# System Zarządzania Konferencjami

Dawid Majchrowski
Piotr Dulęba

22 styczeń 2019

## Spis treści

# 1 Aktorzy

- Administrator bazy danych
- Klient
- Uczestnik konferencji
- Organizator konferencji (Pracownik firmy organizującej konferencję)

# 2 Opis funkcji z punktu widzenia użytkowników

## 2.1 Klient

- Zarejestruj się do systemu
- Edytuj swoje dane (wymagane konto w systemie)
- Przeglądaj dostępne konferencje (wymagane konto w systemie)
- Przeglądaj uczestników konferencji (wymagane konto w systemie)
- uczestników na dany dzień konferencji (wymagane konto w systemie)
- Usuń uczestników z danego dnia konferencji (wymagana rejestracja i rezerwacja miejsca na konferencje)
- Dodaj rezerwację na dany warsztat (wymagana rejestracja na dany dzień konferencji)
- Anuluj rezerwację na dany warsztat (wymagana rejestracja na dany dzień konferencji)
- Ustaw metodę płatności (wymagane konto w systemie i rezerwacja miejsca na konferencje)
- Opłać konferencje (wymagane konto w systemie i rezerwacja miejsca na konferencje i ustawiona metoda płatności)

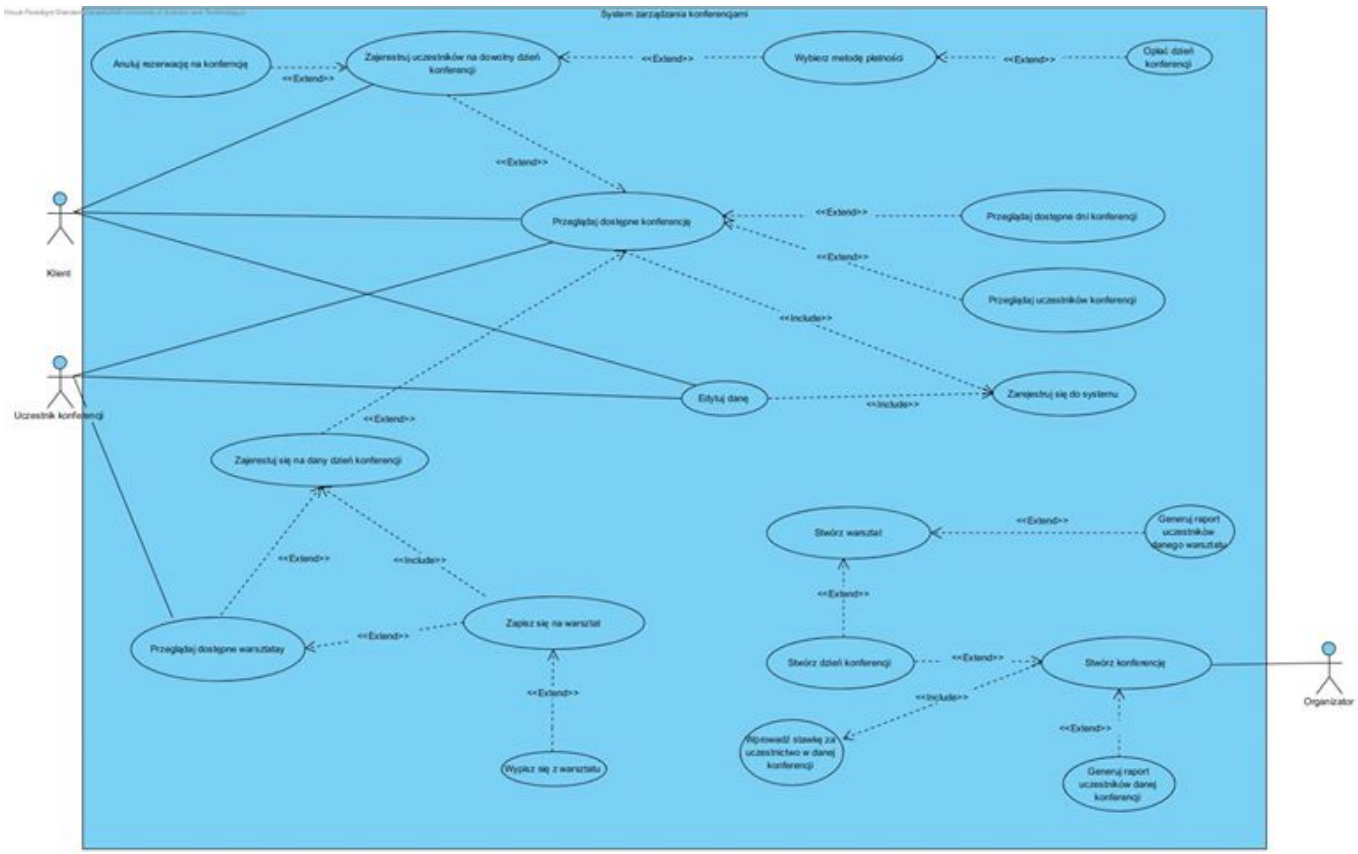
## 2.2 Uczestnik konferencji

- Zarejestruj się do systemu
- Edytuj swoje dane (wymagane konto w systemie)
- Zarejestruj się na dany dzień konferencji (wymagane konto w systemie / rejestracja przez klienta dnia na konferencje)
- Przeglądaj dostępne warsztaty (wymagane konto w systemie)
- kilkudniowych)
- Dodaj rezerwację na dany warsztat (wymagana rejestracja na dany dzień konferencji)
- Anuluj rezerwację na dany warsztat (wymagana rejestracja na dany dzień konferencji)
- Zapisz się na warsztat (wymagane konto w systemie / rejestracja dnia na konferencje / rejestracja warsztat)

## 2.3 Organizator konferencji (pracownicy firmy organizującej konferencję)

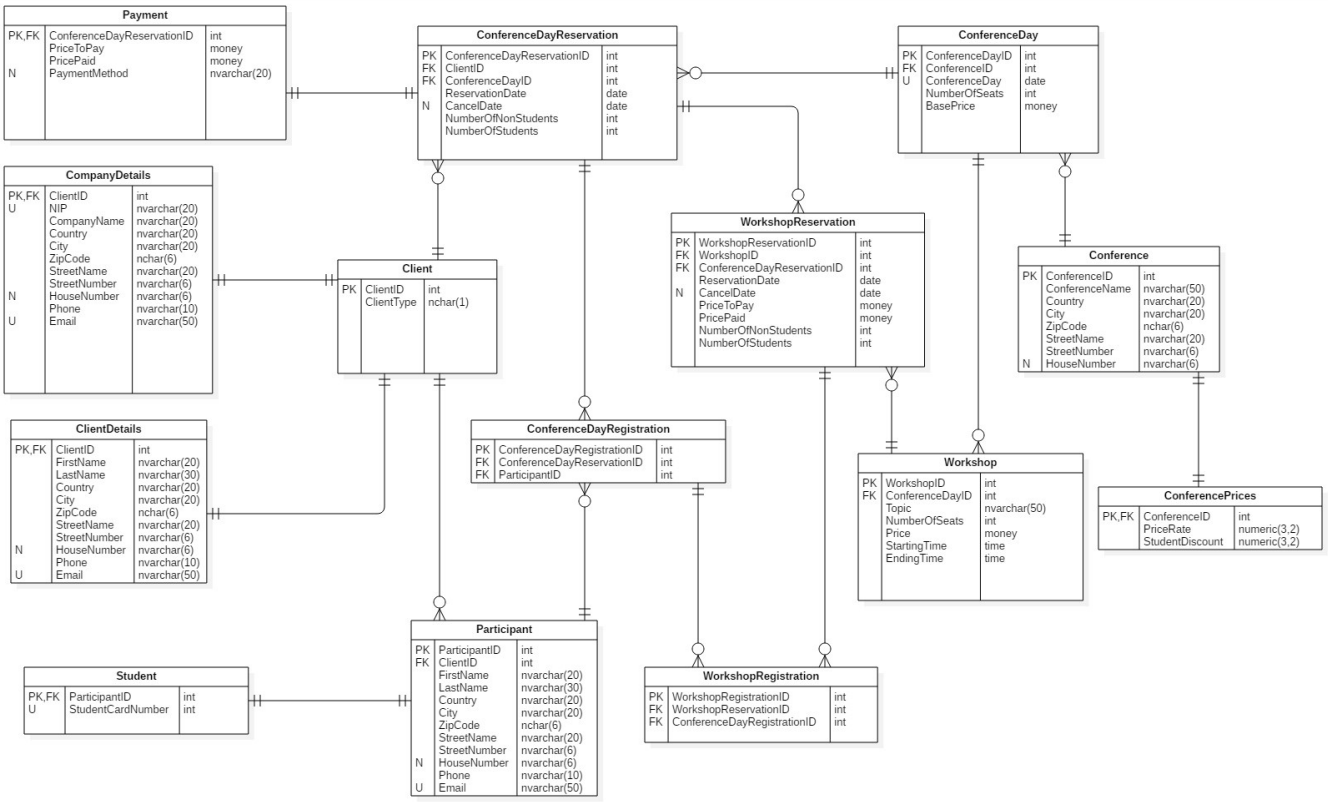
- Stwórz konferencje
- Stwórz dzień konferencji (wymagana konferencja)
- Stwórz warsztat (wymagany dzień konferencji)
- Generuj raport uczestników danej konferencji
- Generuj raport uczestników danego warsztatu
- Wprowadź informacje o płatności przez klienta
- Wprowadź stawkę za uczestnictwo w danej konferencji
- Wprowadź stawkę za uczestnictwo w danym warsztacie

# 3 Diagram Use Case




Rysunek 1: Diagram UC

# 4 Schemat bazy danych




Rysunek 2: Schemat bazy danych

# 5 Spis Tabel

## 5.1 Client

**Opis:** Tabela przechowująca ID klientów oraz informację, czy jest on klientem indywidualnym czy firmą.

- **ClientID** - ID klienta
- **ClientType** - Informacja czy jest to klient indywidualny, czy firma (Company)

**Warunki integralnościowe i ograniczneia:**

- **ClientType** - znak I albo C

**Kod tworzący tabelę:**

```
CREATE TABLE dbo.Client(
ClientID INT PRIMARY KEY IDENTITY(1,1),
ClientType  NCHAR(1) NOT NULL
);
```

**Warunki integralności:**

```
ALTER TABLE dbo.Client
 ADD
   CONSTRAINT ClientTypeIorC
     CHECK (ClientType IN ('C','I'))
```

## 5.2 CompanyDetails

**Opis:** Tabela przechowująca dane klienta który jest firmą.

- **ClientID** - ID klienta
- **NIP** - NIP firmy
- **CompanyName** - nazwa firmy
- **Country** - kraj
- **City** - miasto
- **ZipCode** - kod pocztowy
- **StreetName** - nazwa ulicy
- **StreetNumber** - numer budynku
- **HouseNumber** - numer lokalu
- **Phone** - telefon kontaktowy
- **Email** - adres kontaktowy email

**Warunki integralnościowe i ograniczneia:**

- **ClientID** - klucz obcy do tabeli **Client**
- **NIP** - unikatowy dla każdej firmy

- **Email** - unikatowy dla każdej firmy, musi zawierać znaki **@.**

**Kod tworzący tabelę:**

```
CREATE TABLE dbo.CompanyDetails(
ClientID INT PRIMARY KEY,
NIP NVARCHAR(20) NOT NULL,
CompanyName NVARCHAR(20) NOT NULL,
Country NVARCHAR(20) NOT NULL,
City NVARCHAR(20) NOT NULL,
ZipCode NCHAR(6) NOT NULL,
StreetName NVARCHAR(20) NOT NULL,
StreetNumber NVARCHAR(6) NOT NULL,
HouseNumber NVARCHAR(6) NULL,
Phone NVARCHAR(10) NOT NULL,
Email NVARCHAR(50) NOT NULL
);
```

**Warunki integralności:**

```
ALTER TABLE dbo.CompanyDetails
 ADD
    CONSTRAINT CompanyDetailsFK
      FOREIGN KEY(ClientID)
      REFERENCES Client(ClientID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    CONSTRAINT UniqueNIP
      UNIQUE(NIP),
     CONSTRAINT UniqueCompanyEmail
      UNIQUE(Email),
     CONSTRAINT CompanyEmailLike
      CHECK(Email like '%@%.%')
```

## 5.3 ClientDetails

**Opis:** Tabela przechowująca dane klienta który jest osobą fizyczną.

- **ClientID** - ID klienta
- **FirstName** - imię klienta
- **LastName** - nazwisko klienta
- **Country** - kraj
- **City** - miasto
- **ZipCode** - kod pocztowy
- **StreetName** - nazwa ulicy
- **StreetNumber** - numer budynku
- **HouseNumber** - numer lokalu
- **Phone** - telefon kontaktowy

- **Email** - adres kontaktowy email

**Warunki integralnościowe i ograniczneia:**

- **ClientID** - klucz obcy do tabeli **Client**, modyfikacja i usuwanie kaskadowe
- **Email** - unikatowy dla każdej osoby fizycznej, musi zawierać znaki **@.**

**Kod tworzący tabelę:**

```
CREATE TABLE dbo.ClientDetails(
ClientID INT PRIMARY KEY,
FirstName NVARCHAR(20) NOT NULL,
LastName NVARCHAR(30) NOT NULL,
Country NVARCHAR(20) NOT NULL,
City NVARCHAR(20) NOT NULL,
ZipCode NCHAR(6) NOT NULL,
StreetName NVARCHAR(20) NOT NULL,
StreetNumber NVARCHAR(6) NOT NULL,
HouseNumber NVARCHAR(6) NULL,
Phone NVARCHAR(10) NOT NULL,
Email NVARCHAR(50) NOT NULL
);
```

**Warunki integralności:**

```
ALTER TABLE dbo.ClientDetails
 ADD
    CONSTRAINT ClientDetailsFK
      FOREIGN KEY(ClientID)
      REFERENCES Client(ClientID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    CONSTRAINT UniqueClientEmail
      UNIQUE(Email),
    CONSTRAINT ClientEmailLike
      CHECK(Email like '%@.%%')
```

## 5.4 Participant

**Opis:** Tabela przechowująca dane uczestnika konferencji.

- **ParticipantID** - ID uczestnika
- **ClientID** - ID klienta pod którego "podlega"uczestnik
- **FirstName** - imię uczestnika
- **LastName** - nazwisko uczestnika
- **Country** - kraj
- **City** - miasto
- **ZipCode** - kod pocztowy
- **StreetName** - nazwa ulicy

- **StreetNumber** - numer budynku
- **HouseNumber** - numer lokalu
- **Phone** - telefon kontaktowy
- **Email** - adres kontaktowy email

**Warunki integralnościowe i ograniczneia:**

- **ClientID** - klucz obcy do tabeli **Client**, modyfikacja i usuwanie kaskadowe
- **Email** - unikatowy dla każdego uczestnika, musi zawierać znaki **@.**

**Kod tworzący tabelę:**

```
CREATE TABLE dbo.Participant(
ParticipantID INT PRIMARY KEY IDENTITY(1,1),
ClientID INT NOT NULL,
FirstName NVARCHAR(20) NOT NULL,
LastName NVARCHAR(30) NOT NULL,
Country NVARCHAR(20) NOT NULL,
City NVARCHAR(20) NOT NULL,
ZipCode NCHAR(6) NOT NULL,
StreetName NVARCHAR(20) NOT NULL,
StreetNumber NVARCHAR(6) NOT NULL,
HouseNumber NVARCHAR(6) NULL,
Phone NVARCHAR(10) NOT NULL,
Email NVARCHAR(50) NOT NULL
);
```

**Warunki integralności:**

```
ALTER TABLE dbo.Participant
 ADD
    CONSTRAINT ParticipantFK -- Foreign key to Participant

      FOREIGN KEY(ClientID)
      REFERENCES Client(ClientID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    CONSTRAINT UniqueParticipantEmail
      UNIQUE(Email),
    CONSTRAINT ParticipantEmailLike
      CHECK(Email like '%@%.%')
```

## 5.5 Student

**Opis:** Tabela przechowująca dodatkowe dane uczestnika konferencji, który jest studentem.

- **ParticipantID** - ID uczestnika
- **StudentCardNumber** - numer albumu studenta

**Warunki integralnościowe i ograniczneia:**

- **ParticipantID** - klucz obcy do tabeli **Participant**, modyfikacja i usuwanie kaskadowe

- **StudentCardNumber** - unikatowy dla każdego studenta biorącego udział w konferencjach

**Kod tworzący tabelę:**

```
CREATE TABLE dbo.Student(
ParticipantID INT PRIMARY KEY,
StudentCardNumber INT NOT NULL
);
```

**Warunki integralności:**

```
ALTER TABLE dbo.Student
 ADD
    CONSTRAINT StudentFK— Foreign key to Student
      FOREIGN KEY(ParticipantID)
      REFERENCES Participant(ParticipantID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    CONSTRAINT UniqueStudentCard
      UNIQUE(StudentCardNumber)
```

## 5.6 Conference

**Opis:** Tabela przechowująca dodatkowe o konferencjach rezerwowanych przez klientów.

- **ConferenceID** - ID konferencji
- **ConferenceName** - nazwa konferencji
- **Country** - kraj
- **City** - miasto
- **ZipCode** - kod pocztowy
- **StreetName** - nazwa ulicy
- **StreetNumber** - numer budynku
- **StreetName** - numer mieszkania

**Kod tworzący tabelę:**

```
  CREATE TABLE dbo.Conference(
   ConferenceID INT PRIMARY KEY IDENTITY(1,1),
   ConferenceName NVARCHAR(50) NOT NULL,
   Country NVARCHAR(20) NOT NULL,
   City NVARCHAR(20) NOT NULL,
   ZipCode NCHAR(6) NOT NULL,
   StreetName NVARCHAR(20) NOT NULL,
   StreetNumber NVARCHAR(6) NOT NULL,
   HouseNumber NVARCHAR(6) NULL
   );
GO
```

## 5.7 Conference Prices

**Opis:** Tabela przechowująca dodatkowe dane o cenach konferencji, oraz o wysokości zniżki studenckiej.

- **ConferenceID** - ID konferencji
- **PriceRate** - wysokość ceny konferencji
- **StudentDiscount** - wysokość zniżki studenckiej

**Warunki integralnościowe i ograniczneia:**

- **ConferenceID** - klucz obcy do tabeli **Conference**, modyfikacja i usuwanie kaskadowe
- **PriceRate** - wartość od 0 do 1
- **StudentDiscount** - wartość od 0 do 1

**Kod tworzący tabelę:**

```
CREATE TABLE dbo.ConferencePrices(
ConferenceID INT PRIMARY KEY,
PriceRate NUMERIC(3,2) NOT NULL,
StudentDiscount NUMERIC(3,2) NOT NULL
);
```

**Warunki integralności:**

```
ALTER TABLE dbo.ConferencePrices
 ADD
   CONSTRAINT ConferencePricesFK -- Foreign key to ConferencePrices
     FOREIGN KEY(ConferenceID)
     REFERENCES Conference(ConferenceID)
     ON DELETE CASCADE
     ON UPDATE CASCADE,
   CONSTRAINT PriceValueCheck
     CHECK (PriceRate BETWEEN 0 AND 1),
   CONSTRAINT StudentDiscountCheck
     CHECK (StudentDiscount BETWEEN 0 AND 1)
```

## 5.8 ConferenceDay

**Opis:** Tabela przechowująca informacje na temat danego dnia konferencji, bo konferencje mogą trwać kilka dni

- **ConferenceDayID** - ID dnia konferencji
- **ConferenceID** - ID konferencji
- **ConferenceDay** - data danego dnia konferencji
- **NumberOfSeats** - liczba dostępnych miejsc
- **BasePrice** - cena za uczestnictwo w danym dniu konferencji

**Warunki integralnościowe i ograniczneia:**

- **ConferenceID** - klucz obcy do tabeli **Conference**, modyfikacja i usuwanie kaskadowe

- **ConferenceDay** -Unikatowy dzień, nie może być z przeszłości max 2 lata w przód
- **NumberOfSeats** - nie mniejsze niż 1
- **BasePrice** - nie mniejsza niż 0

**Kod tworzący tabelę:**

```
CREATE TABLE dbo.ConferenceDay(
ConferenceDayID INT PRIMARY KEY IDENTITY(1,1),
ConferenceID INT NOT NULL,
ConferenceDay DATE NOT NULL,
NumberOfSeats INT NOT NULL,
BasePrice MONEY NOT NULL
);
```

**Warunki integralności:**

```
ALTER TABLE dbo.ConferenceDay
  ADD
    CONSTRAINT ConferenceDayFK
      FOREIGN KEY(ConferenceID)
      REFERENCES Conference(ConferenceID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    CONSTRAINT UniqueConferenceDay
      UNIQUE(ConferenceDay),
    CONSTRAINT NumberOfSeatsDay
      CHECK (NumberOfSeats > 0),
    CONSTRAINT ConferenceDayPrice
      CHECK (BasePrice >= 0),
    CONSTRAINT ConferenceDayDate
      CHECK (GETDATE() <= ConferenceDay
      AND DATEDIFF(DAY,GETDATE(),ConferenceDay) <= 730)
GO
```

## 5.9 Workshop

**Opis:** Tabela przechowująca informacje na temat warsztatu odbywającego się w ramach konferencji.

- **WorkshopID** - ID warsztatu
- **ConferenceDayID** - ID dnia konferencji w którym jest warsztat
- **Topic** - opis warsztatu
- **NumberOfSeats** - liczba dostępnych miejsc
- **Price** - cena za uczestnictwo w danym warsztacie
- **StartingTime** - czas rozpoczęcia warsztatów
- **EndingTime** - czas zakończenia warsztatów

**Warunki integralnościowe i ograniczneia:**

- **ConferenceDayID** - klucz obcy do tabeli **ConferenceDay**, modyfikacja i usuwanie kaskadowe

- **StartingTime, EndingTime** - StartingTime < EndingTime
- **NumberOfSeats** - nie mniejsze niż 0
- **Price** - nieujemna

**Kod tworzący tabelę:**

```
CREATE TABLE dbo.Workshop(
WorkshopID INT PRIMARY KEY IDENTITY(1,1),
ConferenceDayID INT NOT NULL,
Topic NVARCHAR(50) NOT NULL,
NumberOfSeats INT NOT NULL,
Price MONEY NOT NULL,
StartingTime TIME NOT NULL,
EndingTime TIME NOT NULL
);
```

**Warunki integralności:**

```
ALTER TABLE dbo.Workshop
 ADD
    CONSTRAINT WorkshopFK -- Foreign key to Workshop
      FOREIGN KEY(ConferenceDayID)
      REFERENCES ConferenceDay(ConferenceDayID)
      ON DELETE CASCADE
      ON UPDATE CASCADE ,
    CONSTRAINT CorrectTimeCheck
      CHECK (StartingTime < EndingTime),
    CONSTRAINT NumberOfSeatsWorkshop
      CHECK (NumberOfSeats > 0),
    CONSTRAINT MoneyWorkshopCheck
        CHECK (Price >= 0)
```

## 5.10 ConferenceDayReservation

**Opis:** Tabela przechowująca informacje o rezerwacjach które dokonują klienci na dane dni konferencji.

- **ConferenceDayReservationID** - ID rezerwacji na dzień konferencji dokonanej przez klienta
- **ClientID** - ID klienta który dokonał rezerwacji
- **ConferenceDayID** - ID dnia na który dokonywana jest rezerwacja
- **ReservationDate** - data dokonania rezerwacji
- **CancelDate** - data odwołania rezerwacji
- **NumberOfParticipants** - liczba zapisanych uczestników
- **NumberOfStudents** - liczba zapisanych studentów

**Warunki integralnościowe i ograniczneia:**

- **ClientID** - klucz obcy do tabeli **Client**
- **ConferenceDayID** - klucz obcy do tabeli **ConferenceDay**

- **NumberOfParticipants** - liczba większa od 0
- **NumberOfStudents** - liczba większa od 0, ale łącznie z NumberOfParticipants musi być dodatnie
- **CancelDate** - jeżeli nie jest nullem, to CancelDate musi być po ReservationDate

**Kod tworzący tabelę:**

```
IF object_id('dbo.ConferenceDayReservation','U') IS NOT NULL
  DROP TABLE dbo.ConferenceDayReservation;
CREATE TABLE dbo.ConferenceDayReservation(
ConferenceDayReservationID INT PRIMARY KEY IDENTITY(1,1),
ClientID INT NOT NULL,
ConferenceDayID INT NOT NULL,
ReservationDate DATE NOT NULL,
CancelDate DATE NULL,
NumberOfNonStudents INT NOT NULL,
NumberOfStudents INT NOT NULL
);
GO
```

**Warunki integralności:**

```
ALTER TABLE dbo.ConferenceDayReservation
  ADD
    CONSTRAINT ConferenceDayReservationFK1
      FOREIGN KEY(ClientID)
      REFERENCES Client(ClientID),
    CONSTRAINT ConferenceDayReservationFK2
      FOREIGN KEY(ConferenceDayID)
      REFERENCES ConferenceDay(ConferenceDayID),
    CONSTRAINT NumberOfParticipantDay
      CHECK (NumberOfNonStudents >= 0
      AND NumberOfStudents >= 0
      AND (NumberOfNonStudents + NumberOfStudents) > 0),
    CONSTRAINT CancelDateCheck
      CHECK(CancelDate IS NULL OR CancelDate > ReservationDate)
GO
```

## 5.11 ConferenceDayRegistration

**Opis:** Tabela przechowująca informacje o zapisach uczestników na dni konferencji

- **ConferenceDayResegistrationID** - ID zapisu uczestnika na dany dzień konferencji
- **ConferenceDayReservationID** - ID rezerwacji której dokonał klient na dany dzień
- **ParticipantID** - ID uczestnika

**Warunki integralnościowe i ograniczneia:**

- **ConferenceDayReservationID** - klucz obcy do tabeli **ConferenceDayReservation**
- **ParticipantID** - klucz obcy do tabeli **Participant**
- **ParticipantID,ConferenceDayReservationID** - para ta musi być unikalna

**Kod tworzący tabelę:**

```
IF object_id('dbo.ConferenceDayRegistration','U') IS NOT NULL
  DROP TABLE dbo.ConferenceDayRegistration;
CREATE TABLE dbo.ConferenceDayRegistration(
ConferenceDayRegistrationID INT PRIMARY KEY IDENTITY(1,1),
ConferenceDayReservationID INT NOT NULL,
ParticipantID INT NOT NULL
);
GO
```

**Warunki integralności:**

```
ALTER TABLE dbo.ConferenceDayRegistration
  ADD
    CONSTRAINT ConferenceDayRegistrationFK1
      FOREIGN KEY(ConferenceDayReservationID)
      REFERENCES
        ConferenceDayReservation(ConferenceDayReservationID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    CONSTRAINT ConferenceDayRegistrationFK2
      FOREIGN KEY(ParticipantID)
      REFERENCES Participant(ParticipantID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    CONSTRAINT OneParticipantPerDay
      UNIQUE(ParticipantID,ConferenceDayReservationID)
GO
```

## 5.12 Payment

**Opis:** Tabela przechowująca informacje o dokonanych opłatach za rezerwację dni konferencji przez klientów.

- **ConferenceDayReservsationIDID** - ID dnia konferencji
- **PriceToPay** - wysokosć opłaty którą należy dokonać
- **PricePaid** - wysokosć opłaty już uiszczonej
- **PaymentMethod** - sposób dokonania zapłaty

**Warunki integralnościowe i ograniczneia:**

- **ConferenceDayReservationID** - klucz obcy do tabeli **ConferenceDayReservation**
- **PricePaid, PriceToPay** -obydwa nieujemne, PricePaid <=PriceToPay

**Kod tworzący tabelę:**

```
CREATE TABLE dbo.Payment(
ConferenceDayReservationID INT PRIMARY KEY,
PriceToPay MONEY NOT NULL,
PricePaid MONEY NOT NULL DEFAULT 0,
PaymentMethod NVARCHAR(20) NULL
);
```

**Warunki integralności:**

```
ALTER TABLE dbo.Payment
 ADD
    CONSTRAINT PaymentFK -- Foreign key to Payment
      FOREIGN KEY(ConferenceDayReservationID)
      REFERENCES ConferenceDayReservation(ConferenceDayReservationID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    CONSTRAINT PaymentMoneyCheck
      CHECK (PricePaid >= 0 AND PriceToPay >= 0 AND PricePaid <= PriceToPay)
```

## 5.13 WorkShopReservation

**Opis:** Tabela przechowująca informacje o dokonanych rezerwacjach na warsztaty przez klientów

- **WorkshopReservationID** - ID rezerwacji na warsztat
- **WorkshopID** - ID warsztatu
- **ConferenceDayReservationID** - ID dnia konferencji w którym jest warsztat
- **ReservationDate** - data dokonania rezerwacji
- **CancelDate** - data odwołania rezerwacji
- **PriceToPay** - kwota do zapłaty za uczestnictwo
- **PricePaid** - kwota jaka została już zapłacona za uczestnictwo
- **NumberOfParticipants** - liczba zapisanych uczestników
- **NumberOfStudents** - liczba zapisanych studentów

**Warunki integralnościowe i ograniczneia:**

- **WorkshopID** - klucz obcy do tabeli **Workshop**
- **ConferenceDaReservationID** - klucz obcy do tabeli **ConferenceDayReservation**
- **PricePaid, PriceToPay** -obydwa nieujemne, PricePaid <=PriceToPay
- **NumberOfParticipants, NumberOfStudents** - oba nieujemne, ich suma > 0

**Kod tworzący tabelę:**

```
  CREATE TABLE dbo.WorkshopReservation(
  WorkshopReservationID INT PRIMARY KEY IDENTITY(1,1),
  WorkshopID INT NOT NULL,
  ConferenceDayReservationID INT NOT NULL,
  ReservationDate DATE NOT NULL,
  CancelDate DATE NULL,
  PriceToPay MONEY NOT NULL,
  PricePaid MONEY NOT NULL DEFAULT 0,
  NumberOfNonStudents INT NOT NULL,
  NumberOfStudents INT NOT NULL,
  );
GO
```

**Warunki integralności:**

```
ALTER TABLE dbo.WorkshopReservation
  ADD
    CONSTRAINT WorkshopReservationFK1
      FOREIGN KEY(WorkshopID)
      REFERENCES Workshop(WorkshopID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    CONSTRAINT WorkshopReservationFK2
      FOREIGN KEY(ConferenceDayReservationID)
      REFERENCES ConferenceDayReservation(ConferenceDayReservationID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    CONSTRAINT MoneyWorkshopReservationCheck
      CHECK (PricePaid >= 0 AND PriceToPay >= 0
      AND PricePaid <= PriceToPay),
    CONSTRAINT NumberOfParticipantWorkshop
      CHECK (NumberOfNonStudents >= 0
      AND NumberOfStudents >= 0
       AND (NumberOfNonStudents + NumberOfStudents) > 0)
GO
```

## 5.14 WorkShopRegistration

**Opis:** Tabela przechowująca informacje o zapisach dokonanych na warsztaty przez uczestników

- **WorkshopRegistrationID** - ID zapisu na warsztat
- **WorkshopReservationID** - ID rezerwacji na warsztat

**Warunki integralnościowe i ograniczneia:**

- **WorkshopReservationID** - klucz obcy do tabeli **WorkshopReservation**
- **ParticipantID** - klucz obcy do tabeli **Participant**
- **ParticipantID, WorkshopReservationID** - para ta musi być unikalna

**Kod tworzący tabelę:**

```
  CREATE TABLE dbo.WorkshopRegistration(
  WorkshopRegistrationID INT PRIMARY KEY IDENTITY(1,1),
  WorkshopReservationID INT NOT NULL,
  ConferenceDayRegistrationID INT NOT NULL
  );
GO
```

**Warunki integralności:**

```
ALTER TABLE dbo.WorkshopRegistration
 ADD
   CONSTRAINT WorkshopRegistrationFK1
     FOREIGN KEY(WorkshopReservationID)
     REFERENCES WorkshopReservation(WorkshopReservationID)
     ON DELETE CASCADE
```

```
    ON UPDATE CASCADE,
CONSTRAINT WorkshopRegistrationFK2
    FOREIGN KEY(ParticipantID)
    REFERENCES Participant(ParticipantID)
    ON DELETE CASCADE
    ON UPDATE CASCADE
CONSTRAINT OneParticipantPerWorkshop
    UNIQUE(ParticipantID, WorkshopReservationID)
```

# 6 Spis triggerów

## 6.1 ConferenceDayReservationTrigger

**Opis:** Trigger ustawienie daty usuwania rezerwacji, która następuje 14 dni po dacie dokonania rezerwacji, uruchamiany po wstawianiu rekodu do tabeli ConferenceDayReservation.

**Kod:**

```
CREATE TRIGGER ConferenceDayReservationTrigger
ON ConferenceDayReservation
AFTER INSERT
AS
UPDATE ConferenceDayReservation
SET ConferenceDayReservation.CancelDate =
DATEADD(DAY,14,ConferenceDayReservation.ReservationDate)
FROM inserted AS i
WHERE ConferenceDayReservation.ConferenceDayReservationID =
i.ConferenceDayReservationID;
```

## 6.2 WorkshopReservationTrigger

**Opis:** Trigger ustawienie daty usuwania rezerwacji, która następuje 14 dni po dacie dokonania rezerwacji na warsztat.

**Kod:**

```
CREATE TRIGGER WorkshopReservationTrigger
ON WorkshopReservation
AFTER INSERT
AS
UPDATE WorkshopReservation
SET WorkshopReservation.CancelDate =
DATEADD(DAY,14,WorkshopReservation.ReservationDate)
FROM inserted AS i
WHERE WorkshopReservation.WorkshopReservationID =
  i.WorkshopReservationID;
GO
```

# 7 Spis widoków

## 7.1 Widok danych klientów indywidualnych

**Kod:**

```
create view ShowIndividualCustomersInfo as
select ClientID, FirstName , LastName, Phone, Email from ClientDetails
go
```

## 7.2 Widok danych klientów firmowych

**Kod:**

```
create view ShowCopanyCustomersInfo as
select ClientID, CompanyName, NIP from CompanyDetails
go
```

### 7.3 Widok danych wszystkich klientów

**Kod:**

```
create view ShowAllCustomersInfo as
select a.ClientID, 'Company' as 'Customer Type', CompanyName
as 'Customer Name', Email   from CompanyDetails a
union
select b.ClientID, 'Individual'as 'Customer Type', b.FirstName+' '
+b.LastName as 'Customer Name', Email  from ClientDetails b
go
```

### 7.4 Widok ilości osób na poszczególnych dniach konferencji

**Kod:**

```
create view ShowDetailedCustomersConferenceAttendance as
select a.ClientID, (select b.ConferenceID from ConferenceDay b
where a.ConferenceDayID=b.ConferenceDayID) as ConferenceID,
a.NumberOfNonStudents+a.NumberOfStudents as NumberOfParticipants
from ConferenceDayReservation a
go
```

### 7.5 Widok ilości osób od danego klienta brało udział w konferencjach

**Kod:**

```
create view ShowCustomersConferenceAttendance as
select a.ClientID, a.ConferenceID, sum(a.NumberOfParticipants)
as NumberOfParticipants from ShowDetailedCustomersConferenceAttendance a
group by a.ConferenceID, a.ClientID
go
```

### 7.6 Widok ilości osób danego klienta brało udział we wszystkich konferencjach

**Kod:**

```
create view ShowCustomersTotalAttendance as
select a.ClientID, sum(a.NumberOfParticipants) as NumberOfParticipants
from ShowDetailedCustomersConferenceAttendance a
group by a.ClientID
go
```

## 7.7 Widok pokazujący informacje o konferencjach dodatkowo ile trwają dni

**Kod:**

```
create view ShowAllConferenceInfo as
select * ,(select count(*) from ConferenceDay b
where a.ConferenceID=b.ConferenceID) as NumberOfDays
from Conference a
go
```

## 7.8 Widok pokazujący ile dany klient wpłacił za wszystkie konferencje i ile ma do zapłaty

**Kod:**

```
create view ShowCustomersPayments as
select a.ClientID, sum(b.PricePaid) as MonneyPaid, sum(b.PriceToPay)
 as MoneyToPay from ConferenceDayReservation a
join Payment b on a.ConferenceDayReservationID = b.ConferenceDayReservationID
group by a.ClientID
go
```

## 7.9 Widok pokazujący klientów którzy jeszcze nie zapłacili za konferencję

**Kod:**

```
create view ShowCustomersWhoHaveToPay as
select a.ClientID, (select c.ConferenceID
from ConferenceDay c where c.ConferenceDayID =
 a.ConferenceDayID) as Conference  from ConferenceDayReservation a
join Payment b on a.ConferenceDayReservationID
= b.ConferenceDayReservationID
where b.PriceToPay >0
go
```

## 7.10 Widok klientów, którzy nie zapłacili za dni konferencji i ile zostało na to dni

**Kod:**

```
create view ShowTimeToPayForConferenceDay as
select a.ClientID, datediff(dd,getdate(),
(select d.ConferenceDay from ConferenceDay d
where d.ConferenceDayID = a.ConferenceDayID ))
 as TimeToPAy from  ConferenceDayReservation a
join Payment b on a.ConferenceDayReservationID=
b.ConferenceDayReservationID
```

```
where b.PriceToPay>0
go
```

### 7.11 Widok pokazujący dni konferencji i jakie osoby biorą w nich udział

**Kod:**

```
create view ShowConferenceDayAndParticipants as
select a.ConferenceDayID, b.ParticipantID
from ConferenceDayReservation a
join ConferenceDayRegistration b
on a.ConferenceDayReservationID=b.ConferenceDayReservationID
go
```

### 7.12 Widok pokazujący ile osób łącznie bierze udział w danym dniu konferencji

**Kod:**

```
create view ShowNumberOfParticipantsPerConferenceDay as
select a.ConferenceDayID, NumberOfParticipants+NumberOfStudents
as NumberOfParticipants
 from ConferenceDayReservation a
go
```

### 7.13 Widok pokazujący ile osób łącznie bierze udział w danej konferencji

**Kod:**

```
create view ShowNumberOfParticipantsPerConference as
select a.ConferenceID, sum(c.NumberOfNonStudents+c.NumberOfStudents)
as TotalNumberOfParticipants from Conference a
join ConferenceDay b on a.ConferenceID = b.ConferenceID
join ConferenceDayReservation c on b.ConferenceDayID = c.ConferenceDayID
group by a.ConferenceID
go
```

### 7.14 Widok pokazujący listę osób na dany warsztat

**Kod:**

```
create view ShowWorkShopParticipants as
select a.WorkshopID, p.ParticipantID from Workshop a
join WorkshopReservation b on a.WorkshopID
= b.WorkshopID
join WorkshopRegistration c
on b.WorkshopReservationID=c.WorkshopReservationID
join ConferenceDayRegistration d
on d.ConferenceDayReservationID = c.ConferenceDayRegistrationID
```

```
join Participant p on p.ParticipantID = d.ParticipantID
go
```

### 7.15 Widok pokazujący listę uczestników na dany dzień konferencji

**Kod:**

```
create view ShowDetailedConferenceParticipantsInfo as
select  b.ConferenceDay, b.ConferenceDayID, d.ParticipantID,
 e.FirstName, e.LastName, a.ConferenceName
from Conference a
join ConferenceDay b on a.ConferenceID=b.ConferenceID
join ConferenceDayReservation c on b.ConferenceDayID=
c.ConferenceDayID
join ConferenceDayRegistration d
on c.ConferenceDayReservationID=d.ConferenceDayReservationID
join Participant e on d.ParticipantID = e.ParticipantID
go
```

### 7.16 Widok pokazujący listę klientów z ich kwotą do zapłaty

**Kod:**

```
create view ShowUnpaidReservations as
select a.ClientID, b.ConferenceDayReservationID, c.PriceToPay from Client a
join ConferenceDayReservation b on a.ClientID=b.ClientID
join Payment c on b.ConferenceDayReservationID=c.ConferenceDayReservationID
where c.PriceToPay>0

go
```

### 7.17 Widok pokazujący Konferencję i jej ceny

**Kod:**

```
CREATE VIEW ConferenceWithPrices AS
 (SELECT C.ConferenceID,
         C.ConferenceName,
         CD.ConferenceDayID,
         CD.ConferenceDay AS 'FirstDay',
         CD.BasePrice * POWER(CP.PriceRate,
         DATEDIFF(WEEK,GETDATE(),MIN(CD.ConferenceDay)))
          AS 'CurrentPrice',
         CD.BasePrice,
         CP.StudentDiscount
 FROM Conference C
 JOIN ConferenceDay CD ON C.ConferenceID = CD.ConferenceID
 JOIN ConferencePrices CP ON C.ConferenceID = CP.ConferenceID
 GROUP BY C.ConferenceID, C.ConferenceName, CD.ConferenceDayID,
  CD.ConferenceDay, CD.BasePrice, CP.PriceRate, CP.StudentDiscount
 HAVING CD.ConferenceDay >= GETDATE()
)
GO
```

## 7.18 Widok pokazujący uczestników zarejestrowanych na dany dzień

**Kod:**

```
CREATE VIEW ConferenceDayRegistrationParticipants AS
  (
    SELECT CDRes.ConferenceDayReservationID,
           CDRes.NumberOfStudents
                AS 'MaxStudentNumber',
           CDRes.NumberOfNonStudents
                  AS 'MaxNonStudentNumber',
           (SELECT COUNT(P.ParticipantID) FROM Participant P
           WHERE P.ParticipantID = CDReg.ParticipantID
             AND P.ParticipantID
             IN (SELECT ParticipantID FROM Student))
              AS 'CurrentStudentsNumber',
           ((SELECT COUNT(P.ParticipantID) FROM Participant P
           WHERE P.ParticipantID = CDReg.ParticipantID)
              -
            (SELECT COUNT(P.ParticipantID) FROM Participant P
           WHERE P.ParticipantID = CDReg.ParticipantID
             AND P.ParticipantID IN
             (SELECT ParticipantID FROM Student)))
             AS 'CurrentNonStudentsNumber'
    FROM ConferenceDayRegistration CDReg
    RIGHT JOIN ConferenceDayReservation CDRes
      ON CDRes.ConferenceDayReservationID =
      CDReg.ConferenceDayReservationID
    GROUP BY CDReg.ConferenceDayRegistrationID
     ,CDRes.ConferenceDayReservationID ,CDReg.ConferenceDayReservationID
           ,CDRes.NumberOfStudents ,CDRes.NumberOfNonStudents
           ,CDReg.ParticipantID
  )
GO
```

## 7.19 Widok pokazujący uczestników warsztatów

**Kod:**

```
CREATE VIEW WorkshopParticipants AS
  (SELECT WorkshopID,
          Topic,
          NumberOfSeats, -- Max number of seats
          ISNULL((SELECT SUM(NumberOfStudents + NumberOfNonStudents)
          FROM WorkshopReservation WR
          WHERE WR.WorkshopID = W.WorkshopID AND
          (CancelDate IS NULL OR CancelDate > ReservationDate)),0)
           AS 'CurrentNumberOfSeats', -- Current reserved number of seats
          Price,
          Price * (1-(SELECT StudentDiscount from ConferencePrices
           WHERE ConferenceID =
          (SELECT ConferenceID from ConferenceDay
          WHERE ConferenceDayID =
         (SELECT ConferenceDayID FROM Workshop W2
```

```
            WHERE W2.WorkshopID = W.WorkshopID)))) AS 'StudentPrice'
  from Workshop W)
GO
```

## 7.20 Widok pokazujący uczestników konferencji wraz ze szczegółami

**Kod:**

```
CREATE VIEW ConferenceParticipant AS
  (SELECT (SELECT ConferenceName
  FROM Conference WHERE
    ConferenceID = CD.ConferenceID) AS 'ConferenceName',
           ConferenceDayID,
           ConferenceDay,
           NumberOfSeats, -- Max number of seats
           ISNULL((SELECT SUM(NumberOfStudents + NumberOfNonStudents)
           FROM ConferenceDayReservation CDR
           WHERE CDR.ConferenceDayID = CD.ConferenceDayID
           AND (CancelDate IS NULL OR CancelDate > ReservationDate)),0)
            AS 'CurrentNumberOfSeats' -- Current reserved number of seats
  FROM ConferenceDay CD)
GO
```

## 7.21 Widok pokazujący uczestników trwających warsztatóW

**Kod:**

```
CREATE VIEW ParticipantsCurrentWorkshop AS
  (SELECT P.ParticipantID,
        CD.ConferenceDayID,
        CD.ConferenceDay,
        W.WorkshopID,
        W.Topic,
        StartingTime,
        EndingTime
FROM Participant P
LEFT JOIN ConferenceDayRegistration CDR ON
 CDR.ParticipantID = P.ParticipantID
LEFT JOIN WorkshopRegistration WRG ON
 WRG.ConferenceDayRegistrationID = CDR.ConferenceDayRegistrationID
LEFT JOIN WorkshopReservation WRS ON
 WRS.WorkshopReservationID = WRG.WorkshopReservationID
LEFT JOIN Workshop W ON W.WorkshopID = WRS.WorkshopID
LEFT JOIN ConferenceDay CD ON W.ConferenceDayID = CD.ConferenceDayID)
GO
```

## 7.22 Widok pokazujący wszystkie warsztaty

**Kod:**

```
CREATE VIEW AllWorkshops AS
  (SELECT C.ConferenceID, C.ConferenceName, CD.ConferenceDayID, CD.ConferenceDa
FROM Conference C
```

```
JOIN ConferenceDay CD on C.ConferenceID = CD.ConferenceID
JOIN Workshop W on W.ConferenceDayID = CD.ConferenceDayID)
GO

CREATE VIEW ReservationTimes AS
  (SELECT WorkshopReservationID, W.WorkshopID, StartingTime, EndingTime
  FROM WorkshopReservation WR
    JOIN Workshop W ON W.WorkshopID=WR.WorkshopID
    )
GO
```

# 8 Spis procedur

## 8.1 Procedura wstawiania rekordu do tabeli Client

**Opis:** Procedura dodawania klienta do tabeli **Client** podając informację czy jest on firmą czy osobą fizyczną, po udanym wstawieniu zwracany jest jego ID.
**Kod:**

```
CREATE PROCEDURE dbo.AddClient(
        @ClientType NCHAR(1),
        @ClientID INT OUTPUT
        )
AS
 BEGIN
   SET NOCOUNT ON;
   BEGIN TRY
     BEGIN TRANSACTION
       BEGIN
         INSERT INTO Client (ClientType)
         VALUES (@ClientType)
         SET @ClientID = @@IDENTITY;
       END
     COMMIT TRANSACTION
   END TRY
   BEGIN CATCH
     ROLLBACK TRANSACTION
     DECLARE @ErrorMessage NVARCHAR(4000);
     SET @ErrorMessage = ERROR_MESSAGE();
     RAISERROR (@ErrorMessage,16,1)
   END CATCH
 END
```

## 8.2 Procedura wstawiania rekordu do tabeli IndividualClient

**Opis:** Procedura dodawania informacji o kliencie indywidualnym do tabeli **Clientdetails** podając informacje takie jak: ClientID,FirstName, LastName, Country, City, ZipCode, StreetName, StreetNumber, HouseNumber, Phone, Email.
**Kod:**

```
CREATE PROCEDURE dbo.AddClient(
        @ClientType NCHAR(1),
```

```
        @ClientID INT OUTPUT
        )
AS
 BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
      BEGIN TRANSACTION
        BEGIN
          INSERT INTO Client (ClientType)
          VALUES (@ClientType)
          SET @ClientID = @@IDENTITY;
        END
      COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
      ROLLBACK TRANSACTION
      DECLARE @ErrorMessage NVARCHAR(4000);
      SET @ErrorMessage = ERROR_MESSAGE();
      RAISERROR (@ErrorMessage,16,1)
    END CATCH
 END
```

## 8.3 Procedura wstawiania rekordu do tabeli CompanyDetails

**Opis:** Procedura dodawania informacji o kliencie firmowym do tabeli **CompanyDetails** podając informacje takie jak: ClientID,NIP ,CompanyName, Country, City, ZipCode, StreetName, StreetNumber, HouseNumber, Phone, Email.

**Kod:**

```
CREATE PROCEDURE dbo.AddCompany(
        @NIP NVARCHAR(20),
        @CompanyName NVARCHAR(20),
        @Country NVARCHAR(20),
        @City NVARCHAR(20),
        @ZipCode NCHAR(6),
        @StreetName NVARCHAR(20),
        @StreetNumber NVARCHAR(6),
        @HouseNumber NVARCHAR(6),
        @Phone NVARCHAR(10),
        @Email NVARCHAR(30)
        )
AS
 BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
      BEGIN TRANSACTION
        DECLARE @ClientID INT;
        BEGIN
          EXEC AddClient 'C', @ClientID = @ClientID OUTPUT
          INSERT INTO CompanyDetails (ClientID, NIP, CompanyName, Country,
            City, ZipCode, StreetName, StreetNumber, HouseNumber, Phone, Email)
          VALUES (@ClientID, @NIP, @CompanyName, @Country, @City, @ZipCode,
            @StreetName, @StreetNumber, @HouseNumber, @Phone, @Email)
```

```
        END
      COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
      ROLLBACK TRANSACTION
      DECLARE @ErrorMessage NVARCHAR(4000);
      SET @ErrorMessage = ERROR_MESSAGE();
      RAISERROR (@ErrorMessage,16,1)
    END CATCH
 END
```

## 8.4 Procedura wstawiania rekordu do tabeli Student

**Opis:** Procedura dodawania informacji o studencie do tabeli **Student** podając informacje takie jak: jego ParticipantID, StudentCard.
**Kod:**

```
CREATE PROCEDURE dbo.AddStudent(
        @ParticipantID INT,
        @StudentCardID INT
        )
AS
 BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
      BEGIN
        INSERT INTO Student
        VALUES (@ParticipantID, @StudentCardID)
      END
    END TRY
    BEGIN CATCH
      DECLARE @ErrorMessage NVARCHAR(4000);
      SET @ErrorMessage = ERROR_MESSAGE();
      RAISERROR (@ErrorMessage,16,1)
    END CATCH
 END
```

## 8.5 Procedura wstawiania rekordu do tabeli Participant

**Opis:** Procedura dodawania informacji o uczestniku do tabeli **Participant** podając informacje takie jak: jego ClientID, FirstName, LastName, Country, City, ZipCode, StreetName, StreetNumber, HouseNumber, Phone, Email.
**Kod:**

```
CREATE PROCEDURE dbo.AddParticipant(
        @ClientID INT,
        @FirstName NVARCHAR(20),
        @LastName NVARCHAR(30),
        @Country NVARCHAR(20),
        @City NVARCHAR(20),
        @ZipCode NCHAR(6),
        @StreetName NVARCHAR(20),
        @StreetNumber NVARCHAR(6),
```

```
        @HouseNumber NVARCHAR(6),
        @Phone NVARCHAR(10),
        @Email NVARCHAR(30),
        @StudentCardID INT
        )
AS
 BEGIN
   SET NOCOUNT ON;
   BEGIN TRY
     BEGIN TRANSACTION
       BEGIN
         INSERT INTO Participant (ClientID, FirstName, LastName,
          Country, City, ZipCode, StreetName, StreetNumber,
           HouseNumber, Phone, Email)
         VALUES (@ClientID, @FirstName, @LastName, @Country,
          @City, @ZipCode, @StreetName, @StreetNumber,
           @HouseNumber, @Phone, @Email)
         IF(@StudentCardID IS NOT NULL)
           EXEC AddStudent @@IDENTITY, @StudentCardID
       END
     COMMIT TRANSACTION
   END TRY
   BEGIN CATCH
     ROLLBACK TRANSACTION
     DECLARE @ErrorMessage NVARCHAR(4000);
     SET @ErrorMessage = ERROR_MESSAGE();
     RAISERROR (@ErrorMessage,16,1)
   END CATCH
 END
```

## 8.6 Procedura wstawiania rekordu do tabeli ConferencePrice

**Opis:** Procedura dodawania informacji o cenie konferencji do tabeli **ConferencePrice** podając informacje takie jak: jego ConferenceID, PriceRate, StudentDiscount

**Kod:**

```
CREATE PROCEDURE dbo.AddConferencePrice(
        @ConferenceID INT,
        @PriceRate NUMERIC(3,2),
        @StudentDiscount NUMERIC(3,2)
        )
AS
 BEGIN
   SET NOCOUNT ON;
   BEGIN TRY
     BEGIN
       INSERT INTO ConferencePrices
       VALUES (@ConferenceID, @PriceRate, @StudentDiscount)
     END
   END TRY
   BEGIN CATCH
     ROLLBACK
     DECLARE @ErrorMessage NVARCHAR(4000);
```

```
      SET @ErrorMessage = ERROR_MESSAGE();
      RAISERROR (@ErrorMessage,16,1)
    END CATCH
 END
```

## 8.7 Procedura wstawiania rekordu do tabeli Conference

**Opis:** Procedura dodawania informacji o konferencji do tabeli **Conference** podając informacje takie jak: ConferenceName, Country, City, ZipCode, StreetName, StreetNumber, HouseNumber
**Kod:**

```
CREATE PROCEDURE dbo.AddConference(
        @ConferenceName NVARCHAR(50),
        @Country NVARCHAR(20),
        @City NVARCHAR(20),
        @ZipCode NCHAR(6),
        @StreetName NVARCHAR(20),
        @StreetNumber NVARCHAR(6),
        @HouseNumber NVARCHAR(6),
        @PriceRate NUMERIC(3,2),
        @StudentDiscount NUMERIC(3,2)
        )
AS
 BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
      BEGIN TRANSACTION TEST
        BEGIN
          INSERT INTO Conference (ConferenceName, Country,
           City, ZipCode, StreetName, StreetNumber, HouseNumber)
          VALUES (@ConferenceName, @Country, @City,
           @ZipCode, @StreetName, @StreetNumber,
            @HouseNumber)
          EXEC AddConferencePrice @@IDENTITY,
           @PriceRate, @StudentDiscount
        END
      COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
      ROLLBACK TRANSACTION
      DECLARE @ErrorMessage NVARCHAR(4000);
      SET @ErrorMessage = ERROR_MESSAGE();
      RAISERROR (@ErrorMessage,16,1)
    END CATCH
 END
```

## 8.8 Procedura wstawiania rekordu do tabeli ConferenceDay

**Opis:** Procedura dodawania informacji o dniach konferencji do tabeli **ConferenceDay** podając informacje takie jak:ConferenceID, ConferenceDay, NumberOfSeats, BasePrice
**Kod:**

```
CREATE PROCEDURE dbo.AddConferenceDay(
```

```
        @ConferenceID INT,
        @ConferenceDay DATE,
        @NumberOfSeats INT,
        @BasePrice MONEY
 )
AS
 BEGIN
   SET NOCOUNT ON;
   BEGIN TRY
     BEGIN TRANSACTION
       BEGIN
         INSERT INTO ConferenceDay (ConferenceID,
          ConferenceDay, NumberOfSeats, BasePrice)
         VALUES (@ConferenceID, @ConferenceDay,
          @NumberOfSeats, @BasePrice)
       END
     COMMIT TRANSACTION
   END TRY
   BEGIN CATCH
     ROLLBACK TRANSACTION
     DECLARE @ErrorMessage NVARCHAR(4000);
     SET @ErrorMessage = ERROR_MESSAGE();
     RAISERROR (@ErrorMessage,16,1)
   END CATCH
 END
GO
```

## 8.9 Procedura wstawiania rekordu do tabeli Workshop

**Opis:** Procedura dodawania informacji o warsztatach do tabeli **Workshop** podając informacje takie ConferenceDayID, Topic, NumberOfSeats, Price, StartingTime, EndingTime
**Kod:**

```
CREATE PROCEDURE dbo.AddWorkshop(
        @ConferenceDayID INT,
        @Topic NVARCHAR(50),
        @NumberOfSeats INT,
        @Price MONEY,
        @StartingTime TIME,
        @EndingTime TIME
 )
AS
 BEGIN
   SET NOCOUNT ON;
   BEGIN TRY
     BEGIN TRANSACTION
       BEGIN
         INSERT INTO Workshop (ConferenceDayID,
          Topic, NumberOfSeats, Price, StartingTime, EndingTime)
         VALUES (@ConferenceDayID, @Topic,
          @NumberOfSeats, @Price, @StartingTime, @EndingTime)
       END
     COMMIT TRANSACTION
```

```
    END TRY
    BEGIN CATCH
      ROLLBACK TRANSACTION
      DECLARE @ErrorMessage NVARCHAR(4000);
      SET @ErrorMessage = ERROR_MESSAGE();
      RAISERROR (@ErrorMessage,16,1)
    END CATCH
  END
GO
```

## 8.10 Procedura odczytania ceny za konferencję

**Opis:** Procedura odczytywania ceny za konferencję podając informacje takie jak: ConferenceDayID, NumberOfParticipants, NumberOfStudents.
**Kod:**

```
CREATE PROCEDURE dbo.GetReservationPrice( -- Geting reservation price
        @ConferenceDayID INT,
        @NumberOfParticipants INT,
        @NumberOfStudents INT,
        @PriceToPay MONEY OUTPUT
  )
AS
  BEGIN
    SET NOCOUNT ON;
    DECLARE @ConferenceID INT;
    SET @ConferenceID = (SELECT ConferenceID FROM ConferenceDay
                         WHERE @ConferenceDayID = ConferenceDayID)
    SET @PriceToPay = (SELECT @NumberOfParticipants*CurrentPrice +
      @NumberOfStudents*CurrentPrice*(1-StudentDiscount)
                       FROM ConferenceWithPrices WHERE @ConferenceID
                        = ConferenceID)
  END
GO
```

## 8.11 Procedura dodania rezerwacji na dany dzień konferencji

**Opis:** Procedura dodaje rezerwację na dany dzień konferencji, podając dane takie jak: ClientID, ConferenceDayID, NumberOfNonStudents, NumberOfStudents.
**Kod:**

```
CREATE PROCEDURE dbo.AddConferenceDayReservation(
        @ClientID INT,
        @ConferenceDayID INT,
        @NumberOfNonStudents INT,
        @NumberOfStudents INT
   )
AS
   BEGIN
     SET NOCOUNT ON;
     DECLARE @ErrorMessage NVARCHAR(4000);
     BEGIN TRY
       BEGIN TRANSACTION
```

```
        BEGIN
          DECLARE @PriceToPay MONEY
          IF(@NumberOfStudents + @NumberOfNonStudents <=
           (SELECT NumberOfSeats - CurrentNumberOfSeats
            from ConferenceParticipant WHERE @ConferenceDayID = ConferenceDayID
            BEGIN
              EXEC GetReservationPrice @ConferenceDayID,
               @NumberOfNonStudents, @NumberOfStudents,
                @PriceToPay = @PriceToPay OUTPUT;
              INSERT INTO ConferenceDayReservation
              (ClientID, ConferenceDayID, ReservationDate,
               NumberOfNonStudents, NumberOfStudents)
              VALUES (@ClientID, @ConferenceDayID,
              GETDATE(), @NumberOfNonStudents, @NumberOfStudents)
              INSERT INTO Payment (ConferenceDayReservationID, PriceToPay)
              VALUES (@@IDENTITY, @PriceToPay)
            END
          ELSE
            BEGIN
              SET @ErrorMessage = 'Number of seats exceeded'
              RAISERROR (@ErrorMessage,16,1)
            END
          END
      COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
      ROLLBACK TRANSACTION
      SET @ErrorMessage = ERROR_MESSAGE();
      RAISERROR (@ErrorMessage,16,1)
    END CATCH
  END
GO
```

## 8.12 Procedura rejestrowania uczestnika na dzień konferencji

**Opis:** Procedura rejestruje danego uczestnika na dzień konferencji , wymaga podania informacji takich jak: ParticipantID, ConferenceDayReservationID

**Kod:**

```
CREATE PROCEDURE dbo.RegisterConferenceDay(
        @ParticipantID INT,
        @ConferenceDayReservationID INT
 )
AS
 BEGIN
   SET NOCOUNT ON;
   DECLARE @ErrorMessage NVARCHAR(4000);
   BEGIN TRY
     BEGIN TRANSACTION
       BEGIN
         IF (EXISTS (SELECT * FROM Student
         WHERE @ParticipantID = ParticipantID) AND
            (SELECT MaxStudentNumber
```

```
                from ConferenceDayRegistrationParticipants CDRP
                WHERE @ConferenceDayReservationID =
                 CDRP.ConferenceDayReservationID)
                  > (SELECT CurrentStudentsNumber
                   from ConferenceDayRegistrationParticipants CDRP
                    WHERE @ConferenceDayReservationID =
                     CDRP.ConferenceDayReservationID))
                   OR
                (EXISTS (SELECT * FROM Participant
                WHERE @ParticipantID = ParticipantID) AND
                (SELECT MaxNonStudentNumber
                from ConferenceDayRegistrationParticipants CDRP
                WHERE @ConferenceDayReservationID =
                 CDRP.ConferenceDayReservationID)
                  > (SELECT MaxNonStudentNumber
                    from ConferenceDayRegistrationParticipants CDRP
                     WHERE @ConferenceDayReservationID =
                      CDRP.ConferenceDayReservationID))
              BEGIN
                INSERT INTO ConferenceDayRegistration
                 (ConferenceDayReservationID, ParticipantID)
                VALUES (@ConferenceDayReservationID,@ParticipantID)
              END
            ELSE
              BEGIN
                SET @ErrorMessage = 'No more place for given day avaliable';
                RAISERROR (@ErrorMessage,16,1)
              END
          END
        COMMIT TRANSACTION
      END TRY
      BEGIN CATCH
        ROLLBACK TRANSACTION
        SET @ErrorMessage = ERROR_MESSAGE();
        RAISERROR (@ErrorMessage,16,1)
      END CATCH
  END
GO
```

## 8.13 Procedura dodania rezerwacji na warsztat

**Opis:** Procedura dodaje rezerwację na dany warsztat, podając dane takie jak: ConferenceDayReservationID, WorkshopID, NumberOfNonStudents, NumberOfStudents.

**Kod:**

```
REATE PROCEDURE dbo.AddWorkshopReservation( -- Adding Reservation of given day
          @ConferenceDayReservationID INT,
          @WorkshopID INT,
          @NumberOfNonStudents INT,
          @NumberOfStudents INT
    )
AS
    BEGIN
```

```
    SET NOCOUNT ON;
    DECLARE @ErrorMessage NVARCHAR(4000);
    BEGIN TRY
      BEGIN TRANSACTION
        BEGIN
          IF (GETDATE() < (SELECT ConferenceDay from ConferenceDay
          WHERE ConferenceDayID = (SELECT ConferenceDayID FROM
               ConferenceDayReservation WHERE
                @ConferenceDayReservationID = ConferenceDayReservationID)))
              AND
            (@NumberOfStudents + @NumberOfNonStudents <=
              (SELECT NumberOfStudents + NumberOfNonStudents
              from ConferenceDayReservation where ConferenceDayReservationID
               = @ConferenceDayReservationID ))
             (@NumberOfNonStudents + @NumberOfStudents <
              (SELECT NumberOfSeats - CurrentNumberOfSeats
              FROM WorkshopParticipants WHERE WorkshopID = @WorkshopID)) -- End
            BEGIN
              DECLARE @PriceToPay MONEY = (SELECT @NumberOfStudents*StudentPric
               + @NumberOfNonStudents*Price
               FROM WorkshopParticipants WHERE WorkshopID = @WorkshopID)
              INSERT INTO WorkshopReservation (WorkshopID,
               ConferenceDayReservationID, ReservationDate,
                PriceToPay, NumberOfNonStudents, NumberOfStudents)
              VALUES (@WorkshopID, @ConferenceDayReservationID,
               GETDATE(), @PriceToPay, @NumberOfNonStudents, @NumberOfStudents)
            END
          ELSE
            BEGIN
              SET @ErrorMessage = 'Adding Workshop Reservation Failed';
              RAISERROR (@ErrorMessage,16,1)
            END
        END
      COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
      ROLLBACK TRANSACTION
      SET @ErrorMessage = ERROR_MESSAGE();
      RAISERROR (@ErrorMessage,16,1)
    END CATCH
  END
GO
```

## 8.14 Procedura rejestrowania się na warsztat

**Opis:** Procedura tworzy rejestrację danego uczestnika na warsztat pod warunkiem, że jest on zarejestrowany na konferencję w ramach której jest warsztat. Wymaga podania informacji takich jak: WorkshopReservationID, ParticipantID

**Kod:**

```
CREATE PROCEDURE dbo.RegisterWorkshop( -- Participant Register Given workshop i
        @WorkshopReservationID INT,
        @ConferenceDayRegistrationID INT
```

```
  )
AS
  BEGIN
    SET NOCOUNT ON;
    DECLARE @ErrorMessage NVARCHAR(4000);
    BEGIN TRY
      BEGIN TRANSACTION
        BEGIN
          IF ((SELECT COUNT(*) FROM WorkshopReservation
      JOIN AllWorkshops AW ON(
      (SELECT ConferenceDayReservationID
      FROM WorkshopReservation
      WHERE @WorkshopReservationID =
       WorkshopReservationID) = ConferenceDayID
    AND (SELECT StartingTime from ReservationTimes
     WHERE WorkshopReservationID = @WorkshopReservationID)
     BETWEEN AW.StartingTime AND AW.EndingTime
       AND (SELECT EndingTime from ReservationTimes
        WHERE WorkshopReservationID = @WorkshopReservationID)
         BETWEEN AW.StartingTime AND AW.EndingTime
               )) = 0 ) -- Times not oveplapping
     BEGIN
            INSERT INTO WorkshopRegistration
            (WorkshopReservationID, ConferenceDayRegistrationID)
            VALUES (@WorkshopReservationID,@ConferenceDayRegistrationID)
          END
          ELSE
            BEGIN
              RAISERROR (@ErrorMessage,16,1)
            END
        END
      COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
      ROLLBACK TRANSACTION
      SET @ErrorMessage = ERROR_MESSAGE();
      RAISERROR (@ErrorMessage,16,1)
    END CATCH
  END
GO
```

## 8.15 Procedura usuwania rezerwacji z dnia konferencji

**Opis:** Usuwa rezerwację z danego dnia konferencji po podaniu ID rezerwacji dnia konferencji.
**Kod:**

```
CREATE PROCEDURE dbo.CancelDayReservation(
    @ConferenceDayReservationID INT
  )
AS
  BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
```

```
    BEGIN TRANSACTION
      BEGIN
        UPDATE ConferenceDayReservation
        SET CancelDate = GETDATE()
        WHERE ConferenceDayReservationID =
          @ConferenceDayReservationID
      END
    COMMIT TRANSACTION
  END TRY
  BEGIN CATCH
    ROLLBACK TRANSACTION
    DECLARE @ErrorMessage NVARCHAR(4000);
    SET @ErrorMessage = ERROR_MESSAGE();
    RAISERROR (@ErrorMessage,16,1)
  END CATCH
 END
GO
```

## 8.16 Procedura usuwania rezerwacji z warsztatu

**Opis:** Usuwa rezerwację z warsztatu po podaniu Id rezerwacji na warsztat.
**Kod:**

```
CREATE PROCEDURE dbo.CancelWorkshopReservation(
  @WorkshopReservationID INT
 )
AS
 BEGIN
  SET NOCOUNT ON;
  BEGIN TRY
    BEGIN TRANSACTION
      BEGIN
        UPDATE WorkshopReservation
        SET CancelDate = GETDATE()
        WHERE WorkshopReservationID =
          @WorkshopReservationID
      END
    COMMIT TRANSACTION
  END TRY
  BEGIN CATCH
    ROLLBACK TRANSACTION
    DECLARE @ErrorMessage NVARCHAR(4000);
    SET @ErrorMessage = ERROR_MESSAGE();
    RAISERROR (@ErrorMessage,16,1)
  END CATCH
 END
GO
```

## 8.17 Procedura wybrania rodzaju płatności

**Opis:** Pozwala wybrać sposób płatności za dni konferencji, po podaniu ID rezerwacji danego dnia konferencji i sposobu płatności.
**Kod:**

```
CREATE PROCEDURE dbo.PickPaymentMethod(
    @ConferenceDayReservationID INT,
    @PaymentMethod NVARCHAR(20)
 )
AS
 BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
      BEGIN TRANSACTION
        BEGIN
          UPDATE Payment
          SET PaymentMethod = @PaymentMethod
          WHERE ConferenceDayReservationID =
            @ConferenceDayReservationID
        END
      COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
      ROLLBACK TRANSACTION
      DECLARE @ErrorMessage NVARCHAR(4000);
      SET @ErrorMessage = ERROR_MESSAGE();
      RAISERROR (@ErrorMessage,16,1)
    END CATCH
 END
GO
```

# 9 INDEXY

Najważniejsze pola do wyszukiwania są polami PRIMARY KEY oraz UNIQUE, zatem mają automatycznie założone indexy, poza tymi polami zakładamy 2 indexy, w celu zwiększenia szybkości wyszukiwania nakładających się warsztatów podczas próby rejestrowania warsztatu

**Kod:**

```
CREATE INDEX start_time ON Workshop(StartingTime)
GO
CREATE INDEX end_time ON Workshop(EndingTime)
GO
```

# 10 Proponowane role w systemie

## 10.1 Klient

Klienci powinni mieć swoją rolę w systemie pozwalającą im rezerwować miejsca na dni konferencji oraz na warsztaty, oprócz tego przeglądać wszystkie szczegóły związane z tym wydarzeniem. Co więcej powinien mieć możliwość zapłacenia za zarezerwowane warsztaty i konferencję oraz na dodanie płatności, które wykonuję, żeby organizator konferencji miał do nich dostęp.

## 10.2 Uczestnik

Uczestnik powinien mieć swoją rolę w systemie pozwalającą przeglądać mu dni konferencji oraz warsztaty, co więcej jeżeli posiada voucher od klienta na dany dzień konferencji, powinien móc się zarejestrować na dany dzień konferencji korzystając z niego. Następnie jeżeli chodzi o warsztaty, to powinien mieć możliwość rezerwowania danego warsztatu, jak i rejestracji na dany warsztat pod warunkiem wcześniejszego zarezerwowania dnia konferencji i posiadania rezerwacji na warsztat (dokonanej przez siebie bądź klienta).

## 10.3 Pracownik Firmy Organizującej Wydarzenia

Pracownicy firmy organizującej wydarzenia powinni mieć swoją rolę, która pozwala im na dodawanie konferencji, warsztatów usuwania ich oraz generowania odpowiednich raportów w celach organizacyjnych dla organizatora.

## 10.4 Administrator

Administrator powinien mieć dostęp do pełnego modyfikowania bazy danych, w szczególności, tylko on powinien mieć prawa do usuwania informacji o klientach oraz uczestnikach jeżeli będzie taka potrzeba.