AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

PROJEKT “KONFERENCJE”

Podstawy Baz Danych

Marek Fudaliński i Julia Żur

1. Analiza wymagań

Konferencje

1) Dodawanie nowych konferencji jest możliwe tylko podając jej nazwę, datę rozpoczęcia, datę zakończenia, limit uczestników, zniżkę studencką oraz cenę za dzień.
2) Dodatkowo można dodać progi cenowe
3) Konferencje mogą być kilkudniowe jednak muszą być ciągłe w czasie
4) Konferencja może być anulowana na tydzień przed jej rozpoczęciem, w takim przypadku wszelkie dokonane płatności na jej rzecz są zwracane
5) Anulowanie konferencji oznacza jej całkowitą anulacje
6) Opłata za dzień konferencji jest zależna od daty rezerwacji
7) Każdy dzień konferencji ma taki sam limit uczestników

Rejestracja na konferencję:

1) Klientem może być firma lub osoba indywidualna
2) Uczestnikiem jest wyłącznie osoba
3) Rejestracja na daną konferencję jest możliwa tylko jeżeli jest odpowiednia ilość miejsc w przeciwnym razie rejestracja nie jest akceptowana.
4) Można dodawać nowych klientów.
5) Firma podczas rejestracji musi podać: NIP, adres, adres e-mail, telefon oraz nawę
6) Firma musi podać ilość rezerwowanych miejsc na odpowiednie dni konferencji oraz warsztaty oraz numery legitymacji ewentualnych studentów wraz z datami ich wygaśnięcia.

7) Uczestnik indywidualny musi podać: Imię, nazwisko, adres, e-mail,telefon a w przypadku studenta numer legitymacji oraz datę jej wygaśnięcia, warsztaty oraz dni konferencji,konferencje, w których chce uczestniczyć.
8) Jeżeli klient nie dokonał wpłaty w ciągu tygodnia od rejestracji skutkuje to anulacją rejestracji.
9) Firma ma obowiązek na 2 tygodnie przed konferencją dodać szczegółowe dane swoich uczestników. (klient indywidualne podaje swoje dane podczas rejestracji)
10) Na 2 tygodnie przed konferencją generowana jest lista firm która nie dostarczyła danych uczestników w celu skontaktowania się z firmą i uzyskania potrzebnych danych.

Warsztaty

1) Warsztaty można utworzyć podając: Przypisaną mu konferencję, nazwę warsztatu, czas trwania, maksymalną liczbę uczestników.
2) Należy podać kiedy dany warsztat się rozpoczyna oraz z jakim dniem konferencji jest powiązany, cenę warsztatu (w szczególności 0).
3) Cena warsztatów nie jest zależna od daty rezerwacji.
4) Różne warsztaty mogą odbywać się równocześnie.
5) Warsztaty mogą się powtarzać.
6) Warsztaty można anulować, wówczas o anulacji zostaje poinformowany klient oraz zwraca się mu wpłatę związaną z daną anulacją.

Rejestracja na warsztaty:

1) Uczestnik może zapisać się na wiele warsztatów z danej konferencji pod warunkiem, że na dane warsztaty są jeszcze wolne miejsca, warsztaty są w dniu konferencji w którym uczestniczy, nie uczestniczy już w takich samych warsztatach ani w warsztatach, które trwają w tym samym czasie.
2) Uczestnik może do tygodnia odwołać rezerwację na warsztat jeśli ich nie opłacił.
3) W przypadku braku opłaty warsztatów na tydzień przed ich rozpoczęciem rezerwacja jest anulowana.

Opłaty

1) Ceny warsztatów jest stała i nie podlega zniżkom.
2) Opłata dla pojedynczego uczestnika będącego studentem wyraża się wzorem:

(Day(Conferences.StartDate-Conferences.EndDate))*(1-Conferences.StudentsDiscount)*
*(1-Discounts.Discount)+WorkshopTerms.price

2) Opłata dla pojedynczego uczestnika nie będącego studentem wyraża się wzorem:

(Day(Conferences.StartDate-Conferences.EndDate))*(1-Discounts.Discount)+
+WorkshopTerms.price

3) Opłata dla firmy jest wyliczana na podstawie sumy opłat dla każdego z pracowników.
4) Klient jest informowany o opłacie w końcowej fazie rejestracji.

Raporty

1) Można wygenerować listę klientów (firmowych lub indywidualnych) (wraz lub bez sumarycznych opłat).
2) Można wygenerować listę uczestników na każdy dzień konferencji.
3) Można wygenerować listę uczestników na każde warsztaty.
4) Można wygenerować listę klientów którzy jeszcze nie dokonali płatności.
5) Można wygenerować fakturę dla każdego uczestnika lub firmy.
6) Można wygenerować listę klientów najczęściej uczestniczących w konferencjach.

2. Warunki Integralności

2.1 Default

1. getdate() - przy rezerwacji terminu - BookingDate
2. Bit isCancelled - default 0 przy tworzeniu konferencji
3. Bit isCancelled - default 0 przy tworzeniu warsztatów
4. Bit isCancelled - default 0 przy tworzeniu rezerwacji
5. EndTime - deafult StartTime + Duration
6. Discount dla konferencji - default 0
7. PaymentDate(Bookings) - default NULL
8. StudentsDiscount - default 0

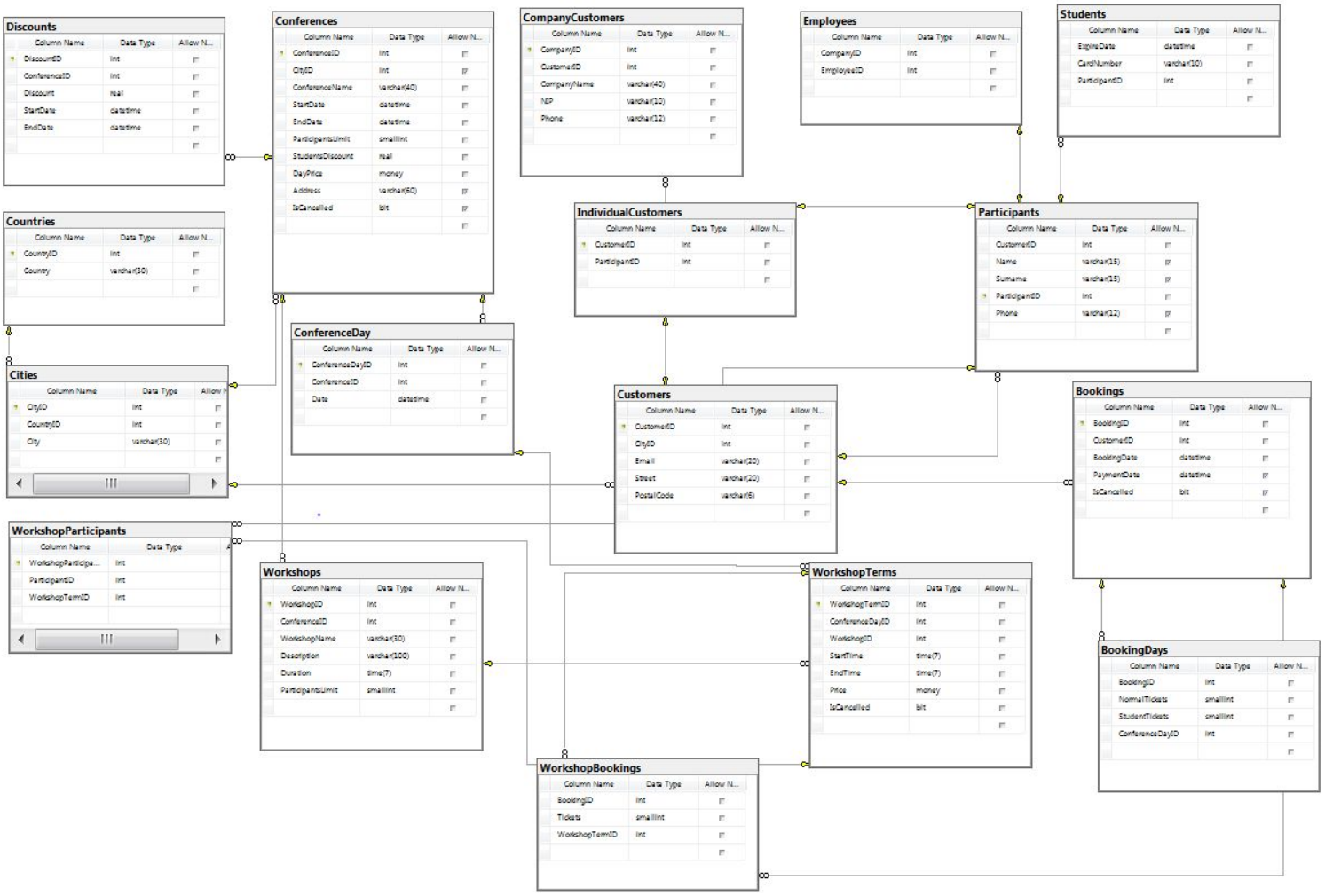
2.2 Unique

1. NIP - varchar(10) dokładnie 10
2. Country (tabela Countries)
3. City (tabela Cities)
4. Email (tabela Customers)
5. Phone (participants)
6. CardNumber(Students)
7. ConferenceID
8. WorkshopID

9. ConferenceDayID
10. CustomerID (wszystkie klucze główne)
11. ConferenceName

2.3 Check

1) Tickets (workshopBookings) (ma byc wiekszy lub rowny 0)
2) NormalTickets (BookingDays) (ma byc wieksze lub rowne 0)
3) StudentTickets (BookingDays) (ma byc wieksze lub rowne 0)
4) ParticipantsLimit(Worksop) (ma być większy od 0)
5) StartDate (tabela Conferences) (wiekszy od getDate())
6) EndDate (tabela Conferences) (wiekszy od StartDate)
7) Date (ConferenceDay) (większy lub równy StartDate (Conferences) oraz mniejszy lub równy EndDate(Conferences))
8) ExpireDate (Students) wiekszy od EndDate(Conferences)
9) EndTime większy niż StartTime (WorkshopTerms)
10) BookingDate (Bookings) równy dacie dzisiejszej
11) EndDate (Discounts) Większy lub równy StartDate(Conferences) większy lub równy Start Date(Discounts) mniejszy lub równy EndDate(Conferences)
12) PaymentDate(Bookings) większy lub równy BookingDate(Bookings) albo jest nullem
13) Duration (workshop) wieksze od 0
14) Price(WorkshopTerms) wiekszy lub rowny 0
15) DayPrice(Conferences) wiekszy lub rowny 0
16) NIP (CompanyCustomers) ma dokładnie 10 cyfr
17) StudentsDiscount (conference) (większe lub równe 0)
18) Discount (Discounts) (większe lub równe 0)
19) Email(Customers) ma mieć w środku @
20) BookingDate - PaymentDate <= 7 - anulowanie rezerwacji po przekroczeniu 7 dni



3. Tabele wraz z kodem utworzenia i opisem

3.1 Tabela Bookings

Tabela Bookings posiada klucz główny BookingID,
kolumne BookingDate zawierającą date dokonania rezerwacji,
kolumne PaymentDate zawierającą date dokonania zapłaty (Null oznacza brak zapłaty),
kolumne IsCancelled zwierająca informacje o anulowaniu rezerwacji,
klucz obcy CustomerID do tabeli Customers .

```
CREATE TABLE Bookings
(BookingID    INT       IDENTITY PRIMARY KEY,
 CustomerID   INT     NOT NULL
 CONSTRAINT Bookings_Customers_CustomerID_fk REFERENCES Customers,
 BookingDate DATETIME DEFAULT getdate() NOT NULL,
```

```
 PaymentDate DATETIME DEFAULT NULL) GO
CREATE UNIQUE INDEX Bookings_BookingID_uindex ON Bookings (BookingID)  GO
```

3.2 Tabela BookingDays

Tabela Bookings posiada klucz główny ConferenceDayID,

kolumne NormalTickets zaiwerającą ilość rezerwacji normalnych biletów,

kolumne StudentTickets zawierającą ilość rezerwacji studeckich biletów

klucz obcy BookingID do tabeli Bookings.

```
CREATE TABLE BookingDays
(ConferenceDayID INT NOT NULL PRIMARY KEY
CONSTRAINT BookingDays_ConferenceDay_ConferenceDayID_fk REFERENCES
ConferenceDay,
 BookingID       INT      NOT NULL
 CONSTRAINT BookingDays_Bookings_BookingID_fk REFERENCES Bookings,
 NormalTickets   SMALLINT NOT NULL,
 StudentTickets  SMALLINT NOT NULL)   GO
```

3.3 Tabela Cities

Tabela Cities zawiera klucz główny CityID,
kolumnę City zawierającą nazwę miasta,
klucz obcy CountryID odnoszący się do tabeli Countries.

```
CREATE TABLE Cities
(CityID    INT IDENTITY
CONSTRAINT Cities_CityID_pk PRIMARY KEY,
 CountryID INT         NOT NULL
 CONSTRAINT Cities_Countries_CountryID_fk REFERENCES Countries,
 City      VARCHAR(30) NOT NULL)   GO
CREATE UNIQUE INDEX Cities_CityID_uindex ON Cities (CityID)   GO
CREATE UNIQUE INDEX Cities_City_uindex ON Cities (City)   GO
```

3.4 Tabela CompanyCustomers

Tabela CompanyCustomers zawiera klucz główny CompanyID,
kolumnę NIP zawierającą numer NIP danej firmy,
kolumnę CompanyName zawierającą nazwę firmy,
kolumnę Phone zawierającą numer telefonu,
klucz obcy CustomerID

```
CREATE TABLE CompanyCustomers

(CompanyID   INT IDENTITY

CONSTRAINT CompanyCustomers_CompanyID_pk PRIMARY KEY,

 CustomerID  INT         NOT NULL

 CONSTRAINT CompanyCustomers_Customers_CustomerID_fk REFERENCES
Customers,

 CompanyName VARCHAR(40) NOT NULL,

 NIP         VARCHAR(10) NOT NULL)   GO

CREATE UNIQUE INDEX CompanyCustomers_NIP_uindex ON CompanyCustomers (NIP)
GO
```

3.5 Tabela ConferenceDay

Tabela ConferenceDay zawiera klucz główny ConferenceDayID,
klumnę Date zawierająca informację kiedy odbywa się dany dzień,
klucz obcy ConferenceID do tabeli Conferences.

```
CREATE TABLE ConferenceDay

(ConferenceDayID INT IDENTITY PRIMARY KEY,

 ConferenceID    INT  NOT NULL

 CONSTRAINT ConferenceDay_Conferences_ConferenceID_fk REFERENCES
Conferences,

 Date            DATETIME NOT NULL)   GO

CREATE UNIQUE INDEX ConferenceDay_ConferenceDayID_uindex ON ConferenceDay
(ConferenceDayID)   GO
```

3.6 Tabela Conferences

Tabela Conferences zawiera klucz główny ConferenceID,
kolumnę ConferenceName zawierającą nzwę danej konferencji,

kolumnę StartDate zawierająca datę rozpoczęcia danej konferencji,
kolumnę EndDate zawierającą datę zakończenia danej konferencji,
kolumnę PariticipantsLimit zawierającą informację o maksymalnej liczbie uczestników,
kolumnę StudentsDiscount zawierającą informację o zniżce studenckiej,
kolumnę DayPrice zawierającą cenę danego dnia konferencji,
kolumnę Address zawierającą adres danej konferencji,
kolumnę IsCancelled zawierająca informację o odwołaniu konferencji,
klucz obcy CityID do tabeli Cities.

```
CREATE TABLE Conferences
(ConferenceID      INT IDENTITY PRIMARY KEY,
 CityID            INT,
 CONSTRAINT Conferences_Cities_CityID_fk REFERENCES Cities,
 ConferenceName    VARCHAR(40)    NOT NULL,
 StartDate         DATETIME       NOT NULL,
 EndDate           DATETIME       NOT NULL,
 ParticipantsLimit SMALLINT       NOT NULL,
 StudentsDiscount  REAL DEFAULT 0 NOT NULL,
 DayPrice          MONEY          NOT NULL,
 Address           VARCHAR(60),
 IsCancelled       BIT DEFAULT 0  NOT NULL) GO

CREATE UNIQUE INDEX Conferences_ConferenceID_uindex ON Conferences
(ConferenceID) GO
CREATE UNIQUE INDEX Conferences_ConferenceName_uindex ON Conferences
(ConferenceName) GO
```

3.7 Tabela Countries

Zawierająca klucz główny CountryID,
kolumnę Country stanowiącą nazwę danego państwa.

```
CREATE TABLE Countries
(CountryID INT IDENTITY PRIMARY KEY,
 Country   VARCHAR(30) NOT NULL)   GO
CREATE UNIQUE INDEX Countries_Country_uindex ON Countries (Country) GO
```

3.8 Tabela Customers

Tabela Customers zawiera klucz główny CustomerID,
kolumnę Email zawierająca email kontaktowy do klienta,
kolumnę Street zawierającą adres danego klienta,
kolumnę PostalCode zawierającą kod pocztowy,
klucz obcy CityID do tabeli Cities.

```
CREATE TABLE Customers

(CustomerID INT IDENTITY

 CONSTRAINT Customers_CustomerID_pk PRIMARY KEY,

 CityID     INT         NOT NULL

 CONSTRAINT Customers_Cities_CityID_fk REFERENCES Cities,

 Email      VARCHAR(20) NOT NULL,

 Street     VARCHAR(20) NOT NULL,

 PostalCode VARCHAR(6)  NOT NULL)   GO

CREATE UNIQUE INDEX Customers_CustomerID_uindex ON Customers (CustomerID)
GO

CREATE UNIQUE INDEX Customers_Email_uindex ON Customers (Email)   GO
```

3.9 Tabela Discount

Tabela Discount zawiera klucz główny DiscountID
kolumnę Discount zawierającą próg zniżki,
kolumny StartDate oraz EndDate określającą ramy czasowe danej zniżki,
klucz obcy ConferenceID do tabeli Conferences.

```
CREATE TABLE Discounts
(DiscountID   INT IDENTITY PRIMARY KEY,
  ConferenceID INT        NOT NULL
  CONSTRAINT Discounts_Conferences_ConferenceID_fk REFERENCES
Conferences,
  Discount     REAL DEFAULT 0 NOT NULL,
  StartDate    DATETIME       NOT NULL,
  EndDate      DATETIME       NOT NULL)   GO
CREATE UNIQUE INDEX Discounts_DiscountID_uindex ON Discounts (DiscountID)
GO
```

3.10 Tabela Employees

Tabela zawiera klucz główny i obcy EmployeeID, który odnosi się do tabeli Participants
Klucz obcy CompanyID, który określa przynależność do firmy

```
CREATE TABLE Employees

(EmployeeID INT IDENTITY PRIMARY KEY

   CONSTRAINT Employees_Participants_ParticipantID_fk REFERENCES
Participants,

 CompanyID  INT NOT NULL

 CONSTRAINT Employees_CompanyCustomers_CompanyID_fk REFERENCES
CompanyCustomers) GO
```

3.11 Tabela IndividualCustomers

Tabela zawiera klucz główny i obcy CustomerID, który odnosi się do tabeli Customers
Klucz obcy ParticipantID, który powiązuje klientów z tabelą Participants

```
CREATE TABLE IndividualCustomers

(CustomerID    INT NOT NULL

 CONSTRAINT IndividualCustomers_CustomerID_pk PRIMARY KEY

 CONSTRAINT IndividualCustomers_Customers_CustomerID_fk REFERENCES
Customers,

 ParticipantID INT NOT NULL

 CONSTRAINT IndividualCustomers_Participants_ParticipantID_fk REFERENCES
Participants)   GO
```

3.12 Tabela Participants

Zawiera klucz główny ParticipantID
Klucz obcy CustomerID, odnośnik do tabeli Customers
Kolumnę Name - imię uczestnika
Kolumnę Surname - nazwisko uczestnika
Kolumnę Phone - numer kontaktowy do uczestnika
Klucz obcy BookingID, odnośnik do tabeli Bookings

```
CREATE TABLE Participants

(CustomerID    INT         NOT NULL

 CONSTRAINT Participants_Employees_EmployeeID_fk REFERENCES Employees

 CONSTRAINT Participants_IndividualCustomers_CustomerID_fk REFERENCES
IndividualCustomers,

 Name          VARCHAR(15) NOT NULL,

 Surname       VARCHAR(15) NOT NULL,

 ParticipantID INT IDENTITY

 CONSTRAINT Participants_ParticipantID_pk PRIMARY KEY,

 Phone         VARCHAR(12) NOT NULL,

 BookingID     INT

CONSTRAINT Participants_Bookings_BookingID_fk REFERENCES Bookings)  GO

 CREATE UNIQUE INDEX Participants_ParticipantID_uindex ON Participants
(ParticipantID)  GO

 CREATE UNIQUE INDEX Participants_Phone_uindex ON Participants (Phone)
GO
```

3.13 Tabela Students

Zawiera klucz główny i obcy ParticipantID, odnośnik do tabeli Participants
Kolumnę ExpireDate - datę ważności legitymacji
Kolumnę CardNumber - numer legitymacji

```
CREATE TABLE Students

(ParticipantID INT IDENTITY

 CONSTRAINT Students_ParticipantID_pk PRIMARY KEY

 CONSTRAINT Students_Participants_ParticipantID_fk REFERENCES
Participants,

 ExpireDate    DATETIME    NOT NULL,

CardNumber    VARCHAR(10) NOT NULL)   GO

CREATE UNIQUE INDEX Students_ParticipantID_uindex ON Students
(ParticipantID)   GO

CREATE UNIQUE INDEX Students_CardNumber_uindex ON Students (CardNumber)
GO
```

3.14 Tabela WorkshopBookings

Zawiera klucz główny i obcy WorkshopTermID, odnośnik do tabeli WorkshopTerms
Klucz obcy BookingID, odnośnik do tabeli Bookings
Kolumnę tickets - liczbę miejsc rezerwowamych

```
CREATE TABLE WorkshopBookings

(WorkshopTermID INT IDENTITY PRIMARY KEY

 CONSTRAINT WorkshopBookings_WorkshopTerms_WorkshopTermID_fk REFERENCES
WorkshopTerms,

 BookingID      INT      NOT NULL

 CONSTRAINT WorkshopBookings_Bookings_BookingID_fk REFERENCES Bookings,

 Tickets        SMALLINT NOT NULL)   GO

CREATE UNIQUE INDEX WorkshopBookings_WorkshopTermID_uindex  ON WorkshopBook
(WorkshopTermID)  GO
```

3.15 Tabela WorkshopParticipants

Zawiera klucz główny WorkshopParticipantID
Klucz obcy ParticipantID, odnośnik do Participants
Klucz obcy WorkshopTermID, odnośnik do WorkshopTerms

```
CREATE TABLE WorkshopParticipants

(WorkshopParticipantID INT IDENTITY PRIMARY KEY,

 ParticipantID         INT NOT NULL

CONSTRAINT WorkshopParticipants_Participants_ParticipantID_fk REFERENCES
Participants,

 WorkshopTermID        INT NOT NULL

 CONSTRAINT WorkshopParticipants_WorkshopTerms_WorkshopTermID_fk REFERENCES
WorkshopTerms)  GO

CREATE UNIQUE INDEX WorkshopParticipants_WorkshopParticipantID_uindex ON
WorkshopParticipants (WorkshopParticipantID)  GO
```

3.16 Tabela Workshop

Zawiera klucz główny WorkshopID
Klucz obcy ConferenceID, odnośnik do Conferences
Kolumnę WorkshopName - nazwę warsztatów
Kolumnę Description - opis warsztatów
Kolumnę Duration - czas trwania warsztatów

Kolumnę ParticipantsLimit - liczba miejsc

```
CREATE TABLE Workshops
(WorkshopID        INT IDENTITY PRIMARY KEY,
 ConferenceID      INT   NOT NULL
 CONSTRAINT Workshops_Conferences_ConferenceID_fk REFERENCES Conferences,
 WorkshopName      VARCHAR(30)  NOT NULL,
 Description       VARCHAR(100) NOT NULL,
 Duration          TIME         NOT NULL,
 ParticipantsLimit SMALLINT     NOT NULL)   GO
CREATE UNIQUE INDEX Workshops_WorkshopID_uindex ON Workshops (WorkshopID) G
CREATE UNIQUE INDEX Workshops_WorkshopName_uindex ON Workshops (WorkshopNam
```

3.17 Tabela WorkshopTerms

Zawiera klucz główny WorkshopTermID
Klucz obcy ConferenceDayID, odnośnik do tabeli ConferenceDays
Klucz obcy WorkshopID, odnośnik do tabeli Workshops
Kolumnę StartTime - czas rozpoczęcie warsztatów
Kolumnę EndTime - czas zakończenia warsztatów
Kolumnę Price - cenę warsztatów
Kolumnę IsCancelled - informację o anulowaniu warsztatów

```
CREATE TABLE WorkshopTerms
(WorkshopTermID  INT IDENTITY PRIMARY KEY,
 ConferenceDayID INT           NOT NULL
 CONSTRAINT WorkshopTerms_ConferenceDay_ConferenceDayID_fk REFERENCES
ConferenceDay,
 WorkshopID      INT           NOT NULL
 CONSTRAINT WorkshopTerms_Workshops_WorkshopID_fk REFERENCES Workshops,
 StartTime       TIME          NOT NULL,
 EndTime         TIME          NOT NULL  DEFAULT WorkshopTerms.StartTime +
Workshops.Duration,
 Price           MONEY         NOT NULL,
 IsCancelled     BIT DEFAULT 0 NOT NULL)  GO
```

```
CREATE UNIQUE INDEX WorkshopTerms_WorkshopTermID_uindex ON WorkshopTerms
(WorkshopTermID)  GO
```

4. Widoki

4.1 Anulowane konferencje

```
CREATE VIEW [dbo].[VIEW_CancelledConferences] as

 SELECT dbo.Conferences.ConferenceID,

        dbo.Conferences.ConferenceName,

        dbo.Conferences.StartDate,

        dbo.Conferences.EndDate,

        dbo.Conferences.ParticipantsLimit,

        CONCAT(dbo.Conferences.Address,' ', dbo.Cities.City,'
',dbo.Countries.Country) as Address

    FROM dbo.Conferences

      INNER JOIN dbo.Cities

        ON Conferences.CityID = Cities.CityID

      INNER JOIN dbo.Countries

        ON Cities.CountryID = Countries.CountryID

    WHERE dbo.Conferences.IsCancelled = 1
```

4.2 Anulowane terminy warsztatów

```
CREATE VIEW [dbo].[VIEW_CancelledWorkshops] AS

  SELECT  dbo.Workshops.WorkshopName,

          dbo.WorkshopTerms.StartTime,

          dbo.Workshops.Duration,

          dbo.WorkshopTerms.ConferenceDayID

  FROM dbo.WorkshopTerms

    INNER JOIN dbo.Workshops

        ON WorkshopTerms.WorkshopID = Workshops.WorkshopID

   WHERE dbo.WorkshopTerms.IsCancelled = 1
```

4.3 Nieopłacone oraz nieodwołane rezerwacje firm

```
CREATE VIEW  [dbo].[View_UnpaidCompanyBookings] AS
  SELECT  dbo.CompanyCustomers.CompanyName,
          dbo.CompanyCustomers.NIP,
          dbo.Bookings.BookingDate,
          dbo.Customers.Email,
          dbo.CompanyCustomers.Phone
  FROM dbo.Bookings
    INNER JOIN dbo.Customers
        ON dbo.Bookings.CustomerID = dbo.Customers.CustomerID
    INNER JOIN dbo.CompanyCustomers
        ON dbo.Customers.CustomerID = dbo.CompanyCustomers.CustomerID
  WHERE Bookings.PaymentDate is NULL
        and Bookings.IsCancelled = 0
```

4.4 Niezapłacone i nieodwołane rezerwacje osób indywidualnych

```
CREATE VIEW [dbo].[View_UnpaiIndyvidualBookings] AS
  SELECT  dbo.Participants.Name,
          dbo.Participants.Surname,
          dbo.Participants.Phone
  FROM dbo.Bookings
      INNER JOIN dbo.Customers
        ON dbo.Bookings.CustomerID = dbo.Customers.CustomerID
      INNER JOIN dbo.IndividualCustomers
        ON dbo.Customers.CustomerID = dbo.IndividualCustomers.CustomerID
      INNER JOIN dbo.Participants
        ON dbo.IndividualCustomers.ParticipantID =
dbo.Participants.ParticipantID
  WHERE dbo.Bookings.IsCancelled = 0
        and dbo.Bookings.PaymentDate is NULL
```

4.5 Firmy które do dnia dzisiejszego muszą zapłacić

```
CREATE VIEW  [dbo].[View_UnpaidCompanyToLastDay] AS
  SELECT  dbo.CompanyCustomers.CompanyName,
          dbo.CompanyCustomers.NIP,
          dbo.Bookings.BookingID,
          dbo.Customers.Email,
          dbo.CompanyCustomers.Phone
  FROM dbo.Bookings
    INNER JOIN dbo.Customers
        ON dbo.Bookings.CustomerID = dbo.Customers.CustomerID
    INNER JOIN dbo.CompanyCustomers
        ON dbo.Customers.CustomerID = dbo.CompanyCustomers.CustomerID
    WHERE Bookings.PaymentDate is NULL
          and Bookings.IsCancelled = 0
          and DATEDIFF(DAY,dbo.Bookings.BookingDate,getdate()) = 7
```

4.6 Rezerwacje osób indywidualnych, które w dniu dzisiejszym muszą zostać opłacone

```
CREATE VIEW [dbo].[View_UnpaidIndyviualToLastDay] AS
  SELECT  dbo.Participants.Name,
          dbo.Participants.Surname,
          dbo.Participants.Phone,
          dbo.Bookings.BookingID
  FROM dbo.Bookings
      INNER JOIN dbo.Customers
        ON dbo.Bookings.CustomerID = dbo.Customers.CustomerID
      INNER JOIN dbo.IndividualCustomers
        ON dbo.Customers.CustomerID = dbo.IndividualCustomers.CustomerID
      INNER JOIN dbo.Participants
        ON dbo.IndividualCustomers.ParticipantID =
dbo.Participants.ParticipantID
  WHERE dbo.Bookings.IsCancelled = 0
```

```
        AND dbo.Bookings.PaymentDate is NULL

        AND DATEDIFF(day,dbo.Bookings.BookingDate,getdate()) = 7
```

4.7 Liczba miejsc pozostałych na nieanulowane warsztaty

```
CREATE VIEW [dbo].[View_FreePlacesOnWorkshopTerms] AS

  SELECT dbo.Workshops.WorkshopName,

        dbo.WorkshopTerms.WorkshopTermID,

        dbo.WorkshopTerms.StartTime,

        dbo.Workshops.ParticipantsLimit,

        SUM(dbo.WorkshopBookings.Tickets) as TicketsReserved,

        dbo.Workshops.ParticipantsLimit - SUM(dbo.WorkshopBookings.Tickets)
as TicketsLeft

  FROM dbo.WorkshopTerms

    INNER JOIN dbo.Workshops

      ON dbo.WorkshopTerms.WorkshopID = dbo.Workshops.WorkshopID

    INNER JOIN dbo.WorkshopBookings

      ON dbo.WorkshopTerms.WorkshopTermID =
dbo.WorkshopBookings.WorkshopTermID

    INNER JOIN dbo.Bookings

      ON dbo.WorkshopBookings.BookingID = dbo.Bookings.BookingID and
dbo.Bookings.IsCancelled = 0

  WHERE dbo.WorkshopTerms.IsCancelled = 0

  GROUP BY
dbo.Workshops.WorkshopName,dbo.WorkshopTerms.WorkshopTermID,dbo.WorkshopTer
ms.StartTime,dbo.Workshops.ParticipantsLimit
```

4.8 Liczba miejsc zarezerwowanych na dany dzień konferencji

```
CREATE VIEW [dbo].[View_FreePlacesOnConferenceDay] AS

  SELECT  dbo.Conferences.ConferenceName,

          dbo.ConferenceDay.Date,

          dbo.Conferences.ParticipantsLimit,

          SUM(dbo.BookingDays.NormalTickets) +
SUM(dbo.BookingDays.StudentTickets) as TicketsBooked,
```

```
          dbo.Conferences.ParticipantsLimit -
(SUM(dbo.BookingDays.NormalTickets) + SUM(dbo.BookingDays.StudentTickets))
as TicketsLeft

  FROM dbo.ConferenceDay

  INNER JOIN dbo.Conferences

      ON ConferenceDay.ConferenceID = Conferences.ConferenceID

  INNER JOIN dbo.BookingDays

      ON dbo.BookingDays.ConferenceDayID =
dbo.ConferenceDay.ConferenceDayID

  INNER JOIN dbo.Bookings

      on dbo.BookingDays.BookingID = dbo.Bookings.BookingID

      and dbo.Bookings.IsCancelled = 0

  GROUP BY dbo.Conferences.ConferenceName, dbo.ConferenceDay.Date,
dbo.Conferences.ParticipantsLimit
```

4.9 Klienci indywidualni oraz liczba ich skutecznych rezerwacji

```
CREATE VIEW [dbo].[View_IndyvidualClientsSucceedBookings] as

  SELECT dbo.Participants.Name,

          dbo.Participants.Surname,

          dbo.Participants.Phone,

          dbo.Customers.Email,

          SUM(Bookings.BookingID) as TimesAttended

    FROM dbo.Customers

  INNER JOIN dbo.Bookings

        ON dbo.Customers.CustomerID = dbo.Bookings.CustomerID

            AND IsCancelled = 0

  INNER JOIN dbo.IndividualCustomers

        ON dbo.Customers.CustomerID = dbo.IndividualCustomers.CustomerID

  INNER JOIN dbo.Participants

        ON IndividualCustomers.ParticipantID = Participants.ParticipantID

  GROUP BY dbo.Customers.CustomerID , dbo.Participants.Name,
 dbo.Participants.Surname,
 dbo.Participants.Phone,dbo.Customers.Email
```

4.10 Widok wypisujący firmy oraz ich skuteczne rezerwacje

```
CREATE VIEW [dbo].[View_CompanyClientsSucceedBookings] AS

  SELECT dbo.CompanyCustomers.CompanyName,

        dbo.CompanyCustomers.Phone,

        dbo.Customers.Email,

        SUM(dbo.BookingDays.NormalTickets + dbo.BookingDays.StudentTickets)
as TicketsBought

  FROM dbo.Customers

  INNER JOIN dbo.CompanyCustomers

      ON dbo.Customers.CustomerID = dbo.CompanyCustomers.CustomerID

  INNER JOIN dbo.Bookings

      ON dbo.Customers.CustomerID = dbo.Bookings.CustomerID AND
dbo.Bookings.IsCancelled = 0

  INNER JOIN dbo.BookingDays

      ON dbo.Bookings.BookingID = dbo.BookingDays.BookingID

  GROUP BY dbo.Customers.CustomerID, dbo.CompanyCustomers.CompanyName,
dbo.CompanyCustomers.Phone,dbo.Customers.Email
```

5. Procedury

5.1 Procedury dot. konferencji

5.1.1 Dodawanie konferencji

```
CREATE PROCEDURE  [dbo].AddConference
 @CoferenceName VARCHAR(40),
 @StartDate DATETIME ,
 @EndDate DATETIME ,
 @Limit SMALLINT,
 @SDiscount REAL,
 @Price MONEY
 AS
 BEGIN
   SET NOCOUNT ON
   BEGIN TRY
    if(DATEDIFF(day,GETDATE(),@StartDate)<0 or @StartDate >
@EndDate)
       BEGIN
         ;THROW 52000,
         'blad podawania daty',
         1;
       END
     INSERT INTO Conferences(
      ConferenceName,
      StartDate,
      EndDate,
      ParticipantsLimit,
      StudentsDiscount,
      DayPrice)
     VALUES (
       @CoferenceName,
       @StartDate,
       @EndDate,
       @Limit,
       @SDiscount,
       @Price
     )
     DECLARE @ConferenceIdd INT;
     SET  @ConferenceIdd = scope_identity();
     Declare @counter INT =1;
     while 0 <=(datediff(day,@StartDate,@EndDate))
       BEGIN
         EXECUTE AddConferenceDay @ConferenceIdd, @StartDate
         Set @StartDate = DATEADD(DAY,1,@StartDate)
       END
   END TRY
   BEGIN CATCH
```

```
     DECLARE @err NVARCHAR(2048) = 'Błąd dodania konferencji' +
ERROR_MESSAGE();
     ;THROW 52000, @err, 1
   END CATCH
 END
GO
```

5.1.2 Dodawanie dnia konferencji

```
CREATE PROCEDURE [dbo].[AddConferenceDay]
 @ConferenceID INT,
 @Date DATETIME
 AS
 BEGIN
   SET NOCOUNT ON
   BEGIN TRY
     IF NOT EXISTS(
       SELECT * FROM Conferences WHERE Conferences.ConferenceID =
@ConferenceID
     )
         BEGIN
              ;THROW 52000, 'nie ma podaje konferencji' ,1;
         END
     IF EXISTS(
         SELECT * FROM ConferenceDay
         WHERE ConferenceID = @ConferenceID and Date = @Date
     )
         BEGIN
              ;THROW 52000, 'juz przypisano ten dzien do tej
konferencji' ,1;
         END
     INSERT INTO ConferenceDay
     (
       ConferenceID,
       Date
     )
     VALUES
     (
       @ConferenceID,
       @Date
     )
   END TRY
   BEGIN CATCH
     DECLARE @err NVARCHAR(2048) = 'Nie mozna dodac konferencji' +
ERROR_MESSAGE();
     ;THROW 52000,@err,1;
   END CATCH
 END
GO
```

5.1.3 Dodawanie adresu konferencji

```
CREATE PROCEDURE [dbo].[AddConferenceAddress]
     @ConferenceName VARCHAR(40),
     @City VARCHAR(30),
     @Adress VARCHAR(60),
     @Country VARCHAR(30)
   AS
   BEGIN
     SET NOCOUNT ON;
     BEGIN
       --  DECLARE @CountryIDD INT;
       --  EXECUTE FindAddCountryID
       --   @Country, @CountryID = @CountryIDD OUTPUT;

       DECLARE @CityIDD INT;
       EXECUTE FindCityID
           @City, @Country ,@cityID = @CityIDD OUTPUT;

       if (@CityIDD is NULL ) EXECUTE AddCity
@City,@Country,@cityID = @CityIDD OUTPUT;

       --juz na 100 mamy wstawione nasze miasteczko teraz musimy
zmodyfikowac nasz wiersz w conferences
       --uzywamy UPDATE
       UPDATE Conferences
         SET Address = @Adress,CityID = @CityIDD
         WHERE ConferenceName = @ConferenceName;
   END
     END
GO
```

5.1.4 Anulowanie konferencji

```
CREATE PROCEDURE [dbo].[CancelConference]
      @ConferenceName VARCHAR(40)
    AS
    BEGIN
      SET NOCOUNT ON;
      BEGIN
        --uzywamy UPDATE
       DECLARE @IsC BIT;
        SELECT @IsC = IsCancelled FROM Conferences WHERE
@ConferenceName = ConferenceName;
        IF @IsC = 0
            UPDATE Conferences
            SET IsCancelled = 1 WHERE ConferenceName =
@ConferenceName;

            UPDATE WorkshopTerms
            SET IsCancelled=1 WHERE WorkshopTermID IN (SELECT
```

```
WorkshopTerms.WorkshopTermID FROM WorkshopTerms
            INNER JOIN Workshops ON WorkshopTerms.WorkshopID =
Workshops.WorkshopID
            INNER JOIN Conferences ON Workshops.ConferenceID =
Conferences.ConferenceID WHERE @ConferenceName=ConferenceName)
            UPDATE Bookings
            SET IsCancelled=1 WHERE BookingID IN (SELECT
Bookings.BookingID FROM Conferences
            INNER JOIN ConferenceDay C2 ON
Conferences.ConferenceID = C2.ConferenceID
            INNER JOIN BookingDays B2 ON C2.ConferenceDayID =
B2.ConferenceDayID
            INNER JOIN Bookings B3 ON B2.BookingID = B3.BookingID
            WHERE ConferenceName=@ConferenceName)
      END
      END
GO
```

5.1.5 Dodawanie zniżki na konferencje - za wcześniejszą rezerwację

```
CREATE PROCEDURE [dbo].[AddConferenceDiscounts]
 @ConfernceN VARCHAR(40),
 @StartDate DATETIME,
 @EndDate DATETIME,
 @Discounts REAL
 AS
 BEGIN
   if (EXISTS(SELECT * FROM Conferences WHERE
@ConfernceN=Conferences.ConferenceName))
     BEGIN
       DECLARE @CIDD INT;--id konferencji
       SELECT @CIDD = ConferenceID FROM Conferences WHERE
@ConfernceN = ConferenceName;

       DECLARE @SCD DATETIME;--data startowa konferencji
       SELECT @SCD = StartDate FROM Conferences WHERE @ConfernceN
= ConferenceName;

       --musimy sprawdzic czy przedzialy na siebie nachodza
       IF (NOT EXISTS(SELECT * FROM Discounts WHERE @CIDD =
ConferenceID AND ((  datediff(DAY,StartDate,@StartDate) >=0 and
datediff(day,@StartDate,EndDate) >= 0 )
       OR(  datediff(DAY,StartDate,@EndDate) >=0 and
datediff(day,@EndDate,EndDate) >= 0     ))) and
datediff(DAY,@StartDate,@EndDate) >=0 )
       BEGIN
             INSERT INTO Discounts
             (ConferenceID,
              Discount,
              StartDate,
              EndDate) VALUES ( @CIDD,
```

```
                @Discounts,
                @StartDate,
                @EndDate)
          END
      END
 END
GO
```

5.2 Procedury dot. miast i państw

5.2.1 Znajdywanie/Wstawianie państwa

```
CREATE PROCEDURE [dbo].[FindAddCountryID]
   @CountryName VARCHAR(30),
   @CountryID int OUTPUT
 AS
   SET NOCOUNT ON;
     SELECT @CountryID = CountryID FROM Countries WHERE
@CountryName = Country
     IF (@CountryID is NULL ) BEGIN INSERT INTO Countries
(Country) VALUES (@CountryName)  SET @CountryID = @@IDENTITY END
     PRINT (CONVERT (INT, @CountryID))
 RETURN
GO
```

5.2.2 Znajdywanie indeksu miasta

```
 CREATE PROCEDURE [dbo].[FindCityID]
      @City VARCHAR(30),
      @cityID int OUTPUT
    AS
    BEGIN
      SET NOCOUNT ON;
      BEGIN
        SET @cityID = NULL
        SET @cityID = (SELECT CityID FROM Cities WHERE City =
@City)
        RETURN
    END
GO
```

5.2.3 Dodawanie miasta

```
CREATE PROCEDURE [dbo].[AddCity]
   @City VARCHAR(30),
   @Country VARCHAR(30),
   @CityID int OUTPUT
```

```
 AS
 BEGIN
   SET NOCOUNT ON;
   DECLARE @CountryIDD INT;
   EXECUTE FindAddCountryID
           @Country, @CountryID = @CountryIDD OUTPUT;
   PRINT @CountryIDD
   INSERT INTO Cities(
     City,
     CountryID
   ) VALUES (
     @City,
     @CountryIDD
   )
   SET @CityID = @@IDENTITY
   RETURN
 END
GO
```

5.3 Procedury dot. klientów

5.3.1 Dodawanie klienta

```
CREATE PROCEDURE [dbo].[AddCustomer]
 @CityName VARCHAR(30),
 @Country VARCHAR(30),
 @Email VARCHAR(20),
 @Street VARCHAR(20),
 @PostalCode VARCHAR(6),
 @CustomerId int OUTPUT
 AS
BEGIN
 SET NOCOUNT ON;
 BEGIN
   DECLARE @CityIDD INT;
   EXECUTE FindCityID
           @CityName,@Country, @cityID = @CityIDD OUTPUT;
     IF (@CityIDD is NULL ) EXECUTE AddCity @CityName, @Country,
@CityID = @CityIDD OUTPUT;
   INSERT INTO Customers (
     CityID,
     Email,
     Street,
     PostalCode) VALUES (
     @CityIDD,
     @Email,
     @Street,
     @PostalCode
   ) SET @CustomerId = scope_identity();
```

```
   RETURN
 END
END
GO
```

5.3.2 Dodanie klienta - firmy

```
CREATE PROCEDURE [dbo].[AddCompanyCustomer]
 @City VARCHAR(30),
 @Country VARCHAR(30),
 @Email VARCHAR(20),
 @Street VARCHAR(20),
 @PostalCode VARCHAR(6),
 @CompanyName VARCHAR(40),
 @Nip VARCHAR(10),
 @Phone VARCHAR(12)
 AS
 BEGIN
   DECLARE @Id INT
   EXECUTE AddCustomer @City, @Country, @Email, @Street,
@PostalCode, @CustomerId = @Id OUTPUT
   INSERT INTO CompanyCustomers(
     CustomerID,
     CompanyName,
     NIP,
     Phone
   ) VALUES (
     @Id,
     @CompanyName,
     @Nip,
     @Phone
   )
 END
GO
```

5.3.3 Dodanie klienta - osoby indywidualnej

```
CREATE PROCEDURE [dbo].[AddIndyvidualCustomer]
 @name VARCHAR(15),
 @surname VARCHAR(15),
 @Phone VARCHAR(12),
 @Email VARCHAR(20),
 @Street VARCHAR(20),
 @Postalcode VARCHAR(6),
 @City VARCHAR(30),
 @Country VARCHAR(30)
 AS
 BEGIN
     SET NOCOUNT ON;
     BEGIN Try
       BEGIN TRAN ADD_ICustomer
```

```
          DECLARE @IdC INT;
          EXECUTE AddCustomer @CityName = @City ,@Country =
@Country,
                        @Email = @Email, @Street = @Street ,
                        @PostalCode = @Postalcode,@CustomerId =
@IdC OUTPUT;
            INSERT INTO Participants (
              CustomerID,
              Name,
              Surname,
              Phone) VALUES (
              @IdC,
              @name,
              @surname,
              @Phone
            ) DECLARE @IdP INT; Set @IdP = SCOPE_IDENTITY();
              INSERT INTO IndividualCustomers (
              CustomerID,
              ParticipantID) VALUES (
              @IdC,
              @IdP)
        COMMIT TRAN ADD_ICustomer
      END TRY
      BEGIN CATCH
        ROLLBACK TRAN ADD_ICustomer
      END CATCH
 END
GO
```

5.3.4 Dodawanie klienta - pracownika firmy

```
CREATE PROCEDURE [dbo].[AddEmployee]
 @Name VARCHAR(15),
 @Surname VARCHAR(15),
 @phone VARCHAR(12),
 @companyID INT,
 @CardNumber VARCHAR(10),
 @ExpireDate DATETIME
 AS
 BEGIN
   SET NOCOUNT ON;
     BEGIN Try
       BEGIN TRAN ADD_Employee
         DECLARE @EID INT;
         DECLARE @CuID INT;
         SELECT @CuID=CustomerID FROM CompanyCustomers WHERE
CompanyID=@companyID;
          INSERT INTO Participants (
            CustomerID, Name, Surname, Phone) VALUES (
```

```
            @CuID,
            @Name,
            @Surname,
            @phone
          ) SET @EID = scope_identity(); PRINT @EID;
        INSERT INTO Employees (CompanyID,EmployeeID)VALUES (
          @companyID,
          @EID
        )
        if(@CardNumber is not null)
        BEGIN
          INSERT INTO Students (ParticipantID,ExpireDate,
CardNumber) VALUES
            (@EID,
              @ExpireDate,
              @CardNumber
            )
        END
        COMMIT TRAN ADD_Employee
      END TRY
      BEGIN CATCH
        ROLLBACK TRAN ADD_Employee
      END CATCH
 END
GO
```

5.4 Procedury dot. studentów

5.4.1 Dodanie numeru legitymacji studenckiej

```
CREATE PROCEDURE [dbo].[AddStudentCard]
   @ParticipantID INT,
   @Cardnumber VARCHAR(10),
   @ExpireDate DATETIME
   AS
   BEGIN
     IF NOT EXISTS(SELECT * FROM Students where ParticipantID =
@ParticipantID)
       BEGIN
           INSERT INTO Students (
             ParticipantID,
             ExpireDate,
             CardNumber) VALUES (
             @ParticipantID,
             @Cardnumber,
             @ExpireDate)
         END

   END
```

```
GO
```

5.5 Procedury dot. rezerwacji

5.5.1 Dodanie rezerwacji

```
CREATE PROCEDURE [dbo].[AddBooking]
   @CustomerID INT,
   @BookingId INT OUTPUT
   AS
   BEGIN
     IF NOT EXISTS(SELECT BookingID FROM Bookings where CustomerID =
@CustomerID and BookingDate = getdate())
       BEGIN
           INSERT INTO Bookings (
             CustomerID,
             BookingDate,
             PaymentDate,
             IsCancelled) VALUES (
             @CustomerID,
             getdate(),
             NULL,
             0
           )
           SET @BookingId = scope_identity()
         END
       SELECT @BookingId=BookingID FROM Bookings where CustomerID =
@CustomerID and BookingDate = getdate();
       RETURN
   END
GO
```

5.5.2 Dodanie rezerwacji - osoba indywidualna

```
CREATE PROCEDURE [dbo].[AddIndyvidualSingleBooking]
 @phone VARCHAR(10),--tylko po nim moge znalezc ID klienta
 @Card VARCHAR(10), -- jezeli podane jako null to znaczy ze normalny
 @Expdate DATETIME,
 @ConferenceN VARCHAR(40),
 @ConferenceD DATETIME,
 @WorkshopN VARCHAR(30),
 @WStart TIME
 AS
 BEGIN
     DECLARE @Pid INT;
     SELECT @Pid=ParticipantID FROM Participants WHERE Phone =
@phone;
     DECLARE @Cid INT;
     SELECT @Cid= CustomerID FROM Participants WHERE Phone = @phone;
```

```
   print -2;
   PRINT @Expdate;
   if @Card is NOT null and NOT EXISTS(SELECT *FROM Students WHERE
@Card=CardNumber)
   BEGIN

     INSERT INTO Students (ExpireDate, CardNumber, ParticipantID)
VALUES (@Expdate,@Card,@Pid);
   END
   print -3;
     DECLARE @BId INT;
     EXECUTE AddBooking @CustomerID = @Cid, @BookingId = @BId
OUTPUT;--mamy juz nasza rezerwacje
PRINT 1;
     DECLARE @ConDID INT = (SELECT ConferenceDayID
                            FROM ConferenceDay
                               INNER JOIN  Conferences
                                 ON ConferenceDay.ConferenceID =
Conferences.ConferenceID AND ConferenceName = @ConferenceN
                                  WHERE
datediff(DAY,Date,@ConferenceD)= 0); --MAMY ID DNI KONFERENCJI NA
KTORY KTOS CHCE ISC
     DECLARE @datecd DATETIME;
     SELECT @datecd=Date FROM ConferenceDay WHERE ConferenceDayID =
@ConDID
PRINT 2;
     IF @ConDID IS NOT NULL
       BEGIN
         IF NOT EXISTS(SELECT * FROM BookingDays where
ConferenceDayID = @ConDID and BookingID = @BId)--jezeli nie dodal
jeszcze zadnej rezerwacji na ten dzien
           BEGIN
               IF (DATEDIFF(d,@Expdate,@datecd) > 0)
                 BEGIN
                   PRINT 3;
                    INSERT INTO BookingDays (ConferenceDayID,
BookingID, NormalTickets, StudentTickets)
                    VALUES (@ConDID,@BId,0,1)
                  END
              ELSE
                BEGIN
                  INSERT INTO BookingDays (ConferenceDayID,
BookingID, NormalTickets, StudentTickets)
                    VALUES (@ConDID,@BId,1,0)
                 END
                 --dodalismy zadana rezerwacje na dni teraz trzebaby
dodac warsztat
            END
           --mamy juz rezerwacje na dany dzien (stworzylismy albo byla
wczesniej) jest dokladnie jedna
```

```
            --teraz trzeba dopisac warsztaty mamy ich rozpoczecie oraz
nazwe
            DECLARE @WTId INT = (SELECT WorkshopTermID FROM
WorkshopTerms
                                 INNER JOIN Workshops On
WorkshopTerms.WorkshopID = Workshops.WorkshopID and WorkshopName =
@WorkshopN
                                 WHERE StartTime = @WStart)
            IF (@WTId is NOT NULL)
              BEGIN
                PRINT 4;
                INSERT INTO WorkshopBookings
(WorkshopTermID,BookingID, Tickets)
                  VALUES (@WTId,@BId,1)
              END
     END
 END
GO
```

5.5.3 Dodanie rezerwacji na dany dzień

```
CREATE PROCEDURE [dbo].[AddBookingDay]
   @CustomerIDD INT,
   @ConfName VARCHAR(40),
   @Date DATETIME,
   @NTick SMALLINT,
   @STick SMALLINT
   AS
   BEGIN
     --musimy sprawdzic czy jest sens dodawac rezerwacje
     if (EXISTS(SELECT * FROM Customers WHERE
CustomerID=@CustomerIDD) and
         EXISTS(SELECT * FROM Conferences WHERE
ConferenceName=@ConfName AND IsCancelled=0) AND
         DATEDIFF(DAY,(SELECT StartDate FROM Conferences WHERE
ConferenceName=@ConfName) ,@Date) >=0 AND
         DATEDIFF(DAY,(SELECT EndDate FROM Conferences WHERE
ConferenceName=@ConfName) ,@Date) <= 0  AND
         @NTick > 0 AND @STick >0 )
       BEGIN
         DECLARE @BID INT;
         EXECUTE [dbo].AddBooking @CustomerID =
@CustomerIDD,@BookingId = @BID OUTPUT;
         --mamy dodany booking
         DECLARE @CDID INT;
         SELECT @CDID = ConferenceDayID FROM ConferenceDay
           INNER JOIN Conferences ON ConferenceDay.ConferenceID =
Conferences.ConferenceID AND
             ConferenceName = @ConfName
```

```
         WHERE DATEDIFF(DAY,Date,@Date) = 0
         --mamy ID dnia konferencji
             if (NOT EXISTS(SELECT * FROM BookingDays WHERE
BookingID=@BID and ConferenceDayID=@CDID))
                BEGIN
                    INSERT INTO BookingDays
                     (ConferenceDayID,
                         BookingID,
                         NormalTickets,
                         StudentTickets) VALUES (
                       @CDID,
                       @BID,
                       @NTick,
                       @STick
                     )
                  END
       END
   END
GO
```

5.5.4 Anulowanie rezerwacji

```
CREATE PROCEDURE [dbo].[CancelBooking]
     @BookingID INT
   AS
   BEGIN
     DELETE WorkshopBookings WHERE BookingID = @BookingID
     DELETE BookingDays WHERE BookingID=@BookingID
     UPDATE Bookings SET IsCancelled = 1 WHERE BookingID=@BookingID
END
GO
```

5.5.5 Dodawanie rezerwacji na warsztat

```
CREATE PROCEDURE [dbo].[AddWorkshopBooking]
   @CustomerIDD INT,
   @ConfN VARCHAR(40),
   @Date DATETIME, --dzien konferencji
   @SDate TIME, --godzina warsztatow
   @WorkN VARCHAR(30), --nazwa warsztatow
   @Tickets SMALLINT
   AS
   BEGIN
     --musimy sprawdzic czy ktos ma rezerwacje na ten dzien na ta
konferencje itd
       DECLARE @BDT INT; --tyle ma bilecikow na ten dzien
       DECLARE @BID INT; --taki numer BookingID
       DECLARE @WTID INT; --taki numerek WorkshopTermID
```

```
       DECLARE @CDID INT; --dzien naszej konferencji

     SELECT @WTID=WorkshopTermID,@CDID=C.ConferenceDayID FROM
WorkshopTerms
     INNER JOIN ConferenceDay C ON WorkshopTerms.ConferenceDayID =
C.ConferenceDayID
                                    and DATEDIFF(day,@Date,Date)=0
     INNER JOIN Conferences C2 ON C.ConferenceID = C2.ConferenceID
                                   AND ConferenceName=@ConfN
     WHERE ABS(DATEDIFF(MINUTE,@SDate,StartTime)) < 2;

     SELECT @BDT = (NormalTickets+StudentTickets),@BID=B.BookingID
FROM BookingDays
       INNER JOIN Bookings B ON BookingDays.BookingID = B.BookingID
                                AND IsCancelled=0 and
ConferenceDayID=@CDID
       INNER JOIN Customers ON B.CustomerID = Customers.CustomerID;

     PRINT @BDT;PRINT @BID;PRINT @WTID;

     IF(@BDT is NOT NULL AND @BDT >= @Tickets
          and Not EXISTS(SELECT * FROM WorkshopBookings WHERE
@WTID=WorkshopTermID and @BID=BookingID ))  --sprawdzamy czy nie ma
juz pokrewnej rezerwacji
          --przydaloby sie sprawdzic czy jest jeszcze miejsce
         BEGIN
           INSERT INTO WorkshopBookings (
             WorkshopTermID,
             BookingID,
             Tickets) VALUES (
             @WTID,
             @BID,
             @Tickets
           )
         END
   END
GO
```

5.6 Procedury dot. warsztatów

5.6.1 Anulowanie terminu warsztatu

```
CREATE PROCEDURE [dbo].[CancelWorkshopTerm]
     @WorkshopTermID INT
   AS
   BEGIN
     UPDATE Bookings
     SET IsCancelled = 1 WHERE BookingID IN (SELECT
Bookings.BookingID FROM WorkshopTerms
```

```
      INNER JOIN WorkshopBookings ON WorkshopTerms.WorkshopTermID =
WorkshopBookings.WorkshopTermID
      INNER JOIN Bookings B ON WorkshopBookings.BookingID =
B.BookingID
      WHERE WorkshopTerms.WorkshopTermID=@WorkshopTermID)

      UPDATE WorkshopTerms
      SET IsCancelled = 1 WHERE WorkshopTermID=@WorkshopTermID
END
GO
```

5.6.2 Dodawanie terminu warsztatu

```
CREATE PROCEDURE [dbo].[AddWorkshopTerm]
 @ConfN VARCHAR(40),
 @WorkshopN VARCHAR(30),
 @Date DATETIME, --do wziecia dnia konferencji
 @StartT TIME,
 @EndT TIME,
 @Price MONEY
 AS
 BEGIN
     DECLARE @CID INT;
     SELECT @CID=ConferenceID FROM Conferences WHERE
@ConfN=ConferenceName;

     DECLARE @WID INT;
     SELECT @WID=WorkshopID FROM Workshops WHERE
WorkshopName=@WorkshopN and @CID = ConferenceID;

     DECLARE @CDID INT;
     SELECT @CDID=ConferenceDayID FROM ConferenceDay WHERE
datediff(day,@Date,Date) =0 AND @CID=ConferenceID;

     IF @CID is NOT NULL and @WID IS NOT NULL and @CDID is NOT NULL
AND DATEDIFF(MINUTE,@StartT,@EndT)>0
       BEGIN
         INSERT INTO WorkshopTerms (
           ConferenceDayID,
           WorkshopID,
           StartTime,
           EndTime,
           Price) VALUES (
           @CDID,
           @WID,
           @StartT,
           @EndT,
           @Price
         )
       END
 END
```

```
GO
```

5.6.3 Dodanie uczestnika warsztatu

```
Alter PROCEDURE [dbo].[AddWorkshopParticipant]
 @WorkshopTermID INT,
 @ParticipantID INT
 AS
   BEGIN
     SET NOCOUNT ON;
     BEGIN Try
        BEGIN TRAN ADD_WP

          --sprawdzamy czy dany uczestnik moze zostac dodany
          --musi miec rezerwacje na ten dzien
          --warsztaty nie moga byc anulowane
          if(EXISTS(Select * FROM WorkshopTerms WHERE
@WorkshopTermID=WorkshopTermID and IsCancelled=0) AND
             exists(Select * FROM Participants
              INNER JOIN Customers ON Participants.CustomerID =
Customers.CustomerID
             INNER JOIN Bookings ON Bookings.CustomerID =
Customers.CustomerID AND IsCancelled =0
              INNER JOIN WorkshopBookings WB ON Bookings.BookingID =
WB.BookingID
             WHERE @ParticipantID = ParticipantID))
           BEGIN
          --dodajemy uczestnika warsztatu
            INSERT INTO WorkshopParticipants (ParticipantID,
WorkshopTermID)
              VALUES (@ParticipantID,@WorkshopTermID)
           END
         COMMIT TRAN ADD_WP
     END TRY
     BEGIN CATCH
       ROLLBACK TRAN ADD_WP
     END CATCH
 END
GO
```

5.6.4 Dodawanie warsztatu

```
CREATE PROCEDURE [dbo].[AddWorkshop]
 @ConfN VARCHAR(40),
 @WorkshopN VARCHAR(30),
 @Duration TIME,
 @Description VARCHAR(100),
 @PartLim SMALLINT
```

```
 AS
 BEGIN
     DECLARE @CID INT;
     SELECT @CID=ConferenceID FROM Conferences WHERE
@ConfN=ConferenceName;
     IF @CID is NOT NULL

         INSERT INTO Workshops (
           ConferenceID,
           WorkshopName,
           Description,
           Duration,
           ParticipantsLimit) VALUES (
           @CID, @WorkshopN,@Description,@Duration,@PartLim
         )

 END
GO
```

6. Funkcje

6.1 Funkcje dotyczące warsztatów

6.1.1 Limit miejsc na warsztat

```
CREATE FUNCTION [dbo].[GetWorkshopLimit]
 (
   @WorkshopID INT
 )
 RETURNS SMALLINT
 AS BEGIN
 RETURN (SELECT ParticipantsLimit FROM Workshops WHERE
WorkshopID=@WorkshopID)
END
```

6.1.2 ConferenceDayID na podstawie WorkshopID

```
CREATE FUNCTION [dbo].[GetWorkshopConferenceDayID]
 (
   @WorkshopID INT
 )
 RETURNS INT
 AS BEGIN
 RETURN (SELECT ConferenceDayID FROM WorkshopTerms WHERE
WorkshopID=@WorkshopID)
END
```

6.1.3 Ilość zajętych miejsc na warsztat

```
CREATE FUNCTION [dbo].[WorkshopUsedPlaces]
 (
   @WorkshopID INT
 )
 RETURNS SMALLINT
 AS BEGIN
 RETURN (SELECT SUM(Tickets) FROM WorkshopBookings
 JOIN WorkshopTerms ON
WorkshopBookings.WorkshopTermID=WorkshopTerms.WorkshopTermID
 JOIN Workshops ON Workshops.WorkshopID=WorkshopTerms.WorkshopID
WHERE Workshops.WorkshopID=@WorkshopID)
END
```

6.1.4 Ilość wolnych miejsc na warsztat

```
CREATE FUNCTION [dbo].[WorkshopFreePlaces]
(
  @WorkshopID INT
)
RETURNS INT
AS BEGIN
DECLARE @Limit INT =
dbo.WorkshopFreePlaces(dbo.GetWorkshopConferenceDayID(@WorkshopID)
)
DECLARE @Used INT = dbo.WorkshopUsedPlaces(@WorkshopID)
RETURN @Limit-@Used
END
```

6.1.5 Koszt rezerwacji warsztatów

```
CREATE FUNCTION [dbo].[WorkshopPrice]
 (
   @WorkshopID INT
 )
 RETURNS MONEY
 AS BEGIN
 RETURN (SELECT Price FROM WorkshopTerms WHERE
WorkshopID=@WorkshopID)
END
```

6.2 Funkcje dotyczące konferencji

6.2.1 ConferenceID na podstawie ConferenceDayID

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

```
   @ConfDayID INT
 )
 RETURNS INT
 AS BEGIN
 RETURN (SELECT ConferenceID FROM ConferenceDay WHERE
ConferenceDayID=@ConfDayID)
END
```

6.2.2 Ilość miejsc na dzień konferencji

```
CREATE FUNCTION [dbo].[GetConferenceLimit]
 (
   @ConfID INT
 )
 RETURNS SMALLINT
 AS BEGIN
 RETURN (SELECT ParticipantsLimit FROM Conferences WHERE
ConferenceID = @ConfID)
END
```

6.2.3 Ilość zajętych miejsc na dzień konferencji

```
CREATE FUNCTION [dbo].[GetConferenceUsedPlaces]
 (
   @ConfDayID INT
 )
 RETURNS SMALLINT
 AS BEGIN
 RETURN (SELECT SUM(NormalTickets)+SUM(StudentTickets) FROM
BookingDays WHERE ConferenceDayID=@ConfDayID)
END
```

6.2.4 Ilość wolnych miejsc na dzień konferencji

```
CREATE FUNCTION [dbo].[ConferenceDayFreePlaces]
 (
   @ConfDayID INT
 )
RETURNS INT
 AS BEGIN
 DECLARE @Limit INT =
dbo.GetConferenceLimit(dbo.GetConferenceDayConferenceID(@ConfDayID
))
 DECLARE @Used INT = dbo.GetConferenceUsedPlaces(@ConfDayID)
 RETURN @Limit-@Used
END
```

6.2.5 ConferenceID na podstawie BookingID

```
CREATE FUNCTION [dbo].[GetBookingIDConfID]
 (
   @BookingID INT
 )
 RETURNS INT
 AS BEGIN
 RETURN (SELECT ConferenceID FROM ConferenceDay
 JOIN BookingDays ON ConferenceDay.ConferenceDayID =
BookingDays.ConferenceDayID
 WHERE BookingDays.BookingID=@BookingID)
END
```

6.2.6 Zniżka na dany dzień na daną konferencję

```
CREATE FUNCTION [dbo].[GetDiscount]
 (
   @ConfID INT,
   @date DATETIME
 )
RETURNS REAL
 AS BEGIN
 RETURN ISNULL((SELECT Discount FROM Discounts WHERE
ConferenceID=@ConfID AND StartDate<=@date AND EndDate>=@date),0)
END
```

6.3 Funkcje dotyczące opłat

6.3.1 Koszt rezerwacji

```
CREATE FUNCTION [dbo].[GetReservationCost]
 (
   @BookingID INT
 )
RETURNS MONEY
 AS BEGIN
 DECLARE @confID INT = dbo.GetBookingIDConfID(@BookingID)
 DECLARE @normalPrice MONEY = dbo.GetNormalTicketPrice(@confID)
 DECLARE @date DATETIME = (SELECT BookingDate FROM Bookings WHERE
@BookingID=BookingID)
 DECLARE @discount MONEY = dbo.GetDiscount(@confID,@date)
 DECLARE @studentDiscount REAL = (SELECT StudentsDiscount FROM
Conferences WHERE ConferenceID=@confID)
 DECLARE @bookingCost MONEY = (SELECT
SUM(NormalTickets)*@normalPrice+SUM(StudentTickets)*@normalPrice*(
1-@studentDiscount)
                                FROM BookingDays WHERE
```

```
BookingID=@BookingID)
 DECLARE @workshopCost MONEY = (SELECT SUM(value) FROM (SELECT
(SELECT SUM(WB.Tickets*WT.Price) FROM WorkshopBookings AS WB
                                JOIN WorkshopTerms AS WT ON
WB.WorkshopTermID = WT.WorkshopTermID
                               WHERE @BookingID=WB.BookingID) AS
value FROM Bookings WHERE @BookingID=BookingID) src)
 RETURN isnull(@bookingCost,0)*@discount+isnull(@workshopCost,0)
END
```

6.3.2 Cena za dzień konferencji (bez zniżek)

```
CREATE FUNCTION [dbo].[GetNormalTicketPrice]
 (
   @ConfID INT
 )
 RETURNS MONEY
 AS BEGIN
 RETURN (SELECT DayPrice FROM Conferences WHERE
@ConfID=ConferenceID)
END
```

6.1.3 Ilość miejsc zarezerwowanych przez klienta

```
CREATE FUNCTION [dbo].[GetBookedPlaces]
 (
   @CustomerID INT
 )
RETURNS SMALLINT
 AS BEGIN
 RETURN (SELECT SUM(Tickets) FROM WorkshopBookings
         JOIN Bookings B ON WorkshopBookings.BookingID =
B.BookingID
         WHERE @CustomerID=B.CustomerID)
END
```

7. Triggery

7.1 Blokowanie rezerwacji na dzień konferencji, jeżeli jest za mało miejsc

```
CREATE TRIGGER [dbo].[TooFewPlacesOnConference] ON
[dbo].[BookingDays]
AFTER INSERT
AS BEGIN
SET NOCOUNT ON;
IF EXISTS
(
  SELECT * FROM inserted AS i
  JOIN ConferenceDay AS CD ON
```

```
CD.ConferenceDayID=i.ConferenceDayID
  JOIN Conferences C ON CD.ConferenceID = C.ConferenceID
  WHERE dbo.ConferenceDayFreePlaces(C.ConferenceID) < 0
)
BEGIN
  ;THROW 50001, 'Niewystarczająca liczba wolnych miejsc, aby
zarezerwować miejsce na dzień konferencji',1
END
END
GO
```

7.2 Blokowanie rezerwacji na warsztat, jeżeli jest za mało miejsc

```
CREATE TRIGGER [dbo].[TooFewPlacesOnWorkshop] ON
[dbo].[WorkshopBookings]
AFTER INSERT
AS BEGIN
SET NOCOUNT ON;
IF EXISTS
(
 SELECT * FROM inserted AS i
 JOIN WorkshopTerms AS WT ON i.WorkshopTermID=WT.WorkshopTermID
 JOIN Workshops AS W ON WT.WorkshopID = W.WorkshopID
 WHERE dbo.WorkshopFreePlaces (W.WorkshopID) < 0
)
BEGIN
 ;THROW 50001, 'Niewystarczająca liczba wolnych miejsc, aby
zarezerwować miejsce na warsztat',1
END
END
GO
```

7.3 Blokowanie rezerwacji, jeżeli klient zarezerwował mniej miejsc na dzień konferencji niż na warsztaty

```
CREATE TRIGGER [dbo].[LessPlacesReservedOnDayThenOnWorkshop] ON
[dbo].[BookingDays]
 AFTER INSERT, UPDATE
 AS BEGIN
 SET NOCOUNT ON;
 IF EXISTS
 (
   SELECT * FROM inserted AS i
   JOIN WorkshopBookings AS WB ON WB.BookingID=i.BookingID
   WHERE WB.Tickets > i.StudentTickets+i.NormalTickets
 )
 BEGIN
   ;THROW 50001, 'Przekroczono ilość miejsc zarezerwowanych na
dany dzień konferencji',1
 END
END
```

```
GO
```

7.4 ConferenceDay.Date musi należeć do przedziału konferencji StartDate-EndDate

```
CREATE TRIGGER [dbo].[CheckConferenceDayRange] ON
[dbo].[ConferenceDay]
 AFTER INSERT, UPDATE
 AS BEGIN
 SET NOCOUNT ON;
 IF EXISTS
 (
   SELECT * FROM inserted AS i
   JOIN Conferences AS C ON i.ConferenceID=C.ConferenceID
   WHERE i.Date NOT BETWEEN C.StartDate AND C.EndDate
 )
 BEGIN
   ;THROW 50001,'Dzień konferencji nie należy do żadnego z
zakresów konferencji',1
 END
END
GO
```

7.5 ConferenceDay.Date musi należeć do przedziału konferencji StartDate-EndDate

```
CREATE TRIGGER [dbo].[CheckConferenceDayRange] ON
[dbo].[ConferenceDay]
 AFTER INSERT, UPDATE
 AS BEGIN
 SET NOCOUNT ON;
 IF EXISTS
 (
   SELECT * FROM inserted AS i
   JOIN Conferences AS C ON i.ConferenceID=C.ConferenceID
   WHERE i.Date NOT BETWEEN C.StartDate AND C.EndDate
 )
 BEGIN
   ;THROW 50001,'Dzień konferencji nie należy do żadnego z
zakresów konferencji',1
 END
END
GO
```

7.5 ConferenceDay.Date musi należeć do przedziału konferencji StartDate-EndDate

```
CREATE TRIGGER [dbo].[CheckConferenceDayRange] ON
[dbo].[ConferenceDay]
 AFTER INSERT, UPDATE
 AS BEGIN
 SET NOCOUNT ON;
```

```
 IF EXISTS
 (
   SELECT * FROM inserted AS i
   JOIN Conferences AS C ON i.ConferenceID=C.ConferenceID
   WHERE i.Date NOT BETWEEN C.StartDate AND C.EndDate
 )
 BEGIN
   ;THROW 50001,'Dzień konferencji nie należy do żadnego z
zakresów konferencji',1
 END
END
GO
```

7.5 Zniżka za wcześniejszą rejestrację może obowiązywać najwyżej do rozpoczęcia konferencji

```
CREATE TRIGGER [dbo].[CheckDiscountConference] ON [dbo].[Discounts]
AFTER INSERT, UPDATE
 AS BEGIN
 SET NOCOUNT ON;
 IF EXISTS
 (
   SELECT * FROM inserted AS i
   JOIN Conferences AS C ON C.ConferenceID=i.ConferenceID
   WHERE C.StartDate < i.StartDate
 )
 BEGIN
   ;THROW 50001,'Zniżka może obowiązywać do rozpoczęcia
konferencji',1
 END
END
```

8. Generator - RedGate

Baza została wygenerowana w programie RedGate Data Generator 4. Poza jedną tabelą, która wymagała stworzenia ciągłych w czasie okresów konferencji - ConferenceDay.

8.1 Generator ConferenceDay

```
SELECT * INTO #TableD
FROM
 (
   SELECT ConferenceID AS ID,
   DATEDIFF(DAY,dbo.Conferences.StartDate,dbo.Conferences.EndDate)
AS quantity,
   dbo.Conferences.StartDate AS refColumn
   FROM Conferences
 ) AS T
;WITH Counter(Number) AS
(
 SELECT 0 UNION ALL
```

```
 SELECT 1+Number FROM Counter WHERE Number<7
)
INSERT INTO [dbo].[ConferenceDay]
SELECT
 ID AS ConferenceID,
 DATEADD( DAY, N.Number, A.refColumn) AS Date
FROM
 #TableD A
 JOIN
 Counter N ON N.Number <= A.quantity
ORDER BY A.ID, Date
```

9. Uprawnienia

9.1 Role w systemie

- Klient - posiada dostęp tylko do wybranych procedur
  - Dodawanie klienta
  - Dodanie klienta - firmy
  - Dodanie klienta - osoby indywidualnej
- Administrator - posiada dostęp do wszystkich widoków, tabel i procedur
- Pracownik - posiada dostęp do widoków i procedur

10. Przykłady wywołań

```
Use jzur_a

--dodwanie konferencji wraz z dniami
	EXEC [dbo].AddConference
	    @CoferenceName = 'Spectro2',
	    @StartDate = '2018/05/01',
	    @EndDate = '2018/05/02',
	    @Limit = 200,
	    @SDiscount = 0.2 ,
	    @Price = 100;

--dodawanie panstwa
	DECLARE @CID INT;

	EXEC [dbo].FindAddCountryID
	      @CountryName = 'Belgia', @CountryID = @CID

--dodawanie miasta
	 DECLARE @CID INT;

	 EXEC [dbo].AddCity
```

```
            @City = 'Warszawa', @Country = 'Norwegia', @CityID
= @CID OUTPUT;

        GO
        EXEC [dbo].AddIndyvidualCustomer @name = 'marcin',
@surname = 'Fudbcvaligfdgfdsfński', @Phone = '782644567',
                                        @Email =
'fudfsadff@gmfdssdsadsafbcvdfsail.com' , @Street =
'markodaswski432ego 10', @Postalcode = '34-881',
                                        @City = 'Keardasak',
@Country = 'jdasdaeqsda'


--dodawanie klienta firmowego analogicznie jak wyżej
        GO
        EXEC [dbo].AddCompanyCustomer @City=
'puszyczka320',@Country = 'puszkolandkotoland',@Email =
'puszkakot@puszka.com',
                                    @Street = 'dlugokrotka
22',@PostalCode = '45-881' ,
                                    @CompanyName =
'puszkikotow Sp z o o ' ,@Nip = '0123453789' ,@Phone =
'123456749'


--dodawanie adresu konferencji
          --dodawanie z adresem ktory jest juz w bazie
          EXECUTE [dbo].AddConferenceAddress @ConferenceName =
'Spectro' ,
                                            @City =
'Warszawa',
                                            @Adress = 'dluga
15',
                                            @Country =
'Norwegia'

          --dodawanie z adresem ktorego jeszcze nie ma, dziala
          EXECUTE [dbo].AddConferenceAddress @ConferenceName =
'Spectro' ,
                                            @City =
'listopad',
                                            @Adress = 'dluga
15',
                                            @Country =
'IRLANDIA'

--anulowanie konferencji trzeba srawdzic jeszcze raz
        EXECUTE [dbo].CancelConference @ConferenceName =
'Spectro'
```

```
--dodawanie znizek do konferencji jezeli nie ma to 100%

        --podajemy jeszcze raz dobrze niepowinno sie dodac i
sie nie dodaje ok
        EXECUTE [dbo].AddConferenceDiscounts @ConfernceN =
'Spectro',@StartDate = '2017/05/01'  ,@EndDate =
'2018/02/01',@Discounts = '0.32'

        --dajmy przedzial ktory powinien sie dodac
        EXECUTE [dbo].AddConferenceDiscounts @ConfernceN =
'Spectro',@StartDate = '2016/05/02'  ,@EndDate =
'2020/02/01',@Discounts = '0.32'


--dodawanie warsztatu
        Go
   EXECUTE AddWorkshop @ConfN = 'Spectro2', @WorkshopN =
'lab23',@Duration = '1:30',@Description = 'jakies laby',
@PartLim = '30';
   Go

--ddawanie terminu warsztatu dziala ale malo wygodnie sie go
dodaje poza tym jest jedna konferencja na ktorej sie da
sprawdzic dzialanie

EXECUTE AddWorkshopTerm @ConfN = 'Spectro2',@WorkshopN =
'lab23' ,@Date = '2018-05-01' ,@StartT = '0:0:0' ,@EndT =
'01:30:00',@Price ='100';



--dodawanie rezerwacji odbywa sie poprzez dodanie rezerwacji na
dany dzien
EXECUTE AddBookingDay @CustomerIDD = '31',@ConfName =
'Spectro',@Date = '2018-05-01',@NTick = '10' ,@STick = '12';

EXECUTE AddBookingDay @CustomerIDD = '33',@ConfName =
'Spectro2',@Date = '2018-05-01',@NTick = '5' ,@STick = '7';



EXECUTE [dbo].AddWorkshopBooking @CustomerIDD ='33' ,@ConfN =
'Spectro2',@Date = '2018-05-01',@SDate ='00:00:00',@WorkN
='lab2' ,@Tickets ='7';
```

```
--anulowanie konferencji powoduje anuloanie warsztatow oraz
wszelkich rezerwacji z nia powiazanych

 EXECUTE CancelConference @ConferenceName = 'Spectro2';

 UPDATE Conferences
 SET IsCancelled = 0 WHERE ConferenceName ='Lambda II New ' ;


--dodawanie pracownika na odpowiednia rezerwacje
 --zwykly
 EXECUTE AddEmployee @Name = 'jan' ,@Surname =
'kowalski',@phone = '154754876',@companyID='33' ,@CardNumber =
NULL ,@ExpireDate = NULL ;

 --student
 EXECUTE AddEmployee @Name = 'łoś',@Surname = 'dziobak',@phone
='123987645' ,@companyID='16' ,@CardNumber = '12345'
,@ExpireDate ='2018-05-01'  ;


--DODAWANIE REZERWACJI DLA INDYWIDUALNEGO
EXECUTE AddIndyvidualSingleBooking @phone = '782644567' ,@Card=
'12212',@Expdate = '2018-05-01',@ConferenceN = 'Spectro',
                                    @ConferenceD ='2018-05-01'
,@WorkshopN = 'lab2',@WStart = '00:00:00';
```