Dokumentacja projektu zarządzania konferencjami

Przedmiot: Podstawy Baz Danych

Rok II, semestr 3

Mateusz Kubicki Jacek Kuśnierz

Styczeń 2019

Spis treści

1 Opis systemu **5**

2 Schemat bazy danych **5**

3 Opis tabel **5**
3.1 Tabela Conferences . . . 5
3.2 Tabela ConferencePrice . . . 6
3.3 Tabela Workshops . . . 6
3.4 Tabela Days . . . 7
3.5 Tabela ConferenceReservations . . . 7
3.6 Tabela ConferencePayments . . . 8
3.7 Tabela Clients . . . 8
3.8 Tabela Company . . . 8
3.9 Tabela ConferenceReservationParticipants . . . 9
3.10 Tabela WorkshopReservation . . . 9
3.11 Tabela WorkshopReservationParticipants . . . 10
3.12 Tabela Participants . . . 10
3.13 Tabela Students . . . 10

4 Widoki **11**
4.1 Uczestnicy nadchodzących konferencji . . . 11
4.2 Uczestnicy nadchodzących warsztatów . . . 11
4.3 Opłaty klientów za konferencje . . . 11
4.4 Identyfikator imienny uczestnika konferencji i warsztatów . . . 11
4.5 Liczba uczestników konferencji w poszczególnych dniach . . . 12
4.6 Liczba uczestników warsztatów w poszczególnych dniach . . . 12
4.7 Klienci, którzy anulowali rezerwację na konferencję . . . 12
4.8 Zwrot opłaty za anulowaną rezerwację na konferencje . . . 12
4.9 Klienci, którzy anulowali rezerwację na warsztat . . . 13
4.10 Informacja jaki procent uczestników konferencji to studenci . . . 13
4.11 Informacja jaki procent uczestników warsztatu to studenci . . . 13
4.12 Lista klientów którzy nie dokonali zapłaty . . . 13
4.13 Progi cenowe konferencji . . . 13
4.14 Dane o kliencie . . . 14
4.15 Ilość dokonanych rezerwacji przez klienta . . . 14
4.16 Pokazuje klientów, którzy dokonali nadpłaty . . . 14

5 Procedury **14**
5.1 Dodawanie klienta . . . 14
5.2 Dodawanie konferencji . . . 15
5.3 Dodawanie rezerwacji do konferencji . . . 16
5.4 Dodawanie warsztatu do konferencji . . . 17
5.5 Dodawanie rezerwacji do warsztatu . . . 18
5.6 Dodawanie uczestnika . . . 19
5.7 Dodawanie nowego progu cenowego . . . 20
5.8 Przypisanie nowego okresu ważności karty studenckiej . . . 20
5.9 Przypisanie nowej opłaty . . . 21
5.10 Przypisanie uczestnika do rezerwacji . . . 22
5.11 Przypisanie uczestnika do warsztatu . . . 22
5.12 Anulowanie konferencji . . . 23

5.13 Anulowanie rezerwacji konferencji . . . 23
5.14 Anulowanie warsztatu . . . 24
5.15 Anulowanie rezerwacji warsztatu . . . 24
5.16 Wyświetlenie danych o konkretnej konferencji . . . 24
5.17 Wyświetlenie opłat dla konkretnej rezerwacji . . . 25

6 Triggery **25**
6.1 Trigger sprawdzający przepełnienie konferencji . . . 25
6.2 Trigger sprawdzający przepełnienie warsztatu . . . 26
6.3 Trigger ustawiający wartość anulowania rezerwacji na True w przypadku, gdy takiego anulowania dokonano . . . 26
6.4 Analogiczny do poprzedniego trigger - anulujący rezerwacje na warsztat, jeśli anulowano rezerwację konferencję, w ramach której warsztat się odbywa . . . 27
6.5 Trigger sprawdzający czy ta sama osoba nie została uwzględniona w rezerwacji warsztatu więcej niż jeden raz . . . 27
6.6 Analogiczny trigger sprawdzający czy ta sama osoba nie została uwzględniona w rezerwacji konferencji więcej niż jeden raz . . . 27
6.7 Trigger aktualizujący cenę konferencji przy dodaniu nowego warsztatu . . . 28

7 Indeksy **28**
7.1 FK ConferenceID w tabeli ConferencePrice . . . 28
7.2 FK ConferenceID w tabeli Days . . . 29
7.3 FK DayID w tabeli Workshops . . . 29
7.4 FK ClientID w tabeli Company . . . 29
7.5 FK ReservationID w tabeli ConferencePayments . . . 29
7.6 FK DayID w tabeli ConferenceReservations . . . 29
7.7 FK ClientID w tabeli ConferenceReservations . . . 29
7.8 FK ParticipantID w tabeli Students . . . 30
7.9 FK ReservationID w tabeli ConferenceReservationParticipants . . . 30
7.10 FK ParticipantID w tabeli ConferenceReservationParticipants . . . 30
7.11 FK ParticipantID w tabeli WorkshopReservationParticipants . . . 30
7.12 FK WorkshopReservationID w tabeli WorkshopReservationParticipants . . . 30
7.13 FK WorkshopID w tabeli WorkshopReservation . . . 31
7.14 FK ReservationID w tabeli WorkshopReservation . . . 31
7.15 TimeBegin w tabeli Workshops . . . 31
7.16 TimeEnd w tabeli Workshops . . . 31
7.17 Date w tabeli Days . . . 31
7.18 DateBegin w tabeli Conferences . . . 31
7.19 DateEnd w tabeli Conferences . . . 32

8 Role **32**

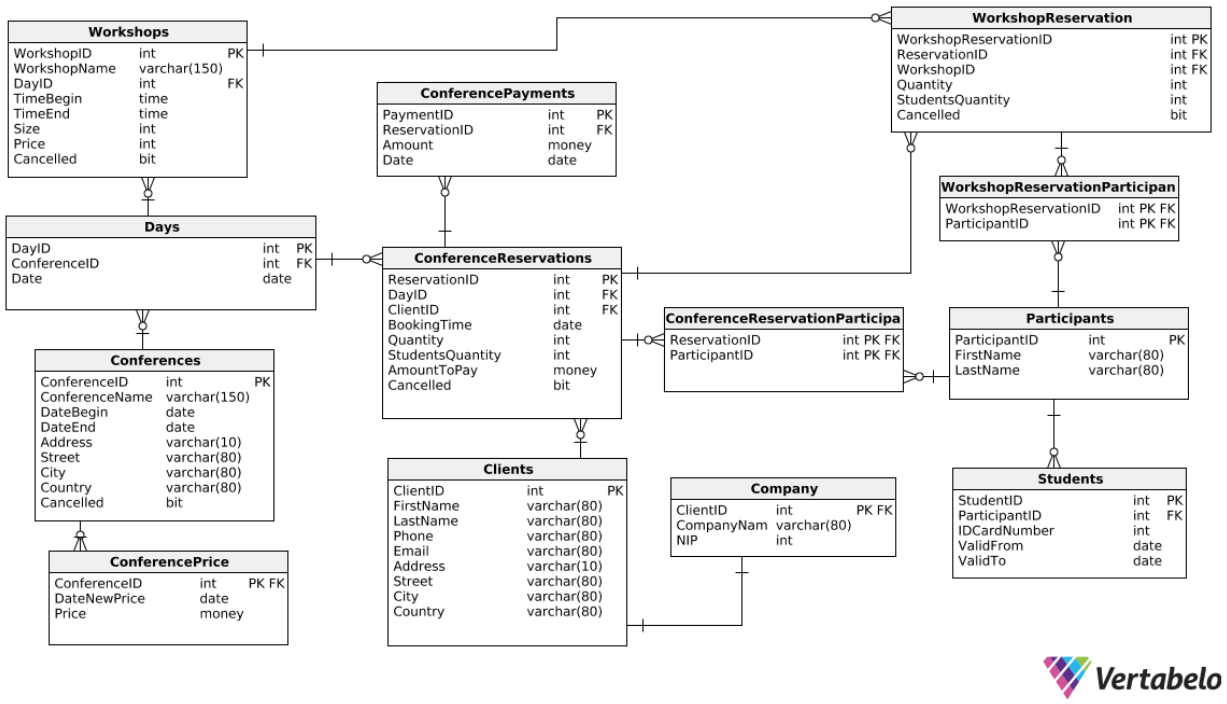
9 Generator **32**
9.1 Clients . . . 32
9.2 Participants . . . 33
9.3 Students . . . 33
9.4 Conferences . . . 33
9.5 Workshops . . . 34
9.6 Reservations . . . 34
9.7 Payments . . . 34
9.8 Adding to reservations . . . 35
9.9 MainFunction . . . 35

9.10 Conferences Sample Output 35

1 Opis systemu

Celem projektu było zaplanowanie systemu bazodanowego dla firmy, która organizuje konferencje. Mogą one być jedno- lub kilkudniowe. Klientami mogą być zarówno indywidualne osoby jak i firmy, natomiast uczestnikami konferencji są osoby (firma nie musi podawać od razu przy rejestracji listy uczestników - może zarezerwować odpowiednią ilość miejsc na określone dni oraz na warsztaty, natomiast na 2 tygodnie przed rozpoczęciem musi te dane uzupełnić). Dla konferencji kilkudniowych, uczestnicy mogą rejestrować na dowolne z tych dni, dowolną liczbę osób. Klient może zmienić liczbę osób na rezerwacji, lub całkiem ją anulować (do 2 tygodni przed konferencją). Zarówno dni konferencji, jak i warsztaty mogą być płatne, jednak w przypadku tych pierwszych cena nie jest stała, a zależy od terminu, w którym dokonujemy opłaty za rezerwację. Od ceny tej obowiązują przypisane do konferencji informacje o zniżce dla uczestników, którzy są studentami.

2 Schemat bazy danych



3 Opis tabel

3.1 Tabela Conferences

Opisuje konferencję organizowaną przez firmę. Informuje o jej lokalizacji (adresie) i dacie rozpoczęcia oraz zakończenia danej konferencji.

ConferenceID - Autoinkrementowany identyfikator konferencji.
ConferenceName - Nazwa danej konferencji
DateBegin - Data rozpoczęcia konferencji.

DateEnd - Data zakończenia konferencji.
Address - Numer budynku/budynku i lokalu, w kótrym odbywa się konferencja.
Street - Ulica, na której konferencja się odbywa.
City - Miasto, w którym konferencja się odbywa.
Country - Kraj, w którym konferencja się odbywa.
Cancelled - Określa czy konferencja została anulowana (domyślnie 0).

```
CREATE TABLE Conferences (
    ConferenceID int IDENTITY(1,1) NOT NULL,
    ConferenceName varchar(150) NOT NULL,
    DateBegin date NOT NULL,
    DateEnd date NOT NULL,
    Address varchar(10) NOT NULL,
    Street varchar(80) NOT NULL,
    City varchar(80) NOT NULL,
    Country varchar(80) NOT NULL,
    Cancelled bit NOT NULL DEFAULT 0,
    CONSTRAINT ValidDate CHECK (DATEDIFF (day, DateBegin, DateEnd) > 0),
    CONSTRAINT Conferences_pk PRIMARY KEY (ConferenceID)
);
```

3.2 Tabela ConferencePrice

Informuje o cenie danej konferencji.

ConferenceID - Identyfikator konferencji (jest to także klucz obcy).
DateNewPrice - Data, od której zaczyna się wyższy próg cenowy danej konferencji.
Price - Podstawowa cena konferencji.

```
CREATE TABLE ConferencePrice (
    ConferenceID int NOT NULL,
    DateNewPrice date NOT NULL,
    Price money NOT NULL CHECK (Price >=0),
    CONSTRAINT ConferencePrice_pk PRIMARY KEY (ConferenceID,DateNewPrice)
);
```

3.3 Tabela Workshops

Informuje o warsztatach odbywających się w ramach danej konferencji.

WorkshopID - Autoinkrementowany identyfikator warsztatu.
WorkshopName - Nazwa danego warsztatu
DayID - Klucz obcy, określa w jakim dniu konferencji odbywa się dany warsztat.
TimeBegin - Godzina rozpoczęcia warsztatu.
TimeEnd - Godzina zakończenia warsztatu.
Size - Maksymalna liczba osób, która może uczestniczyć w warsztacie.
Price - Cena warsztatu.
Cancelled - Określa czy warsztat został anulowany (domyślnie 0).

```
CREATE TABLE Workshops (
    WorkshopID int IDENTITY(1,1) NOT NULL,
```

```
    WorkshopName varchar(150) NOT NULL,
    DayID int NOT NULL,
    TimeBegin time NOT NULL,
    TimeEnd time NOT NULL,
    Size int NOT NULL CHECK (Size>=0),
    Price money NOT NULL CHECK (Price>=0),
    Cancelled bit NOT NULL DEFAULT 0,
    CONSTRAINT ValidTime CHECK (TimeBegin <= TimeEnd),
    CONSTRAINT Workshops_pk PRIMARY KEY (WorkshopID)
);
```

3.4 Tabela Days

Informuje nas o dniach, w których odbywa się konferencja.

DayID - Autoinkrementowany identyfikator dnia.
ConferenceID - Klucz obcy, identyfikator konferencji.
Date - Informuje o dacie danego DayID.

```
CREATE TABLE Days (
    DayID int IDENTITY(1,1) NOT NULL,
    ConferenceID int NOT NULL,
    Date date NOT NULL,
    CONSTRAINT Days_pk PRIMARY KEY (DayID)
);
```

3.5 Tabela ConferenceReservations

Informuje o dokonanych rezerwacjach na konkretny dzień danej konferencji.

ReservationID - Autoinkrementowany identyfikator rezerwacji.
DayID - Klucz obcy, identyfikator dnia, na który złożono rezerwację.
ClientID - Klucz obcy, identyfikator klienta, który dokonał rezerwacji.
BookingTime - Data i godzina, w której klient złożył rezerwację.
Quantity - Określa ile miejsc zostało zarezerwowanych dla uczestników, którzy nie są studentami.
StudentsQuantity - Informuje ile miejsc zostało zarezerwowanych dla uczestników, którzy są studentami.
AmountToPay - Określa ile należy zapłacić za dokonaną rezerwację.
Cancelled - Informuje czy dokonana rezerwacja została anulowana (domyślnie wartość 0).

```
CREATE TABLE ConferenceReservations (
    ReservationID int IDENTITY(1,1) NOT NULL,
    DayID int NOT NULL,
    ClientID int NOT NULL,
    BookingTime date NOT NULL,
    Quantity int NOT NULL CHECK (Quantity>=0),
    StudentsQuantity int NOT NULL CHECK (StudentsQuantity>=0),
    AmountToPay money NOT NULL CHECK (AmountToPay>=0),
    Cancelled bit NOT NULL DEFAULT 0,
    CONSTRAINT ConferenceReservations_pk PRIMARY KEY (ReservationID)
);
```

3.6 Tabela ConferencePayments

Przechowuje informacje o wpłatach klientów za zarezerwowane miejsca.

PaymentID - Autoinkrementowany identyfikator płatności.
ReservationID - Klucz obcy, identyfikator rezerwacji za którą klient dokonał płatności.
Amount - Ilość wpłaconych pieniędzy.
Date - Data dokonania wpłaty.

```
CREATE TABLE ConferencePayments (
    PaymentID int IDENTITY(1,1) NOT NULL,
    ReservationID int NOT NULL,
    Amount money NOT NULL CHECK (Amount>=0),
    Date date NOT NULL,
    CONSTRAINT ConferencePayments_pk PRIMARY KEY (PaymentID)
);
```

3.7 Tabela Clients

Zawiera informacje o klientach korzystających z usług systemu. Może być to klient indywidualny lub reprezentant pewnej firmy dokonujący zamówienia tylko dla siebie lub dla większej grupy osób.

ClientID - Autoinkrementowany identyfikator klienta, który dokonał rezerwacji.
FirstName - Imię klienta.
LastName - Nazwisko klienta.
Phone - Numer telefonu kontaktowego klienta.
Email - Adres poczty elektronicznej klienta.
Address - Numer budynku/budynku i mieszkania, podany jako część adresu do korespondencji klienta.
Street - Ulica, jako część adresu do korespondencji klienta.
City - Miasto, jako część adresu do korespondencji klienta.
Country - Kraj, jako część adresu do korespondencji klienta.

```
CREATE TABLE Clients (
    ClientID int IDENTITY(1,1) NOT NULL,
    FirstName varchar(80) NOT NULL,
    LastName varchar(80) NOT NULL,
    Phone varchar(80) NOT NULL CHECK(ISNUMERIC(Phone)=1),
    Email varchar(80) UNIQUE NOT NULL,
    Address varchar(10) NOT NULL,
    Street varchar(80) NOT NULL,
    City varchar(80) NOT NULL,
    Country varchar(80) NOT NULL,
    CONSTRAINT Clients_pk PRIMARY KEY (ClientID)
);
```

3.8 Tabela Company

Zawiera informację o firmie, którą reprezentuje dany klient. Oczywiście zawiera informacje tylko o tych klientach, którzy nie są klientami indywidualnymi.

ClientID - Identyfikator klienta, który dokonał rezerwacji i jest przedstawicielem pewnej firmy.
CompanyName - Nazwa firmy, której klient jest przedstawicielem.
NIP - NIP firmy, w przypadku firm, które go nie posiadają (np. zagraniczne) pole przyjmuje wartość null.

```
CREATE TABLE Company (
    ClientID int NOT NULL,
    CompanyName varchar(80) NOT NULL,
    NIP int NOT NULL CHECK (NIP>=100000000),
    CONSTRAINT Company_pk PRIMARY KEY (ClientID)
);
```

3.9 Tabela ConferenceReservationParticipants

Informuje o uczestnikach konferencji, których wymieniono w danej rezerwacji. Pełni funkcję tabeli łączącej.

ReservationID - Klucz obcy, identyfikator rezerwacji konferencji, w której weźmie udział dany uczestnik.
ParticipantID - Klucz obcy, identyfikator uczestnika konferencji, wymienionego przy składaniu danej rezerwacji.

```
CREATE TABLE ConferenceReservationParticipants (
    ReservationID int NOT NULL,
    ParticipantID int NOT NULL
    CONSTRAINT ConferenceReservationParticipants_pk PRIMARY KEY (ReservationID,ParticipantID)
);
```

3.10 Tabela WorkshopReservation

Informuje o dokonanych rezerwacjach na dany warsztat.

Tabela WorkshopReservationID - Autoinkrementowany identyfikator rezerwacji warsztatu.
ReservationID - Klucz obcy, określa w jakiej rezerwacji zarezerwowano dany warsztat.
WorkshopID - Klucz obcy, informuje na jaki warsztat dokonano rezerwacji.
Quantity - Określa ile miejsc zostało zarezerwowanych dla uczestników, którzy nie są studentami.
StudentsQuantity - Informuje ile miejsc zostało zarezerwowanych dla uczestników, którzy są studentami.
Cancelled - Określa czy rezerwację warsztatu anulowano. Domyślnie 0.

```
CREATE TABLE WorkshopReservation (
    WorkshopReservationID int IDENTITY(1,1) NOT NULL,
    ReservationID int NOT NULL,
    WorkshopID int NOT NULL,
    Quantity int NOT NULL CHECK (Quantity>=0),
    StudentsQuantity int NOT NULL CHECK (StudentsQuantity>=0),
    Cancelled bit NOT NULL DEFAULT 0,
    CONSTRAINT WorkshopReservation_pk PRIMARY KEY (WorkshopReservationID)
);
```

3.11 Tabela WorkshopReservationParticipants

Informuje o uczestnikach warsztatu, których wymieniono w danej rezerwacji. Pełni funkcję tabeli łączącej.

WorkshopReservationID - Klucz obcy, identyfikator rezerwacji warsztatu, w którym weźmie udział dany uczestnik.
ParticipantID Klucz obcy, identyfikator uczestnika warsztatu, wymienionego przy składaniu danej rezerwacji.

```
CREATE TABLE WorkshopReservationParticipants (
    WorkshopReservationID int NOT NULL,
    ParticipantID int NOT NULL
    CONSTRAINT WorkshopReservationParticipants_pk PRIMARY KEY (WorkshopReservationID,ParticipantID)
);
```

3.12 Tabela Participants

Zawiera informacje o uczestnikach konferencji i/lub warsztatów.

ParticipantID - Autoinkrementowany identyfikator uczestnika.
FirstName - Imię uczestnika.
LastName - Nazwisko uczestnika.

```
CREATE TABLE Participants (
    ParticipantID int IDENTITY(1,1) NOT NULL,
    FirstName varchar(80) NOT NULL,
    LastName varchar(80) NOT NULL,
    CONSTRAINT Participants_pk PRIMARY KEY (ParticipantID)
);
```

3.13 Tabela Students

Przechowuje informacje o uczestnikach konferencji i/lub warsztatów, którzy są studentami. Relacja tabeli Participants z tabelą Students to 1 do wielu, ponieważ uczestnik może być studentem w różnych przedziałach czasu (np. ze względu na urlop dziekański). O statusie studenta w danym okresie czasu będzie decydować ważność jego legitymacji w tym okresie.

StudentID - Autoinkrementowany identyfikator studenta.
ParticipantID - Klucz obcy, ogólny identyfikator uczestnika.
IDCardNumber - Numer legitymacji studenckiej uczestnika. W przypadku studentów zagranicznych - nr legitymacji ISIC.
ValidFrom - Data, od której legitymacja jest ważna.
ValidTo - Data, od której ważność legitymacji wygasa i osoba traci status studenta.

```
CREATE TABLE Students (
    StudentID int IDENTITY(1,1) NOT NULL,
    ParticipantID int NOT NULL,
    IDCardNumber int NOT NULL CHECK (IDCardNumber>=100000),
    ValidFrom date NOT NULL,
```

```
    ValidTo date NOT NULL,
   CONSTRAINT ValidID CHECK (DATEDIFF (day, ValidFrom, ValidTo) > 0),
    CONSTRAINT Students_pk PRIMARY KEY (StudentID)
);
```

4 Widoki

4.1 Uczestnicy nadchodzących konferencji

```
CREATE VIEW UpcomingConferencesParticipants AS
select distinct p.ParticipantID, p.FirstName,p.LastName, c.CompanyName from Participants p
join ConferenceReservationParticipants crp on crp.ParticipantID = p.ParticipantID
join ConferenceReservations cr on
crp.ReservationID = cr.ReservationID and cr.Cancelled = 0
join Days on Days.DayID = cr.DayID
join Conferences con on Days.ConferenceID = con.ConferenceID
and con.Cancelled = 0
join Clients cl on cl.ClientID = cr.ClientID
join Company c on c.ClientID = cl.ClientID
```

4.2 Uczestnicy nadchodzących warsztatów

```
CREATE VIEW WorkshopParticipants AS
select distinct p.ParticipantID, p.FirstName,p.LastName, w.WorkshopID from Participants p
join WorkshopReservationParticipants wrp
on wrp.ParticipantID = p.ParticipantID
join WorkshopReservation wr
on wrp.WorkshopReservationID = wr.WorkshopReservationID
join Workshops w on w.WorkshopID = wr.WorkshopID
```

4.3 Opłaty klientów za konferencje

```
CREATE VIEW ClientsPayments AS
select distinct c.ClientID, c.FirstName,c.LastName, com.CompanyName, cp.Amount, con.ConferenceID
    from Clients c
join Company com on c.ClientID = com.ClientID
join ConferenceReservations cr on
c.ClientID = cr.ClientID and cr.Cancelled = 0
join ConferencePayments cp on cp.ReservationID = cr.ReservationID
join Days on Days.ConferenceID = cr.DayID
join Conferences con on Days.ConferenceID = con.ConferenceID
and con.Cancelled = 0
```

4.4 Identyfikator imienny uczestnika konferencji i warsztatów

```
CREATE VIEW ParticipantNameIdentifier as
select p.FirstName, p.LastName, c.CompanyName, con. ConferenceID, wr.WorkshopID from Participants p
```

```
join ConferenceReservationParticipants crp on crp.ParticipantID = p.ParticipantID
join ConferenceReservations cr on
crp.ReservationID = cr.ReservationID and cr.Cancelled = 0
join Clients cl on cl.ClientID = cr.ClientID
join Company c on c.ClientID = cl.ClientID
join Days on Days.DayID = cr.DayID
join Conferences con on Days.ConferenceID = con.ConferenceID and con.Cancelled = 0
join WorkshopReservation wr on wr.ReservationID= cr.ReservationID
```

4.5 Liczba uczestników konferencji w poszczególnych dniach

```
CREATE VIEW ConfParticipantsCount as
select Days.Date, c.ConferenceID, c.ConferenceName, count(*) as totalParticipants from Days
join Conferences c on c.ConferenceID = Days.ConferenceID
join ConferenceReservations cr on Days.DayID = cr.DayID
join ConferenceReservationParticipants crp on crp.ReservationID = cr.ReservationID
group by Days.Date, c.ConferenceID, c.ConferenceName
```

4.6 Liczba uczestników warsztatów w poszczególnych dniach

```
create VIEW WorkshopParticipantsCount as
select Days.Date, w.WorkshopID, w.WorkshopName, count(*) as totalParticipants from Days
join Workshops w on w.DayID = Days.DayID
join WorkshopReservation wr on w.WorkshopID = wr.WorkshopID
join WorkshopReservationParticipants wrp on wrp.WorkshopReservationID = wr.WorkshopReservationID
group by Days.Date, w.WorkshopID, w.WorkshopName
```

4.7 Klienci, którzy anulowali rezerwację na konferencję

```
CREATE VIEW CancelledConfReservations as
select con.ConferenceName, Days.Date, c.FirstName,c.LastName, cmp.CompanyName from Clients c join
ConferenceReservations cr
on c.ClientID = cr.ClientID
join Days on Days.DayID = cr.DayID
join Conferences con on con.ConferenceID = Days.ConferenceID
join Company cmp on cmp.ClientID = c.ClientID
where cr.Cancelled = 1
```

4.8 Zwrot opłaty za anulowaną rezerwację na konferencje

```
CREATE VIEW ReturnConfPayment AS
select distinct c.ClientID, c.FirstName,c.LastName, com.CompanyName, cp.Amount, con.ConferenceID
    from Clients c
join Company com on c.ClientID = com.ClientID
join ConferenceReservations cr on
c.ClientID = cr.ClientID and cr.Cancelled = 0
join ConferencePayments cp on cp.ReservationID = cr.ReservationID
join Days on Days.ConferenceID = cr.DayID
```

```
join Conferences con on Days.ConferenceID = con.ConferenceID
where con.Cancelled = 1 and cp.Amount is not null
```

4.9 Klienci, którzy anulowali rezerwację na warsztat

```
CREATE VIEW CancelledWorkshopsfReservations as
select w.WorkshopName, Days.Date, c.FirstName,c.LastName, cmp.CompanyName from Clients c join
ConferenceReservations cr
on c.ClientID = cr.ClientID
join Days on Days.DayID = cr.DayID
join Workshops w on w.DayID = Days.DayID
join WorkshopReservation wr on wr.WorkshopID = w.WorkshopID
join Company cmp on cmp.ClientID = c.ClientID
where cr.Cancelled = 1
```

4.10 Informacja jaki procent uczestników konferencji to studenci

```
CREATE VIEW StudentsConfPercentage as
select ConferenceName, (StudentsQuantity/(StudentsQuantity+Quantity))*100 as studentsPercentage
    from Conferences
join Days on Days.ConferenceID = Conferences.ConferenceID
join ConferenceReservations cr on Days.DayID = cr.DayID
```

4.11 Informacja jaki procent uczestników warsztatu to studenci

```
CREATE VIEW StudentsWorkshopPercentage as
select WorkshopName, (StudentsQuantity/(StudentsQuantity+Quantity))*100 as studentsPercentage from
    Workshops w
join WorkshopReservation wr on w.WorkshopID = wr.WorkshopID
```

4.12 Lista klientów którzy nie dokonali zapłaty

```
CREATE VIEW ClientsWithNoPayment as
select c.LastName, c.FirstName, cmp.CompanyName, cr.AmountToPay from Clients c
join ConferenceReservations cr on cr.ClientID=c.ClientID
join Company cmp on cmp.ClientID=c.ClientID
where AmountToPay > 0
```

4.13 Progi cenowe konferencji

```
CREATE VIEW ConfPriceValues as
select ConferenceName, DateNewPrice, Price from Conferences c
join ConferencePrice cp on c.ConferenceID = cp.ConferenceID
```

4.14 Dane o kliencie

```
CREATE VIEW ClientData as
select c.*, ISNULL(cmp.CompanyName, 'Individual') as CompanyName from Clients c
join Company cmp on cmp.ClientID=c.ClientID
```

4.15 Ilość dokonanych rezerwacji przez klienta

```
CREATE VIEW ClientResCount as
select c.FirstName, c.LastName, count (*) as totalReservationsNumber
from Clients c
join ConferenceReservations cr on cr.ClientID = c.ClientID
group by c.FirstName, c.LastName
```

4.16 Pokazuje klientów, którzy dokonali nadpłaty

```
CREATE VIEW ExcessPayment AS
select c.ClientID from Clients c
join ConferenceReservations cr on cr.ClientID = c.ClientID
join ConferencePayments cp on cp.ReservationID = cr.ReservationID
where cp.Amount > cr.AmountToPay
```

5 Procedury

5.1 Dodawanie klienta

```
CREATE PROCEDURE InsertClient
   @FirstName varchar(80),
   @LastName varchar(80),
   @Phone varchar(80),
   @Email varchar(80),
   @Address varchar(10),
   @Street varchar(80),
   @City varchar(80),
   @Country varchar(80),
   @CompanyName varchar(80),
   @NIP int
AS
BEGIN
   SET NOCOUNT ON;
   BEGIN TRY
      INSERT INTO Clients
      (
      FirstName,
      LastName,
      Phone,
      Email,
      Address,
      Street,
```

```
      City,
      Country
      )
      VALUES
      (
      @FirstName,
      @LastName,
      @Phone,
      @Email,
      @Address,
      @Street,
      @City,
      @Country
      )
      if @CompanyName is not null
      INSERT INTO Company
      (
      ClientID,
      CompanyName,
      NIP
      )
      VALUES
      (
      (SELECT ClientID from Clients WHERE Email like @Email),
      @CompanyName,
      @NIP
      )
   END TRY
   BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048)
         = 'Cannot add client. Error message : '
         + ERROR_MESSAGE () ;
      THROW 52000 , @errorMsg ,1;
   END CATCH
END
GO
```

5.2 Dodawanie konferencji

```
CREATE PROCEDURE InsertConference
   @ConferenceName varchar(150),
   @DateBegin date,
   @DateEnd date,
   @Address varchar(10),
   @Street varchar(80),
   @City varchar(80),
   @Country varchar(80),
   @Price money
   AS
   BEGIN
   SET NOCOUNT ON
      BEGIN TRY
      INSERT INTO Conferences
      (
```

```
    ConferenceName,
    DateBegin,
    DateEnd,
    Address,
    Street,
    City,
    Country
    )
    VALUES
    (
    @ConferenceName,
    @DateBegin,
    @DateEnd,
    @Address,
    @Street,
    @City,
    @Country
    )
    DECLARE @ConferenceID int=(SELECT ConferenceID from Conferences where ConferenceName LIKE
        @ConferenceName AND DateBegin LIKE @DateBegin)
    DECLARE @BeginDate date =@DateBegin
    WHILE (@BeginDate>=DATEADD(month,-1,@DateBegin))
      BEGIN
        EXEC InsertNewThresholdDate @ConferenceID,@BeginDate,@Price
        SET @Price=@Price*1.2
        SET @BeginDate=DATEADD(week,-1,@BeginDate)
      END
    SET @BeginDate =@DateBegin
    WHILE (@BeginDate<=@DateEnd)
      BEGIN
        INSERT INTO Days
        (ConferenceID,Date)
        VALUES
        (
        @ConferenceID,
        @BeginDate
        )
        set @BeginDate=DATEADD(day,1,@BeginDate)
      END
    END TRY
    BEGIN CATCH
    DECLARE @errorMsg nvarchar (2048)
    = 'Cannot add conference . Error message : '
    + ERROR_MESSAGE () ;
     ; THROW 52000 , @errorMsg ,1
    END CATCH
  END

GO
```

5.3 Dodawanie rezerwacji do konferencji

```
CREATE PROCEDURE InsertReservation
  @DayID int,
```

```
   @ClientID int,
   @Quantity int,
   @StudentsQuantity int
   AS
   BEGIN
   SET NOCOUNT ON
      BEGIN TRY
      DECLARE @BookingTime date= DATEADD(WEEK,-2,(SELECT Date FROM Days WHERE DayID LIKE @DayID))
      INSERT INTO ConferenceReservations
      (
         DayID,
         ClientID,
         BookingTime,
         Quantity,
         StudentsQuantity,
         AmountToPay
      )
      VALUES
      (
         @DayID,
         @ClientID,
         @BookingTime,
         @Quantity,
         @StudentsQuantity,
         (@Quantity+@StudentsQuantity/2)*(SELECT t.Price FROM (SELECT TOP 1 price FROM
             ConferencePrice
JOIN Conferences ON Conferences.ConferenceID=ConferencePrice.ConferenceID
JOIN Days ON Conferences.ConferenceID=Days.ConferenceID
WHERE Days.DayID=@DayID AND DateNewPrice>@BookingTime
ORDER BY DateNewPrice ASC) t)
      )

      END TRY
      BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048)
      = 'Cannot add conference reservation . Error message : '
      + ERROR_MESSAGE () ;
       ; THROW 52000 , @errorMsg ,1
      END CATCH
   END
GO
```

5.4 Dodawanie warsztatu do konferencji

```
CREATE PROCEDURE InsertWorkshop
   @WorkshopName varchar(150),
   @DayID int,
   @TimeBegin time,
   @TimeEnd time,
   @Size int,
   @Price int
   AS
   BEGIN
   SET NOCOUNT ON
```

```
      BEGIN TRY
      IF NOT EXISTS
      (
         SELECT * FROM Days
         WHERE Days.DayID = @DayID
      )
      BEGIN
      ; THROW 52000 , 'That is not a conference day' ,1
      END
      INSERT INTO Workshops
      (
      WorkshopName,
      DayID,
      TimeBegin,
      TimeEnd,
      Size,
      Price
      )
      VALUES
      (
      @WorkshopName,
      @DayID,
      @TimeBegin,
      @TimeEnd,
      @Size,
      @Price
      )
      END TRY
      BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048)
      = 'Cannot add workshop . Error message : '
      + ERROR_MESSAGE () ;
       ; THROW 52000 , @errorMsg ,1
      END CATCH
   END
GO
```

5.5 Dodawanie rezerwacji do warsztatu

```
CREATE PROCEDURE InsertWorkshopReservation
   @DayID int,
   @ClientID int,
   @WhichWorkshop int,
   @Quantity int,
   @StudentsQuantity int
   AS
   BEGIN
   SET NOCOUNT ON
      BEGIN TRY
      DECLARE @ReservationID int=(
         select ReservationID from ConferenceReservations WHERE ClientID=@ClientID AND DayID=@DayID
         )
      DECLARE @WorkshopID int=(
```

```
      select t.WorkshopID from (select Workshops.WorkshopID,row_number() over(order by
          Days.DayID) as 'row'
            FROM  Workshops JOIN Days ON Days.DayID=Workshops.DayID) t
      where row=@WhichWorkshop
      )
      IF @WhichWorkshop IS NOT NULL
    INSERT INTO WorkshopReservation
    (
      ReservationID,
      WorkshopID,
      Quantity,
      StudentsQuantity
    )
    VALUES
    (
      @ReservationID,
      @WorkshopID,
      @Quantity,
      @StudentsQuantity
    )
    END TRY
    BEGIN CATCH
    DECLARE @errorMsg nvarchar (2048)
    = 'Cannot add workshop reservation . Error message : '
    + ERROR_MESSAGE () ;
     ; THROW 52000 , @errorMsg ,1
    END CATCH
  END
GO
```

5.6 Dodawanie uczestnika

```
CREATE PROCEDURE InsertParticipant
  @FirstName varchar(80),
  @LastName varchar(80)
AS
BEGIN
  SET NOCOUNT ON;
  BEGIN TRY
    INSERT INTO Participants
    (
    FirstName,
    LastName
    )
    VALUES
    (
    @FirstName,
    @LastName
    )
  END TRY
  BEGIN CATCH
    DECLARE @errorMsg nvarchar (2048)
      = 'Cannot add client. Error message : '
```

```
        + ERROR_MESSAGE () ;
      THROW 52000 , @errorMsg ,1;
   END CATCH
END

GO
```

5.7 Dodawanie nowego progu cenowego

```
CREATE PROCEDURE InsertNewThresholdDate
   @ConferenceID int,
   @DateNewPrice date,
   @Price money
AS
BEGIN
   SET NOCOUNT ON;
   BEGIN TRY
      INSERT INTO ConferencePrice
      (
   ConferenceID,
   DateNewPrice,
   Price
      )
      VALUES
      (
   @ConferenceID,
   @DateNewPrice,
   @Price
      )
   END TRY
   BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048)
         = 'Cannot add new price. Error message : '
         + ERROR_MESSAGE () ;
      THROW 52000 , @errorMsg ,1;
   END CATCH
END

GO
```

5.8 Przypisanie nowego okresu ważności karty studenckiej

```
CREATE PROCEDURE AddStudentValidRange
   @ParticipantID int,
   @IDCardNumber int,
   @ValidFrom date,
   @ValidTo date
AS
BEGIN
   SET NOCOUNT ON;
```

```
   BEGIN TRY
      INSERT INTO Students
      (
   ParticipantID,
   IDCardNumber,
   ValidFrom,
   ValidTo
      )
      VALUES
      (
   @ParticipantID,
   @IDCardNumber,
   @ValidFrom,
   @ValidTo
      )
   END TRY
   BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048)
         = 'Cannot add student. Error message : '
         + ERROR_MESSAGE () ;
      THROW 52000 , @errorMsg ,1;
   END CATCH
END

GO
```

5.9 Przypisanie nowej opłaty

```
CREATE PROCEDURE AddPayment
CREATE PROCEDURE AddPayment
   @ReservationID int,
   @Amount money,
   @WhichPayment int
   AS
   BEGIN
      BEGIN TRY
         Declare @Date date=(SELECT DateAdd(Day,5*@WhichPayment,(Select BookingTime from
             ConferenceReservations WHERE ReservationID=@ReservationID)))
         INSERT INTO ConferencePayments
         (
         ReservationID,
         Amount,
         Date
         )
         VALUES
         (
         @ReservationID,
         @Amount,
         @Date
         )
      END TRY
      BEGIN CATCH
         PRINT 'Cannot add payment.';
         THROW 51000, 'ERROR', 1;
```

```
    END CATCH
  END
GO
```

5.10 Przypisanie uczestnika do rezerwacji

```
CREATE PROCEDURE AddParticipantToReservation
  @ReservationID int,
  @ParticipantID int
  AS
  BEGIN
    BEGIN TRY
      INSERT INTO ConferenceReservationParticipants
      (
      ReservationID,
      ParticipantID
      )
      VALUES
      (
      @ReservationID,
      @ParticipantID
      )
    END TRY
    BEGIN CATCH
      PRINT 'Cannot add participant.';
      THROW 51000, 'ERROR', 1;
    END CATCH
  END


GO
```

5.11 Przypisanie uczestnika do warsztatu

```
CREATE PROCEDURE AddParticipantToWorkshop
  @ReservationID int,
  @WhichWorkshop int,
  @ParticipantID int
  AS
  BEGIN
    BEGIN TRY
      DECLARE @WorkshopReservationID int=(
      select t.WorkshopReservationID from (select
          WorkshopReservation.WorkshopReservationID,row_number() over(order by DayID) as 'row'
            FROM WorkshopReservation JOIN Workshops on
                WorkshopReservation.WorkshopID=Workshops.WorkshopID WHERE
                ReservationID=@ReservationID) t
      where row=@WhichWorkshop
      )
      IF @WorkshopReservationID IS NOT NULL
      BEGIN
      INSERT INTO WorkshopReservationParticipants
```

```
        (
        WorkshopReservationID,
        ParticipantID
        )
        VALUES
        (
        @WorkshopReservationID,
        @ParticipantID
        )
        END
    END TRY
    BEGIN CATCH
        THROW 51000, 'Cannot add participant to workshop.', 1;
    END CATCH
  END
GO
```

5.12 Anulowanie konferencji

```
CREATE PROCEDURE InvalidateConference
  @ConferenceID int
  AS
  BEGIN
    BEGIN TRY
      UPDATE Conferences
      SET Cancelled=1
      WHERE ConferenceID LIKE @ConferenceID
    END TRY
    BEGIN CATCH
      THROW 51000, 'Cannot invalidate conference.', 1;
    END CATCH
  END
GO
```

5.13 Anulowanie rezerwacji konferencji

```
CREATE PROCEDURE InvalidateConferenceReservation
  @ReservationID int
  AS
  BEGIN
    BEGIN TRY
      UPDATE ConferenceReservations
      SET Cancelled=1
      WHERE ReservationID LIKE @ReservationID
    END TRY
    BEGIN CATCH
      THROW 51000, 'Cannot invalidate conference reservation.', 1;
    END CATCH
  END
GO
```

5.14 Anulowanie warsztatu

```
CREATE PROCEDURE InvalidateWorkshop
   @WorkshopID int
   AS
   BEGIN
      BEGIN TRY
         UPDATE Workshops
         SET Cancelled=1
         WHERE WorkshopID LIKE @WorkshopID
      END TRY
      BEGIN CATCH
         THROW 51000, 'Cannot invalidate workshop.', 1;
      END CATCH
   END
GO
```

5.15 Anulowanie rezerwacji warsztatu

```
CREATE PROCEDURE InvalidateWorkshopReservation
   @WorkshopReservationID int
   AS
   BEGIN
      BEGIN TRY
         UPDATE WorkshopReservation
         SET Cancelled=1
         WHERE WorkshopReservationID LIKE @WorkshopReservationID
      END TRY
      BEGIN CATCH
         THROW 51000, 'Cannot invalidate workshop reservation.', 1;
      END CATCH
   END
GO
```

5.16 Wyświetlenie danych o konkretnej konferencji

```
CREATE PROCEDURE ViewConference
   @ConferenceID int
   AS
   BEGIN
   SET NOCOUNT ON
      BEGIN TRY
      SELECT * FROM ConferenceReservations
      JOIN Days ON ConferenceReservations.DayID=Days.DayID
      JOIN Conferences ON Conferences.ConferenceID=Days.ConferenceID
      WHERE Conferences.ConferenceID=@ConferenceID ORDER BY Days.DayID ASC

      END TRY
      BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048)
      = 'Cannot select conference. Error message : '
```

```
      + ERROR_MESSAGE () ;
       ; THROW 52000 , @errorMsg ,1
      END CATCH
   END
GO
```

5.17 Wyświetlenie opłat dla konkretnej rezerwacji

```
CREATE PROCEDURE ViewPaymentsForReservation
   @ConferenceID int
   AS
   BEGIN
   SET NOCOUNT ON
      BEGIN TRY
      SELECT Amount FROM ConferenceReservations
      JOIN ConferencesPayments ON ConferenceReservations.DayID=Days.DayID
      JOIN Conferences ON Conferences.ConferenceID=Days.ConferenceID
      WHERE Conferences.ConferenceID=@ConferenceID ORDER BY Days.DayID ASC

      END TRY
      BEGIN CATCH
      DECLARE @errorMsg nvarchar (2048)
      = 'Cannot select conference. Error message : '
      + ERROR_MESSAGE () ;
       ; THROW 52000 , @errorMsg ,1
      END CATCH
   END

GO
```

6 Triggery

6.1 Trigger sprawdzający przepełnienie konferencji

```
CREATE TRIGGER CheckOverflowPeopleConference ON ConferenceReservations FOR INSERT AS
BEGIN
   DECLARE @DayID AS int
   SET @DayID = (SELECT DayID FROM inserted)

   DECLARE @ParticipantsCount AS int
   SET @ParticipantsCount = (SELECT COUNT(*) FROM ConferenceReservationParticipants
   JOIN Participants on Participants.ParticipantID =
       ConferenceReservationParticipants.ParticipantID
   JOIN Students on Participants.ParticipantID = Students.ParticipantID
   JOIN ConferenceReservations on ConferenceReservations.ReservationID =
       ConferenceReservationParticipants.ReservationID
   JOIN Days on Days.DayID = ConferenceReservations.DayID
   WHERE ((Date < ValidFrom) or (Date > ValidTo)) and ConferenceReservations.DayID = @DayID)

   DECLARE @StudentsCount AS int
   SET @StudentsCount = (SELECT COUNT(*) FROM ConferenceReservationParticipants
```

```
   JOIN Participants on Participants.ParticipantID =
       ConferenceReservationParticipants.ParticipantID
   JOIN Students on Participants.ParticipantID = Students.ParticipantID
   JOIN ConferenceReservations on ConferenceReservations.ReservationID =
       ConferenceReservationParticipants.ReservationID
   JOIN Days on Days.DayID = ConferenceReservations.DayID
   WHERE ((Date > ValidFrom) and (Date < ValidTo)) and ConferenceReservations.DayID = @DayID)

   DECLARE @ParticipantsNo AS int
   SET @ParticipantsNo = (SELECT sum(Quantity) FROM ConferenceReservations WHERE DayID =@DayID)

   DECLARE @StudentsNo AS int
   SET @StudentsNo = (SELECT sum(StudentsQuantity) FROM ConferenceReservations WHERE DayID =@DayID)

IF ((@ParticipantsNo < @ParticipantsCount) or (@StudentsNo < @StudentsCount))
    BEGIN
    RAISERROR('ConferenceOverflow', -1, -1);
    ROLLBACK TRANSACTION
    END
END
```

6.2 Trigger sprawdzający przepełnienie warsztatu

```
CREATE TRIGGER [dbo].[CheckOverflowPeopleWorkshop] ON [dbo].[WorkshopReservation] FOR INSERT AS
BEGIN
   DECLARE @WorkshopID AS int
   SET @WorkshopID = (SELECT WorkshopID FROM inserted)

   DECLARE @Total AS int
   SET @Total = (SELECT Size from Workshops WHERE WorkshopID = @WorkshopID)

   DECLARE @ParticipantsNo AS int
   SET @ParticipantsNo = (SELECT sum(Quantity) FROM WorkshopReservation WHERE WorkshopID
       =@WorkshopID)

   DECLARE @StudentsNo AS int
   SET @StudentsNo = (SELECT sum(StudentsQuantity) FROM WorkshopReservation WHERE WorkshopID
       =@WorkshopID)

IF ((@ParticipantsNo + @StudentsNo > @Total))
    BEGIN
    RAISERROR('WorkshopOverflow', -1, -1);
    ROLLBACK TRANSACTION
    END
END
```

6.3 Trigger ustawiający wartość anulowania rezerwacji na True w przypadku, gdy takiego anulowania dokonano

```
CREATE TRIGGER CancelConferenceReservation ON ConferenceReservations FOR UPDATE AS
BEGIN
IF UPDATE(Cancelled)
```

```
   BEGIN
   IF (SELECT Cancelled FROM inserted) = 1
      BEGIN
      DECLARE @ReservationID AS int
      SET @ReservationID = (SELECT ReservationID FROM inserted)
      UPDATE ConferenceReservations SET Cancelled = 1 WHERE ReservationID = @ReservationID
      END
   END
END
```

6.4 Analogiczny do poprzedniego trigger - anulujący rezerwacje na warsztat, jeśli anulowano rezerwację konferencję, w ramach której warsztat się odbywa

```
CREATE TRIGGER CancelWorkshop ON ConferenceReservations FOR UPDATE AS
BEGIN
IF UPDATE(Cancelled)
   BEGIN
   IF (SELECT Cancelled FROM inserted) = 1
      BEGIN
      DECLARE @ReservationID AS int
      SET @ReservationID = (SELECT ReservationID FROM inserted where Cancelled = 1)
      UPDATE WorkshopReservation SET Cancelled = 1 WHERE ReservationID = @ReservationID
      END
   END
END
```

6.5 Trigger sprawdzający czy ta sama osoba nie została uwzględniona w rezerwacji warsztatu więcej niż jeden raz

```
CREATE TRIGGER participantExistsInWorkshop
  ON WorkshopReservationParticipants
  AFTER INSERT, UPDATE
  AS
BEGIN
  SET NOCOUNT ON
  IF (SELECT COUNT(*)
      FROM WorkshopReservationParticipants wrp
      WHERE wrp.WorkshopReservationID = (SELECT WorkshopReservationID FROM inserted)
      and wrp.ParticipantID = (SELECT ParticipantID FROM inserted)) > 1
    BEGIN
      ; THROW 70100, 'Participants has already registered to this workshop', 4
    END
END
GO
```

6.6 Analogiczny trigger sprawdzający czy ta sama osoba nie została uwzględniona w rezerwacji konferencji więcej niż jeden raz

```
CREATE TRIGGER participantExistsInConference
  ON ConferenceReservationParticipants
  AFTER INSERT, UPDATE
  AS
BEGIN
  SET NOCOUNT ON
  IF (SELECT COUNT(*)
      FROM ConferenceReservationParticipants crp
      WHERE crp.ReservationID = (SELECT ReservationID FROM inserted)
      and crp.ParticipantID = (SELECT ParticipantID FROM inserted)) > 1
    BEGIN
      ; THROW 70100, 'Participant has already registered to this conference', 4
    END
END
```

6.7 Trigger aktualizujący cenę konferencji przy dodaniu nowego warsztatu

```
CREATE TRIGGER updatePriceAfterWorkshop
  ON WorkshopReservation
  AFTER INSERT, UPDATE
  AS
BEGIN
  SET NOCOUNT ON
  DECLARE @inserted int=(SELECT WorkshopReservationID from inserted)
  UPDATE ConferenceReservations
  SET ConferenceReservations.AmountToPay+=(
  SELECT (wr.Quantity+wr.StudentsQuantity/2)*w.Price FROM WorkshopReservation wr
  JOIN Workshops w on w.WorkshopID=wr.WorkshopID
  WHERE wr.WorkshopReservationID=@inserted)

    BEGIN
      ; THROW 70100, 'Cannot update price', 4
    END
END
```

7 Indeksy

Indeksy nieklastrowe zostały utworzone dla wszystkich kluczy obcych we wszystkich tabelach, oraz dla kolumn zawierających daty. Rozwiązanie takie jest optymalne, gdyż system nie przewiduje częstego usuwania wpisów z bazy danych (zamiast tego zastosowaliśmy flagę Cancelled), a zakładamy częste wyszukiwanie dla kluczy obcych oraz dla dat. Ponadto indeksy nieklastrowe przechowują dane w logicznym porządku i nie sortują wierszy (unikamy narzutu obliczeniowego). Pozwala to na bardzo szybkie uzyskanie danych z tabeli.

7.1 FK ConferenceID w tabeli ConferencePrice

```
CREATE NONCLUSTERED INDEX ConfPrice_ConferenceID ON ConferencePrice
(
[ConferenceID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.2 FK ConferenceID w tabeli Days

```
CREATE NONCLUSTERED INDEX Days_ConferenceID ON Days
(
[ConferenceID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.3 FK DayID w tabeli Workshops

```
CREATE NONCLUSTERED INDEX Workshops_DayID ON Workshops
(
[DayID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.4 FK ClientID w tabeli Company

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

7.5 FK ReservationID w tabeli ConferencePayments

```
CREATE NONCLUSTERED INDEX ConferencePayments_ReservationID ON ConferencePayments
(
[ReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.6 FK DayID w tabeli ConferenceReservations

```
CREATE NONCLUSTERED INDEX ConferenceReservations_DayID ON ConferenceReservations
(
[DayID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.7 FK ClientID w tabeli ConferenceReservations

```
CREATE NONCLUSTERED INDEX ConferenceReservations_ClientID ON ConferenceReservations
(
[ClientID] ASC
```

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

7.8 FK ParticipantID w tabeli Students

```
CREATE NONCLUSTERED INDEX Students_ParticipantID ON Students
(
[ParticipantID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.9 FK ReservationID w tabeli ConferenceReservationParticipants

```
CREATE NONCLUSTERED INDEX ConferenceReservationsParticipants_ReservationID ON
    ConferenceReservationParticipants
(
[ReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.10 FK ParticipantID w tabeli ConferenceReservationParticipants

```
CREATE NONCLUSTERED INDEX ConferenceReservationsParticipants_ParticipantID ON
    ConferenceReservationParticipants
(
[ParticipantID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.11 FK ParticipantID w tabeli WorkshopReservationParticipants

```
CREATE NONCLUSTERED INDEX WorkshopReservationParticipants_ParticipantID ON
    WorkshopReservationParticipants
(
[ParticipantID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.12 FK WorkshopReservationID w tabeli WorkshopReservationParticipants

```
CREATE NONCLUSTERED INDEX WorkshopReservationParticipants_WorkshopReservationID ON
    WorkshopReservationParticipants
(
[WorkshopReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.13 FK WorkshopID w tabeli WorkshopReservation

```
CREATE NONCLUSTERED INDEX WorkshopReservation_WorkshopID ON WorkshopReservation
(
[WorkshopID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.14 FK ReservationID w tabeli WorkshopReservation

```
CREATE NONCLUSTERED INDEX WorkshopReservation_ReservationID ON WorkshopReservation
(
[ReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.15 TimeBegin w tabeli Workshops

```
CREATE NONCLUSTERED INDEX Workshops_TimeBegin ON Workshops
(
[TimeBegin] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.16 TimeEnd w tabeli Workshops

```
CREATE NONCLUSTERED INDEX Workshops_TimeEnd ON Workshops
(
[TimeEnd] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.17 Date w tabeli Days

```
CREATE NONCLUSTERED INDEX Days_Date ON Days
(
Date ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

7.18 DateBegin w tabeli Conferences

```
CREATE NONCLUSTERED INDEX Conference_DateBegin ON Conferences
(
[DateBegin] ASC
```

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

7.19 DateEnd w tabeli Conferences

```
CREATE NONCLUSTERED INDEX Conference_DateEnd ON Conferences
(
[DateEnd] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8 Role

Proponujemy zdefiniowac następujące role w systemie:
- Administrator systemu - nieograniczony dostęp do bazy danych, możliwość modyfikacji widoków, triggerów, pełna kontrola nad systemem.
- Pracownik firmy zajmującej się organizacją konferencji - dostęp do widoków, możliwość modyfikowania danych w tabeli.
- Klient - dostęp do funkcji powiązanych z tworzeniem nowych rezerwacj, anulowaniem i usuwaniem,a także do widoków związanych z wolnymi miejscami na konferencje

9 Generator

9.1 Clients

```
let generateClients = function(number) {
    for (var i = 0; i < number; i++) {
        var FirstName = generujImie()
        var LastName = generujNazwisko()
        var Email = FirstName + LastName + "@gmail.com"
        var s = Email.toLowerCase();
        s = s.replace("ę", "e");
        s = s.replace("ó", "o");
        s = s.replace("ą", "a");
        s = s.replace("ś", "s");
        s = s.replace("ł", "l");
        s = s.replace("ż", "z");
        s = s.replace("ź", "z");
        s = s.replace("ć", "c");
        s = s.replace("ń", "n");
        let person = "exec InsertClient '" +
            FirstName + "','" +
            LastName + "','" +
            parseInt(Math.random() * 899999999 + 100000000) + "','" +
            s + "','" +
            parseInt(Math.random() * 100) + "','" +
            generujUlica() + "','" +
            generujMiasta() + "','" +
            "Polska'";
            person = person.concat(",'" +
```

```
            generujNazweFirmy() + "','" +
            parseInt(generujNIP() / 10) +
            "')";
        console.log(person);
    }
}
```

9.2 Participants

```
let generateParticipant = function(number) {
    for (var i = 1; i < number; i++)
        console.log("exec InsertParticipant '" +
            generujImie() + "','" +
            generujNazwisko() +
            "')";
}
```

9.3 Students

```
let generateStudents = function(number) {
    for (var i = 1; i < number; i++) {
        if (i % 2 == 0)
            for (var j = 0; j < Math.random() * 5; j++) {
                console.log("exec AddStudentValidRange '" +
                    i + "','" +
                    parseInt(Math.random() * 900000000 + 100000000) + "','" +
                    (2013 + parseInt(j / 2)) + "/" + (j % 2 == 0 ? 10 : 3) + "/" + 31 + "','" +
                    (2013 + parseInt(j / 2) + (j % 2 == 0 ? 1 : 0)) + "/" + (j % 2 == 0 ? 3 : 10) +
                        "/" + 31 +
                    "')";
            }
    }
}
```

9.4 Conferences

```
let generateConferences = function(number) {
    for (var year = 2013; year < 2013 + number; year++) {
        for (var month = 1; month < 13; month++) {
            let month2 = month < 10 ? "0" + month : month

            let start1 = parseInt(Math.random() * 9 + 10);
            console.log("exec InsertConference '" +
                (generujNazweFirmy()).substring(3,10) + "','" +
                year + "/" + month2 + "/" + (parseInt(start1)) + "','" +
                year + "/" + month2 + "/" + (parseInt(start1) + 3) + "','" +
                parseInt(Math.random() * 1000) + "','" +
                generujUlica() + "','" +
                generujMiasta() + "','" +
                "Polska" + "','" +
```

```
            parseInt(Math.random() * 1000) +
            "')");
        console.log("exec InsertConference '" +
            (generujNazweFirmy()).substring(3,10) + "','" +
            year + "/" + month2 + "/" + (parseInt(start1) + 7) + "','" +
            year + "/" + month2 + "/" + (parseInt(start1) + 10) + "','" +
            parseInt(Math.random() * 1000) + "','" +
            generujUlica() + "','" +
            generujMiasta() + "','" +
            "Polska" + "','" +
            parseInt(Math.random() * 1000) +
            "')");
    }
  }
}
```

9.5 Workshops

```
let generateWorkshops = function(number) {
  for (var i = 1; i < number + 1; i++) {
    for (var h = 0; h < Math.random() * 3 + 2; h++) {
      let random = parseInt(Math.random() * 22 + 1);
      console.log("exec InsertWorkshop '" +
        (generujNazweFirmy()).substring(3,10) + "','" +
        i + "','" +
        random + ":00','" +
        (random + 1) + ":00','" +
        parseInt(Math.random() * 20 + 100) + "','" +
        parseInt(Math.random() * 5 + 50) +
        "')");
    }
  }
}
```

9.6 Reservations

```
let generateReservations = function(number) {
  for (var i = 1; i <= number; i++) {
    for (var j = parseInt(Math.random() * 10) + 1; j <= Math.random() * 10 + 6; j++) {
      console.log("exec InsertReservation '" +
        i + "','" +
        j + "','" +
        parseInt(Math.random() * 20) + "','" +
        parseInt(Math.random() * 15) +
        "')");
    }
  }
}
```

9.7 Payments

```
let generatePayments = function(number) {
    for (var i = 1; i < number; i++) {
        for (var j = 1; j < Math.random() * 5 + 2; j++) {
            console.log("exec AddPayment '" +
                i + "','" +
                Math.random() * 3000 +
                "'")
        }
    }
}
```

9.8 Adding to reservations

```
let addParticipantsToAll = function(number) {
    for (var i = 1; i < number; i++) {
        for (var j = parseInt(Math.random() * 10); j < parseInt(Math.random() * 15) + 2; j++) {
            console.log("exec AddParticipantToReservation '" +
                i + "','" +
                j+
                "'")
            for(var h=1;h<3;h++){
                console.log("exec AddParticipantToWorkshop '" +
                h + "','" +
                j +
                "'")
            }
        }
    }
}
```

9.9 MainFunction

```
let startGenerating = function(years){
    generateClients(10);
    generateParticipant(20);
    generateStudents(10);
    generateConferences(years);
    generateWorkshops(years*12 * 2)
    generateReservations(years*12 * 2);
    generatePayments(years*12*2)
    addParticipantsToAll(years*12*2)
}
startGenerating(1)
```

9.10 Conferences Sample Output

```
exec InsertConference 'erryroX','2013/01/15','2013/01/18','134','Wandy Jordanówny','Brześć
    Kujawski','Polska','412'
```

```
exec InsertConference
    'aloryTo','2013/01/22','2013/01/25','894','Krechowiecka','Działoszyn','Polska','325'
exec InsertConference 'er rzut','2013/02/17','2013/02/20','757','Jana
    Husa','Jastrzębie-Zdrój','Polska','223'
exec InsertConference 'ertamal','2013/02/24','2013/02/27','901','Żabieniec','Krzywiń','Polska','31'
exec InsertConference 'l giątk','2013/03/10','2013/03/13','801','Wantule','Wschowa','Polska','276'
exec InsertConference 'ra modn','2013/03/17','2013/03/20','659','Amona','Raszków','Polska','100'
```