień konferencji została złożona rezerwacja). Liczba uczestników rezerwacji jest losowa w obrębie wolnych miejsc, natomiast liczba studentów stanowi losowy ułamek ilości uczestników (w połowie przypadków nie ma w ogóle studentów). Taki generator utworzył 149 rezerwacji o łącznej liczbie 3259 uczestników.

9.1.8 Tabela WorkshopReservations

```
-- WorkshopReservations

DECLARE
@ConfCount INT,
@Coin INT,
@WorkCount INT,
@ConfDayResID INT,
@WorkshopID INT,
@ParticipantsNumber INT,
@MaxPartNum INT
SET @ConfCount = (SELECT COUNT(*) FROM ConfDayReservations)
WHILE @ConfCount > 0
  BEGIN
    SET @ConfDayResID = ( SELECT ConfDayReservationID
                          FROM ConfDayReservations
                          ORDER BY ConfDayReservationID DESC
                          OFFSET (@ConfCount-1) ROWS FETCH NEXT 1 ROWS ONLY)

    SET @WorkCount = (SELECT COUNT(t.WorkshopID)
                        FROM (SELECT w.WorkshopID
                          FROM Workshops w
                          WHERE w.ConferenceDayID = (SELECT c.ConferenceDayID
                            FROM ConfDayReservations c
                            WHERE ConfDayReservationID=@ConfDayResID )) t)
    WHILE @WorkCount > 0
      BEGIN
```

```
        SET @WorkshopID = ( SELECT t.WorkshopID
                            FROM (SELECT w.WorkshopID
                              FROM Workshops w
                              WHERE w.ConferenceDayID = (SELECT c.ConferenceDayID
                                FROM ConfDayReservations c
                                WHERE ConfDayReservationID=@ConfDayResID )) t
                            ORDER BY t.WorkshopID DESC
                            OFFSET (@WorkCount-1) ROWS FETCH NEXT 1 ROWS ONLY)
        SET @Coin = CONVERT(INT, RAND()*100) % 2
        IF (@Coin=1)
          BEGIN
          SET @MaxPartNum = (SELECT ParticipantsNumber
                             FROM ConfDayReservations
                             WHERE ConfDayReservationID=@ConfDayResID)
          IF (@MaxPartNum > dbo.FN_GetWorkshopFreePlaces(@WorkshopID))
            SET @MaxPartNum = dbo.FN_GetWorkshopFreePlaces(@WorkshopID)
          IF(@MaxPartNum > 0)
            BEGIN
              SET @ParticipantsNumber = CONVERT(INT, RAND()*1000) % @MaxPartNum + 1
              EXEC PR_AddWorkshopReservation @WorkshopID, @ConfDayResID, @ParticipantsNumber
            END
          END
        SET @WorkCount = @WorkCount -1
      END
    SET @ConfCount = @ConfCount - 1
  END
```

Dla każdego dnia konferencji wybieraliśmy listę związanych z nim warsztatów, a następnie dla losowo co drugich warsztatów składaliśmy rezerwację. Liczba uczestników dla każdej rezerwacji była losowana w obrębie minimum z liczby uczestników dla powiązanej rezerwacji na dzień konferencji oraz maksymalnej liczby uczestników danego warsztatu. Kod wygenerował 292 rezerwacje o łącznej liczbie 1918 uczestników.

9.1.9 Tabela ConferenceParticipants

```
-- ConferenceParticipants

DECLARE @counter INT = (
    SELECT count(*)
    FROM ConfDayReservations
)
DECLARE @participantsNumber INT = (
      SELECT count(*)
      FROM Participants
    )
WHILE(@counter > 0)
  BEGIN

    DECLARE @confDayResID INT = (
      SELECT ConfDayReservationID
      FROM ConfDayReservations
      ORDER BY ConfDayReservationID DESC
      OFFSET (@counter-1) ROWS FETCH NEXT 1 ROWS ONLY
    ) -- wybiera rezerwacje na dzien konferencji

    DECLARE @students INT = (
      SELECT StudentsNumber
      FROM ConfDayReservations
      WHERE ConfDayReservationID = @confDayResID
    )

    DECLARE @participants INT = (
      SELECT ParticipantsNumber
      FROM ConfDayReservations
```

```
    WHERE ConfDayReservationID = @confDayResID
  )

  WHILE(@participants > 0)
    BEGIN

      DECLARE @randRow INT = convert(INT, RAND() * 10000) % @participantsNumber
      DECLARE @randParticipantID INT = (
          SELECT ParticipantID
          FROM Participants
          ORDER BY ParticipantID
          OFFSET @randRow ROWS FETCH NEXT 1 ROWS ONLY
      )

      WHILE(
        @randParticipantID IN (
          SELECT ParticipantID
          FROM ConferenceParticipants
          WHERE ConfDayReservationID = @confDayResID
        )
      )
      BEGIN
        SET @randRow = convert(INT, RAND() * 10000) % @participantsNumber
        SET @randParticipantID = (
          SELECT ParticipantID
          FROM Participants
          ORDER BY ParticipantID
          OFFSET @randRow ROWS FETCH NEXT 1 ROWS ONLY
      ) -- losuje nowego uczestnika do skutku
      END

      DECLARE @studentCardID CHAR(6) = NULL

      IF(@students > 0)
        BEGIN

          SET @studentCardID = substring(dbo.GEN_FN_GetRandomPesel (RAND()), 1, 6)

          IF EXISTS(
            SELECT StudentCardID
            FROM ConferenceParticipants
            WHERE ParticipantID = @randParticipantID AND StudentCardID IS NOT NULL
          )
          BEGIN
            SET @studentCardID = (
              SELECT TOP 1 StudentCardID
              FROM ConferenceParticipants
              WHERE ParticipantID = @randParticipantID AND StudentCardID IS NOT NULL
            )
          END -- jesli student ma juz w bazie nr legitymacji, to jest on odszukany i przypisany
        END

      EXEC dbo.PR_AddConferenceParticipant @confDayResID, @randParticipantID, @studentCardID

      SET @participants = @participants - 1
      SET @students = @students - 1
    END

    SET @counter = @counter - 1
  END
```

Dla każdej rezerwacji na dzień konferencji wybieraliśmy losowo odpowiednio dużo osób na miejsca normalne oraz studenckie (jeśli wylosowaliśmy osobę, która jest już zarejestrowana w danej rezerwacji - powtarzaliśmy losowanie). W przypadku dodawania nowego studenta najpierw szukaliśmy, czy nie został on już zarejestrowany na którąś konferencję jako student i jeśli

tak, pobieraliśmy z takiego rekordu numer legitymacji studenckiej. W przeciwnym wypadku generowaliśmy losowy ciąg sześciu cyfr na miejsce numeru legitymacji. Generator wygenerował 3259 uczestników dni konferencji, czyli zajął wszystkie zarezerwowane miejsca.

9.1.10 Tabela WorkshopParticipants

```
-- WorkshopParticipants

DECLARE @WorkshopResID INT,
@WorkshopResCount INT,
@PartNumber INT,
@SignedPartNum INT,
@ConfDayResID INT,
@Min INT,
@ConfPartID INT
SET @WorkshopResCount=(Select COUNT(*) FROM WorkshopReservations)
WHILE @WorkshopResCount>0
  BEGIN
    SET @WorkshopResID = (SELECT WorkshopReservationID
                          FROM WorkshopReservations
                          ORDER BY WorkshopReservationID DESC
                          OFFSET (@WorkshopResCount-1) ROWS FETCH NEXT 1 ROWS ONLY)
    SET @PartNumber =( SELECT ParticipantsNumber FROM WorkshopReservations
                                    WHERE WorkshopReservationID=@WorkshopResID)
    SET @ConfDayResID = (SELECT ConfDayReservationID FROM WorkshopReservations
                                    WHERE WorkshopReservationID=@WorkshopResID)
    SET @SignedPartNum = dbo.FN_GetConfDayReservationSignedParticipantsNumber(@ConfDayResID)
    IF(@PartNumber > @SignedPartNum)
      SET @Min = @SignedPartNum
    ELSE
      SET @Min = @PartNumber
    WHILE @Min >0
      BEGIN
        SET @ConfPartID = (SELECT ConferenceParticipantID
                          FROM ConferenceParticipants
                          WHERE ConfDayReservationID=@ConfDayResID
                          ORDER BY ConferenceParticipantID DESC
                          OFFSET (@Min-1) ROWS FETCH NEXT 1 ROWS ONLY)
        BEGIN TRY
        EXEC PR_AddWorkshopParticipant @WorkshopResID, @ConfPartID
        END TRY
        BEGIN CATCH
        END CATCH
        SET @Min = @Min - 1
      END
    SET @WorkshopResCount = @WorkshopResCount - 1
  END
```

Generowanie uczestników warsztatów polegało na pobraniu listy uczestników połączonego z warsztatem dnia konferencji, a następnie rejestrowanie ich na warsztat do momentu wypełnienia wszystkich miejsc z rezerwacji (lub przetworzenia całej listy). Z powodu oczywistej możliwości kolizji warsztatów, nie wszystkie miejsca zostały wypełnione (wyjątki rzucane przez triggery zostawały przechwytywane i ignorowane przez generator). Po dwukrotnym uruchomieniu generatora (za drugim razem z odwrotną kolejnością przeglądania potencjalnych uczestników warsztatów) udało się wygenerować 1907 niekolidujących ze sobą rekordów (a więc tylko 11 zarezerwowanych miejsc na warsztaty pozostało niezajętych).

9.1.11 Tabela Payments

```
DECLARE @Paid INT, @ReservationID INT, @ReservationCt INT,
@PaymentsNO INT, @PayStartDate DATE, @Step INT, @Pay MONEY
```

```
SET @ReservationCt = (SELECT COUNT(*)
                      FROM Reservations)
WHILE @ReservationCt>0
  BEGIN
    SET @ReservationID = (SELECT ReservationID
                          FROM Reservations
                          ORDER BY ReservationID DESC
                          OFFSET (@ReservationCt-1) ROWS FETCH NEXT 1 ROWS ONLY)
    SET @Paid = CONVERT(INT, RAND()*1000) % 50
    SET @PaymentsNO = CONVERT(INT, RAND()*100) % 3 + 1
    SET @PayStartDate = (SELECT BookingDate FROM Reservations Where @ReservationID=ReservationID)
    IF @Paid = 0
      BEGIN
        WHILE @PaymentsNO>0
          BEGIN
            SET @Step=CONVERT(INT, RAND()*100) %2+1
            SET @PayStartDate = DATEADD(DAY,@Step,@PayStartDate)
            SET @Pay = CONVERT(MONEY,
                           CONVERT(INT,RAND()*10000) %
                           (CONVERT(INT, dbo.FN_GetReservationTotalPrice(@ReservationID) -
                           dbo.FN_GetReservationAlreadyPaidAmmount(@ReservationID)) +1))
            BEGIN TRY
              EXEC PR_AddPayments @ReservationID, @PayStartDate, @Pay
            END TRY
            BEGIN CATCH
            END CATCH
            SET @PaymentsNO = @PaymentsNO -1
          END
      END
    ELSE
      BEGIN
        WHILE @PaymentsNO>1
          BEGIN
            SET @Step=CONVERT(INT, RAND()*100) %2+1
            SET @PayStartDate = DATEADD(DAY,@Step,@PayStartDate)
            SET @Pay = CONVERT(MONEY,
                               CONVERT(INT,RAND()*10000) % (CONVERT(INT,
                                      dbo.FN_GetReservationTotalPrice(@ReservationID)
                                      - dbo.FN_GetReservationAlreadyPaidAmmount(@ReservationID)) +1))
            BEGIN TRY
              EXEC PR_AddPayments @ReservationID, @PayStartDate, @Pay
            END TRY
            BEGIN CATCH
            END CATCH
            SET @PaymentsNO = @PaymentsNO -1
          END
        SET @Step=CONVERT(INT, RAND()*100) %2+1
        SET @PayStartDate = DATEADD(DAY,@Step,@PayStartDate)
        SET @Pay = (dbo.FN_GetReservationTotalPrice(@ReservationID)
                    - dbo.FN_GetReservationAlreadyPaidAmmount(@ReservationID))
        BEGIN TRY
          EXEC PR_AddPayments @ReservationID, @PayStartDate, @Pay
        END TRY
        BEGIN CATCH
        END CATCH
        SET @PaymentsNO = @PaymentsNO -1
      END
    SET @ReservationCt = @ReservationCt-1
  END
```

Dla każdej rezerwacji losowaliśmy liczbę rat w zakresie od 1 do 3. Następnie z prawdopodobieństwem 98 procent dodawaliśmy raty pokrywające pełną wartość rezerwacji (średnio 1 na 50 rezerwacji powinna zostać niecałkowicie opłacona). Terminy płatności losowaliśmy w obrębie tygodnia od daty złożenia rezerwacji. Generator utworzył 172 rekordy płatności, przy

czym jedna rezerwacja pozostała nieopłacona w całości.

10 Opis ról w systemie

10.1 Księgowy

Jego główną rolą jest wprowadzanie do bazy danych dotyczących płatności. Korzysta z widoków informujących o stanie płatności rezerwacji, w wypadku niedopłaty może anulować rezerwacje

10.2 Organizator

Ma dostęp do wszystkich funkcji związanych z organizacją konferencji jak na przykład dodawanie konferencji, jej dni, jak i realizowanych warsztatów. Dzięki funkcjom aktualizacji może również modyfikować wprowadzone wcześniej dane. Ma dostęp do widoków dotyczących statystyk oraz funkcji służących do tworzenia list osobowych.

10.3 Konsultant

Ma dostęp do widoków odpowiadających za kompletność rezerwacji, po kontakcie z klientem może uzupełniać odpowiednie dane uczestników jak i przyporządkowywać ich do odpowiednich warsztatów i dni konferencji.

10.4 Klient

Korzysta z interfejsu częściowo obsłużonego poprzez widoki związane z dostępnością rezerwacji. Ma możliwość dodawania do tabel związanych ze składaniem rezerwacji jak i ich modyfikacji. Z łatwością może anulować swoją rezerwację. Dodatkowo ma możliwość dodawania uczestników oraz przydzielania ich do swoich rezerwacji.