Dokumentacja, Krzysztof Hardek, Nazar Kordiumov

Aktorzy:

- Organizator - Firma organizująca konferencję
    - Planuje nadchodzące wydarzenia oraz wykłady i warsztaty, które będą się odbywać w trakcie konferencji
    - Ustala ceny na w/w wydarzenia
    - Pilnuje płatności klientów, chcących zorganizować konferencję
    - Wystawia identyfikatory imienne dla uczestników konferencji
    - Generuje raporty związane z informacjami dotyczących klientów oraz listy uczestników na poszczególne dni konferencji
- Firmy lub klienci indywidualni
    - Rejestrują się za pomocą systemu www. wybierając odpowiednią konferencję
    - Zapoznają się z cenami, które są zależne od daty złożenia zamówienia
    - Rezerwują odpowiednią ilość osób lub podają od razu przy rejestracji listę uczestników
- Uczestnicy konferencji
    - Uzupełniają swoje dane osobowe, w celu wystawienia identyfikatora
    - Zapisują się na warsztaty oraz gdy jest to potrzebne, dokonują płatności za nie.

Przykłady użycia:

1. Klient rejestruje się na konferencję za pomocą systemu www. Następnie musi zweryfikować swoją tożsamość. Po zarejestrowaniu się klient podaje listę uczestników konferencji. Każdy z uczestników musi zarejestrować się (uzupełniając swoje dane). Uczestnicy otrzymują identyfikator imienny na poszczególne konferencje, jeżeli są kilkudniowe.
2. Klient rejestruje się na konferencje. Nie podaje przy rejestracji listy uczestników tylko rezerwuje liczbę miejsc. Na 2 tygodnie przed rozpoczęciem konferencji jest do niego wykonywany telefon z

prośbą o uzupełnienie tych danych. Klient to robi i baza danych zostaje przygotowana do użycia. Uczestnicy otrzymują swoje identyfikatory.

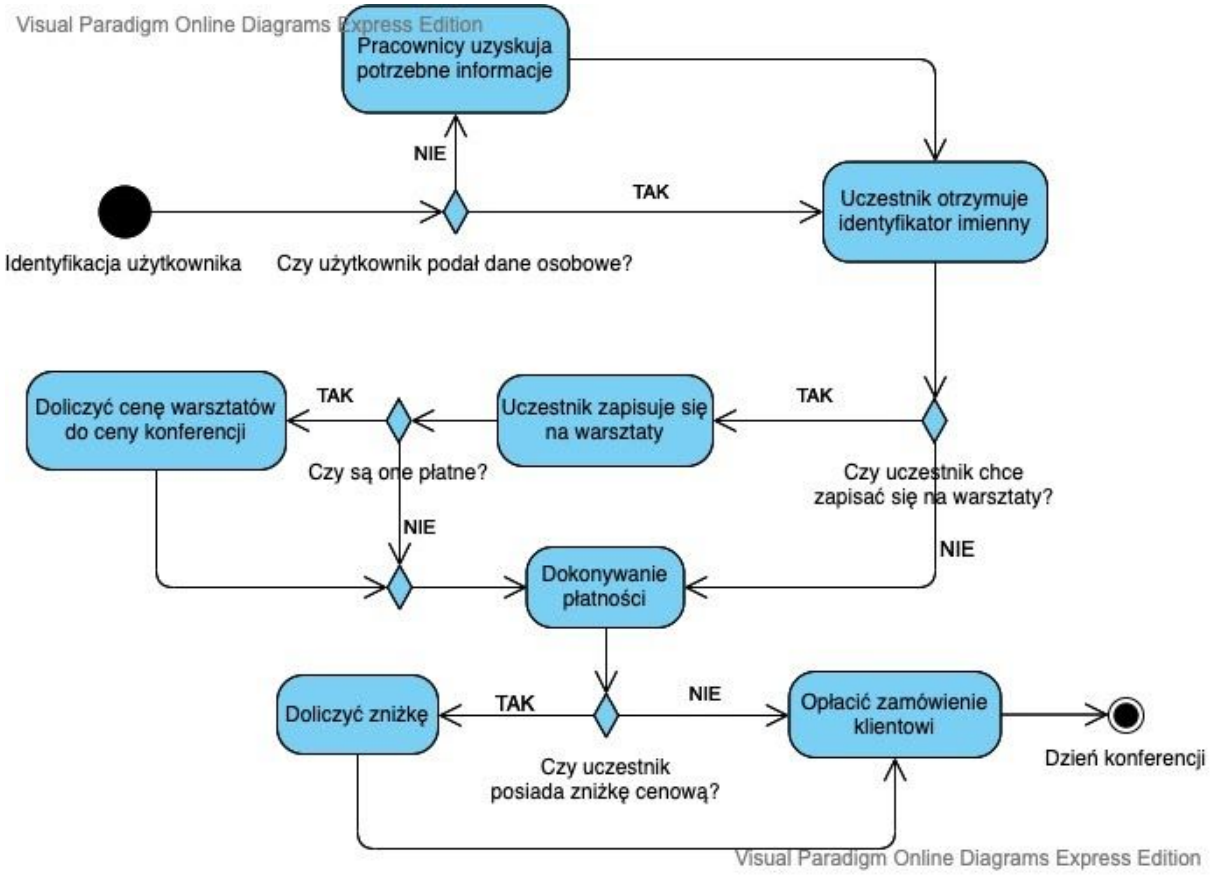
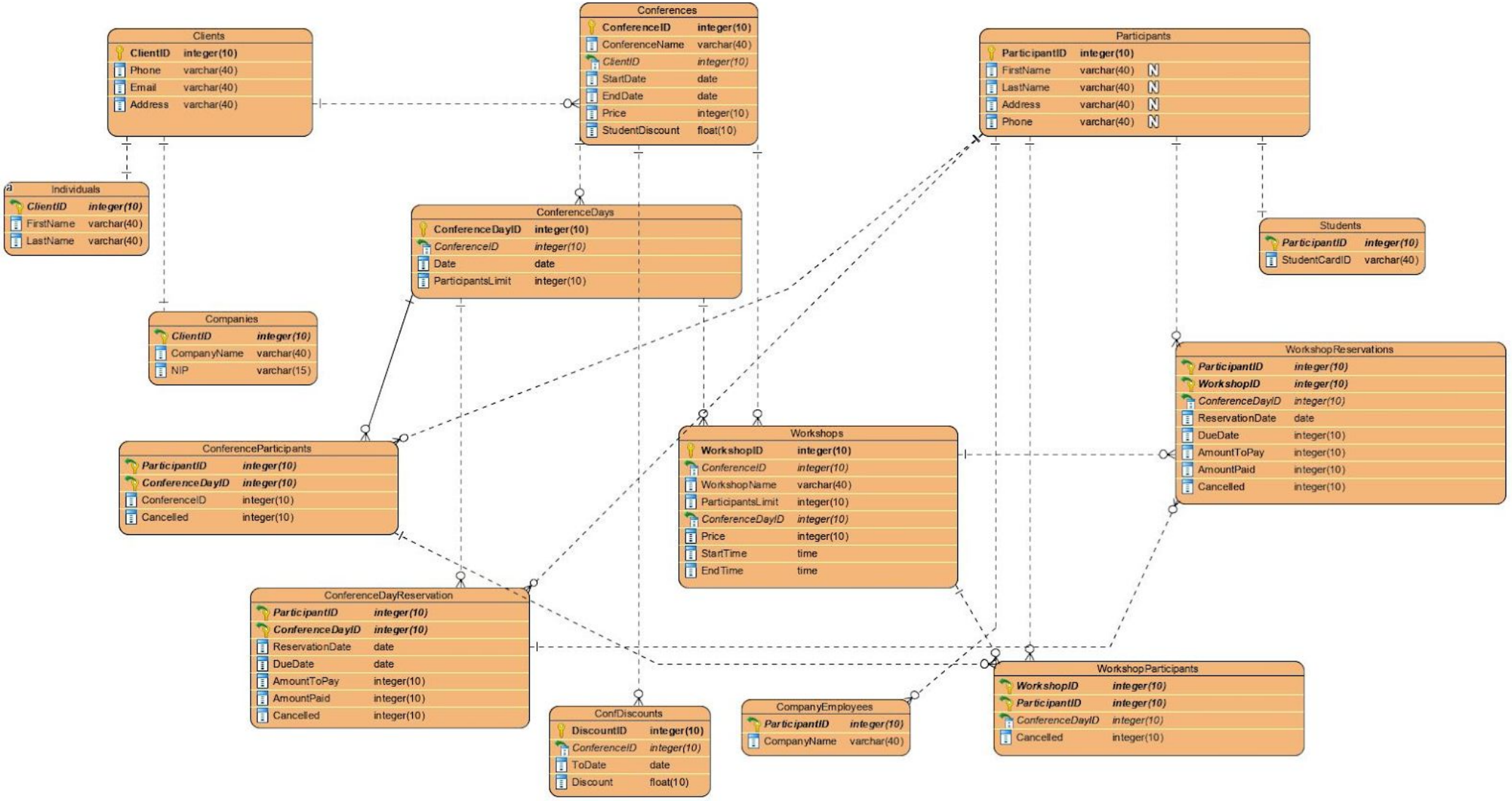


Diagram dotyczący 3,4,5 przykładu użycia

3. Uczestnik konferencji rejestruje się na konferencje oraz wybiera warsztaty, w których chce uczestniczyć. Podaje przy tym nr legitymacji studenckiej (jeżeli taką posiada), która upoważnia uczestnika do zniżek cenowych. Po tygodniu następuje weryfikacja płatności uczestnika. Jeżeli w systemie nie znajdzie się opłata uczestnika, rezerwacja zostaje anulowana.
4. Uczestnik konferencji rejestruje się na warsztaty. Potwierdzane jest, że uczestnik jest zapisany tego dnia na konferencje oraz czy warsztaty, na które się zapisał nie kolidują ze sobą. Uczestnik dokonuje płatności za uczestnictwo w warsztatach i zostaje zarejestrowany na warsztat.
5. Uczestnik konferencji rejestruje się na warsztaty. Okazuje się że uczestnik nie jest tego dnia zapisany na konferencje. rejestracja kończy się niepowodzeniem. Nie zostaje wpisany na listę.

6. Organizator klika na stronie prośbę o wygenerowanie listy uczestników konferencji oraz na każdy wykład. Strona uzyskuje potrzebne informacje z bazy danych i generuje odpowiednie listy. Klient przystępuje do weryfikacji ludzi pojawiających się na konferencji
7. Organizator wysyła prośbę o wygenerowanie informacji odnośnie płatności jego klientów. Strona uzyskuje potrzebne informacje z bazy danych i generuje listę klientów i stan ich płatności.
8. Organizator wysyła prośbę o informację o klientach korzystających z jego usług. Strona uzyskuje potrzebne informacje z bazy danych i sortuje dane względem najczęściej korzystających klientów. Następnie generuje listę bądź plik, który jest wysyłany organizatorowi.


Clients
ClientID integer(10)
Phone varchar(40)
Email varchar(40)
Address varchar(40)
Conferences
ConferenceID integer(10)
ConferenceName varchar(40)
ClientID integer(10)
StartDate date
EndDate date
Price integer(10)
StudentDiscount float(10)
Participants
ParticipantID integer(10)
FirstName varchar(40) N
LastName varchar(40) N
Address varchar(40) N
Phone varchar(40) N
Individuals
ClientID integer(10)
FirstName varchar(40)
LastName varchar(40)
ConferenceDays
ConferenceDayID integer(10)
ConferenceID integer(10)
Date date
ParticipantsLimit integer(10)
Students
ParticipantID integer(10)
StudentCardID varchar(40)
Companies
ClientID integer(10)
CompanyName varchar(40)
NIP varchar(15)
WorkshopReservations
ParticipantID integer(10)
WorkshopID integer(10)
ConferenceDayID integer(10)
ReservationDate date
DueDate integer(10)
AmountToPay integer(10)
AmountPaid integer(10)
Cancelled integer(10)
ConferenceParticipants
ParticipantID integer(10)
ConferenceDayID integer(10)
ConferenceID integer(10)
Cancelled integer(10)
Workshops
WorkshopID integer(10)
ConferenceID integer(10)
WorkshopName varchar(40)
ParticipantsLimit integer(10)
ConferenceDayID integer(10)
Price integer(10)
StartTime time
EndTime time
ConferenceDayReservation
ParticipantID integer(10)
ConferenceDayID integer(10)
ReservationDate date
DueDate date
AmountToPay integer(10)
AmountPaid integer(10)
Cancelled integer(10)
ConfDiscounts
DiscountID integer(10)
ConferenceID integer(10)
ToDate date
Discount float(10)
CompanyEmployees
ParticipantID integer(10)
CompanyName varchar(40)
WorkshopParticipants
WorkshopID integer(10)
ParticipantID integer(10)
ConferenceDayID integer(10)
Cancelled integer(10)

TABELE

Conferences

Tabela przechowuje podstawowe dane o konferencjach. Zawiera identyfikator konferencji (ConferenceID), daty rozpoczęcia i zakończenia (StartTime, EndTime), jej nazwę (ConferenceName) oraz cenę (Price).

```
CREATE TABLE [dbo].[Conferences](
   [ConferenceID] integer IDENTITY(1,1) NOT NULL,
   [ConferenceName] varchar(40) NOT NULL ,
   [ClientID] integer NOT NULL,
   [StartTime] date NOT NULL,
   [EndTime] date NOT NULL,
   [Price] integer NOT NULL,
   [StudentDiscount] float(10) NOT NULL,
   CONSTRAINT [PK_Conferences] PRIMARY KEY CLUSTERED ([ConferenceID]
ASC)
      WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
         ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].[Conferences] WITH CHECK ADD CONSTRAINT
[FK_Conferences_ClientID] FOREIGN KEY([ClientID])
   REFERENCES [dbo].[Clients] ([ClientID]);

ALTER TABLE Conferences ADD CONSTRAINT
[UniqueConferenceName_Conferences] UNIQUE NONCLUSTERED
([ConferenceName] ASC)
     WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
        ALLOW_PAGE_LOCKS = ON) ON [PRIMARY];

ALTER TABLE [dbo].[Conferences] WITH CHECK ADD CONSTRAINT
[fractionalDiscountValue] check (([StudentDiscount] >= 0 and [StudentDiscount] <=
1))
```

Clients

Przechowuje informacje o klientach korzystających z usług systemu. Zawiera ona dane niezbędne do skontaktowania się z klientem m.in. unikalny numer telefonu. Możemy także stwierdzić czy mamy do czynienia z firmą czy osobą prywatną, co daje nam możliwość grupowania uczestników ze względu na to, czy zostali zgłoszeni przez swoich pracodawców.

```
CREATE TABLE [dbo].[Clients] (
  [ClientID] integer IDENTITY(1,1) NOT NULL,
  [Phone] varchar(40)  NOT NULL,
  [Email] varchar(40) NOT NULL,
  [Address] varchar(40) NOT NULL,
  CONSTRAINT [PK_Clients] PRIMARY KEY CLUSTERED ([ClientID] ASC)
    WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
      ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
  CONSTRAINT [UniqueEMail_Clients] UNIQUE NONCLUSTERED ([Email] ASC)
    WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
      ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Companies

```
CREATE TABLE [dbo].[Companies] (
  [ClientID] [integer] NOT NULL,
  [CompanyName] varchar(40) NOT NULL ,
  [NIP] varchar(15) NOT NULL,
  CONSTRAINT [PK_Companies] PRIMARY KEY CLUSTERED ([ClientID] ASC)
    WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
      ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],

  CONSTRAINT [UniqueNIP_Clients] UNIQUE NONCLUSTERED ([NIP] ASC)
    WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
      ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

```
ALTER TABLE [dbo].[Companies] WITH CHECK ADD CONSTRAINT
[FK_Companies_ClientID] FOREIGN KEY([ClientID])
  REFERENCES [dbo].[Clients] ([ClientID]);
```

Individuals

```
CREATE TABLE [dbo].[Individuals] (
    [ClientID] [integer] NOT NULL,
    [FirstName] varchar(40) NOT NULL ,
    [LastName] varchar(40) NOT NULL,
    CONSTRAINT [PK_Individuals] PRIMARY KEY CLUSTERED ([ClientID] ASC)
       WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
          ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
) ON [PRIMARY]

ALTER TABLE [dbo].[Individuals] WITH CHECK ADD CONSTRAINT
[FK_Individuals_ClientID] FOREIGN KEY([ClientID])
  REFERENCES [dbo].[Clients] ([ClientID]);
```

Conference Days

Zawiera informacje o poszczególnych dniach konferencji. Pole ConferenceID to klucz obcy łączący z tabelą Conferences w relacji 1:n, natomiast conferenceDayID określa któremu z kolei dniu konferencji odpowiada dany rekord.

```
CREATE TABLE [dbo].[ConferenceDays](
    [ConferenceDayID] [int] IDENTITY(1,1) NOT NULL,
    [ConferenceID] [int] NOT NULL,
    [ParticipantsLimit] [int] NOT NULL,
    [Date] [date] NOT NULL,
    CONSTRAINT [PK_ConferenceDays] PRIMARY KEY CLUSTERED
([ConferenceDayID] ASC)
       WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
          ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].[ConferenceDays] WITH CHECK ADD CONSTRAINT
[FK_ConferenceDays_Conferences] FOREIGN KEY([ConferenceID])
  REFERENCES [dbo].[Conferences] ([ConferenceID])
```

```
ALTER TABLE [dbo].[ConferenceDays] CHECK CONSTRAINT
[FK_ConferenceDays_Conferences]

ALTER TABLE [dbo].[ConferenceDays] WITH CHECK ADD CONSTRAINT
[CK_ParticipantsLimit] CHECK
  (([ParticipantsLimit]>(0)))

ALTER TABLE [dbo].[ConferenceDays] CHECK CONSTRAINT
[CK_ParticipantsLimit]
```

Workshops

Zawiera podstawowe dane o warsztatach. Każdy warsztat określony jest poprzez nazwę, dzień konferencji w którym się odbywa, długość trwania danego warsztatu, maksymalną liczbę uczestników.

```
CREATE TABLE [dbo].[Workshops](
    [WorkshopID] [int] IDENTITY(1,1) NOT NULL,
    [ConferenceID] [int] NOT NULL,
    [ConferenceDayID] [int] NOT NULL,
    [WorkshopName] [varchar](100) NOT NULL,
    [ParticipantsLimit] [int] NOT NULL,
    [StartTime] [datetime] NOT NULL ,
    [EndTime] [datetime] NOT NULL ,
    [Price] [money] NOT NULL,
    CONSTRAINT [PK_Workshops] PRIMARY KEY CLUSTERED ([workshopID]
ASC)
        WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS
= ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].[Workshops] WITH CHECK ADD CONSTRAINT
[FK_Workshops_ConferenceDays] FOREIGN KEY([ConferenceDayID])
  REFERENCES [dbo].[ConferenceDays] ([ConferenceDayID]);

ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT
[FK_Workshops_ConferenceDays]

ALTER TABLE [dbo].[Workshops] WITH CHECK ADD CONSTRAINT
[FK_Workshops_Conferences] FOREIGN KEY([ConferenceID])
  REFERENCES [dbo].[Conferences] ([ConferenceID]);
```

```
ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT
[FK_Workshops_Conferences]

ALTER TABLE [dbo].[Workshops] WITH CHECK ADD CONSTRAINT
[positiveLimitNumber] CHECK
  (([ParticipantsLimit]>(0)))

ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [positiveLimitNumber]

ALTER TABLE [dbo].[Workshops] WITH CHECK ADD CONSTRAINT
[notNegativePrice] CHECK (([Price]>=(0)))

ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [notNegativePrice]
```

Participants

Zawiera informację o wszystkich uczestnikach konferencji. Oprócz klucza głównego, zawiera dwa pola określające imię i nazwisko uczestnika, adres i numer telefonu.

```
CREATE TABLE [dbo].[Participants](
   [ParticipantID] [int] IDENTITY(1,1) NOT NULL,
   [FirstName] [varchar](40),
   [LastName] [varchar](40),
   [Address] [varchar](40),
   [Phone] [varchar](40),
   CONSTRAINT [PK_Participants] PRIMARY KEY CLUSTERED ([ParticipantID]
ASC)
      WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
         ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

ConferenceParticipants

Tabela zawiera listę uczestników związaną z konkretną rezerwacją dnia konferencji. Także zawiera identyfikator dnia konferencji (ConferenceDayID), który jako klucz obcy łączy z tabelą

ConferenceDays w relacji n:1, identyfikator uczestnika (ParticipantID), który jako klucz główny i obcy łączy z tabelą Participants w relacji n:1.

```
CREATE TABLE [dbo].[ConferenceParticipants](
  [ParticipantID] [int] NOT NULL,
  [ConferenceDayID] [int] NOT NULL,
  [ConferenceID] [int] NOT NULL,
  [Cancelled] [int] NOT NULL,
  CONSTRAINT [PK_ConferenceParticipants] PRIMARY KEY CLUSTERED
([ParticipantID] ASC, [ConferenceDayID] ASC)
    WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
      ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
) ON [PRIMARY]

ALTER TABLE [dbo].[ConferenceParticipants] WITH CHECK ADD CONSTRAINT
  [FK_ConferenceParticipants_Participants] FOREIGN KEY([ParticipantID])
    REFERENCES [dbo].[Participants] ([ParticipantID])

ALTER TABLE [dbo].[ConferenceParticipants] CHECK CONSTRAINT
[FK_ConferenceParticipants_Participants]

ALTER TABLE [dbo].[ConferenceParticipants] WITH CHECK ADD CONSTRAINT
  [FK_ConferenceParticipants_ConferenceDays] FOREIGN
KEY([ConferenceDayID])
    REFERENCES [dbo].[ConferenceDays] ([ConferenceDayID])

ALTER TABLE [dbo].[ConferenceParticipants] CHECK CONSTRAINT
[FK_ConferenceParticipants_ConferenceDays]
```

WorkshopParticipants

Tabela zawiera listę uczestników związaną z konkretną rezerwacją warsztatu. Zawiera identyfikator powiązanej rezerwacji warsztatu (WorkshopID) oraz identyfikator uczestnika konferencji (ParticipantID).

```
CREATE TABLE [dbo].[WorkshopParticipants](
  [WorkshopID] [int] NOT NULL,
  [ParticipantID] [int] NOT NULL,
  [ConferenceDayID] [int] NOT NULL,
  [Cancelled] [int] NOT NULL,
```

```
  CONSTRAINT [PK_WorkshopParticipants] PRIMARY KEY CLUSTERED
([ParticipantID] ASC, [WorkshopID] ASC)
        WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
            ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
) ON [PRIMARY]

ALTER TABLE [dbo].[WorkshopParticipants] WITH CHECK ADD CONSTRAINT
  [FK_WorkshopParticipants_Participants] FOREIGN KEY([ParticipantID],
[ConferenceDayID)
      REFERENCES [dbo].[Participants] ([ParticipantID], [ConferenceDayID)

ALTER TABLE [dbo].[WorkshopParticipants] CHECK CONSTRAINT
[FK_WorkshopParticipants_Participants]

ALTER TABLE [dbo].[WorkshopParticipants] WITH CHECK ADD CONSTRAINT
  [FK_WorkshopParticipants_Workshops] FOREIGN KEY([WorkshopID])
    REFERENCES [dbo].[Workshops] ([WorkshopID])

ALTER TABLE [dbo].[WorkshopParticipants] CHECK CONSTRAINT
[FK_WorkshopParticipants_Workshops]

ALTER TABLE [dbo].[WorkshopParticipants] WITH CHECK ADD CONSTRAINT
  [FK_WorkshopParticipants_ConferenceParticipants] FOREIGN
KEY([ParticipantID])
      REFERENCES [dbo].[ConferenceParticipants] ([ParticipantID])

ALTER TABLE [dbo].[WorkshopParticipants] CHECK CONSTRAINT
[FK_WorkshopParticipants_ConferenceParticipants]
```

Students

Połączona z tabelą participants, zawiera listę uczestników, którzy są (lub byli) studentami i obowiązuje ich zniżka na uczestnictwo w konferencjach. Pole StudentCardID określa unikatowy numer legitymacji studenckiej.

```
CREATE TABLE [dbo].[Students] (
  [ParticipantID] integer NOT NULL,
  [StudentCardID] varchar(40) NOT NULL,
  CONSTRAINT [PK_Students] PRIMARY KEY CLUSTERED ([ParticipantID] ASC)
    WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
```

```
        ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
  CONSTRAINT [UniqueCardID_Students] UNIQUE NONCLUSTERED
([StudentCardID] ASC)
      WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
        ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].[Students] WITH CHECK ADD CONSTRAINT
  [FK_Students_Participants] FOREIGN KEY([ParticipantID])
      REFERENCES [dbo].[Participants] ([ParticipantID])

ALTER TABLE [dbo].[Students] CHECK CONSTRAINT [FK_Students_Participants]
```

ConferenceDayReservation

Tabela przechowuje dane dotyczące rezerwacji na konferencje. identyfikator konferencji, której dotyczy rezerwacja (ConferenceID), identyfikator uczestnika składającego rezerwację (ParticipantID), datę rezerwacji (ReservationDate) oraz pola związane z płatnościami za rezerwację.

```
CREATE TABLE [dbo].[ConferenceDayReservation](
    [ParticipantID] [int] NOT NULL,
    [ConferenceDayID] [int] NOT NULL,
    [ReservationDate] [date] NOT NULL,
    [DueDate] [date] NOT NULL,
    [AmountToPay] [money] NOT NULL,
    [AmountPaid] [money] NOT NULL,
    [Cancelled] [int] NOT NULL,
    CONSTRAINT [PK_ConfDayBooking] PRIMARY KEY CLUSTERED
([ParticipantID] ASC, [ConferenceDayID] ASC)
      WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
        ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].[ConferenceDayReservation] WITH CHECK ADD
CONSTRAINT
  [FK_ConferenceDayReservation_ConferenceDays] FOREIGN
KEY([ConferenceDayID])
```

```
    REFERENCES [dbo].[ConferenceDays] ([ConferenceDayID])

ALTER TABLE [dbo].[ConferenceDayReservation] CHECK CONSTRAINT
[FK_ConferenceDayReservation_ConferenceDays]

ALTER TABLE [dbo].[ConferenceDayReservation] WITH CHECK ADD
CONSTRAINT
  [FK_ConferenceDayReservation_Participants] FOREIGN KEY([ParticipantID])
    REFERENCES [dbo].[Participants] ([ParticipantID])

ALTER TABLE [dbo].[ConferenceDayReservation] CHECK CONSTRAINT
[FK_ConferenceDayReservation_Participants]
```

WorkshopReservation

Jest odpowiednikiem tabeli ConferenceDayReservation z tą różnicą, że potwierdza rezerwację na dany warsztat. Jeden rekord reprezentuje jednego uczestnika biorącego udział w jednym warsztacie.

```
CREATE TABLE [dbo].[WorkshopReservation](
   [ParticipantID] [int] NOT NULL,
   [WorkshopID] [int] NOT NULL,
   [ConferenceDayID] [int] NOT NULL ,
   [ReservationDate] [date] NOT NULL,
   [DueDate] [date] NOT NULL,
   [AmountToPay] [money] NOT NULL,
   [AmountPaid] [money] NOT NULL,
   [Cancelled] [int] NOT NULL,
   CONSTRAINT [PK_WorkshopReservation] PRIMARY KEY CLUSTERED
([ParticipantID] ASC, [WorkshopID] ASC)
      WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
         ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].[WorkshopReservation] WITH CHECK ADD CONSTRAINT
  [FK_WorkshopReservation_ConferenceDayReservation] FOREIGN
KEY([ParticipantID], [ConferenceDayID])
    REFERENCES [dbo].[ConferenceDayReservation] ([ParticipantID],
[ConferenceDayID])

ALTER TABLE [dbo].[WorkshopReservation] CHECK CONSTRAINT
[FK_WorkshopReservation_ConferenceDayReservation]
```

```
ALTER TABLE [dbo].[ConferenceDayReservation] WITH CHECK ADD
CONSTRAINT
  [FK_WorkshopReservation_Participants] FOREIGN KEY([ParticipantID])
    REFERENCES [dbo].[Participants] ([ParticipantID])

ALTER TABLE [dbo].[ConferenceDayReservation] CHECK CONSTRAINT
[FK_WorkshopReservation_Participants]

ALTER TABLE [dbo].[WorkshopReservation] WITH CHECK ADD CONSTRAINT
  [FK_WorkshopReservation_Workshops] FOREIGN KEY([WorkshopID])
    REFERENCES [dbo].[Workshops] ([WorkshopID])

ALTER TABLE [dbo].[WorkshopReservation] CHECK CONSTRAINT
[FK_WorkshopReservation_Workshops]
```

ConfDiscounts

Tabela zawiera listę zniżek na konferencje. Każda zniżka jest przypisana do danej konferencji oraz ma datę wygaśnięcia. Dla danej konferencji przy płatności obowiązuje zniżka której data jest późniejsza lub taka sama jak data rezerwacji. Z wszystkich zniżek spełniających ten warunek wybieramy tą z najwcześniejszą datą

```
CREATE TABLE [dbo].[ConfDiscounts](
    [DiscountID] [int] IDENTITY(1,1) NOT NULL,
    [ConferenceID] [int] NOT NULL,
    [ToDate] [date] NOT NULL,
    [Discount] [float] NOT NULL,
    CONSTRAINT [PK_ConfDiscounts] PRIMARY KEY CLUSTERED ([DiscountID]
ASC)
       WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
          ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].[ConfDiscounts] WITH CHECK ADD CONSTRAINT
[FK_ConfDiscounts_Conferences] FOREIGN KEY([ConferenceID]) REFERENCES
[dbo].[Conferences] ([ConferenceID])

ALTER TABLE [dbo].[ConfDiscounts] CHECK CONSTRAINT
[FK_ConfDiscounts_Conferences]
```

```
ALTER TABLE [dbo].[ConfDiscounts] WITH CHECK ADD CONSTRAINT
[CK_Discount] CHECK
  (([Discount]>=(0) and [Discount]<=(1)))

ALTER TABLE [dbo].[ConfDiscounts] CHECK CONSTRAINT [CK_Discount]
```

CompanyEmployees

Tabela zawiera informacje o uczestnikach, którzy są pracownikami danej firmy.

```
CREATE TABLE [dbo].[CompanyEmployees] (
  [ParticipantID] integer NOT NULL,
  [CompanyName] varchar(40) NOT NULL,
  CONSTRAINT [PK_CompanyEmployees] PRIMARY KEY CLUSTERED
([ParticipantID] ASC)
     WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
         ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
) ON [PRIMARY]

ALTER TABLE [dbo].[CompanyEmployees] WITH CHECK ADD CONSTRAINT
  [FK_CompanyEmployees_Participants] FOREIGN KEY([ParticipantID])
     REFERENCES [dbo].[Participants] ([ParticipantID])

ALTER TABLE [dbo].[CompanyEmployees] CHECK CONSTRAINT
[FK_CompanyEmployees_Participants]
```

WIDOKI

UpcomingConferences

Widok wyświetlający konferencje które jeszcze się nie odbyły wraz z datą rozpoczęcia izakończenia oraz limitem miejsc i liczbą jeszcze wolnych miejsc

```
CREATE VIEW UpcomingConferences AS
SELECT Conferences.ConferenceName, Conferences.StartTime,
Conferences.EndTime,
     SUM(ConferenceDays.ParticipantsLimit) AS ParticipantsLimit,
     (select count(*) from ConferenceDays
     inner join ConferenceParticipants
```

```
        on ConferenceParticipants.ConferenceDayID =
ConferenceDays.ConferenceDayID
        where ConferenceDays.ConferenceID = Conferences.ConferenceID) +
      (select count (*) from ConferenceDayReservation
      inner join ConferenceDays on ConferenceDayReservation.ConferenceDayID =
ConferenceDays.ConferenceDayID
      inner join Conferences C on ConferenceDays.ConferenceID = C.ConferenceID
        where C.ConferenceID = ConferenceDays.ConferenceID and
            ((GETDATE() < DueDate) or (GETDATE() > DueDate and AmountPaid >
AmountToPay)))
        as 'CurrentParticipantsAndReservation'
FROM Conferences LEFT OUTER JOIN ConferenceDays ON
Conferences.ConferenceID =
                                    ConferenceDays.ConferenceID
WHERE (Conferences.EndTime >= GETDATE())
GROUP BY Conferences.ConferenceID, Conferences.ConferenceName,
Conferences.StartTime, Conferences.EndTime;
```

UpcomingWorkshops

Warsztaty, które odbędą się podczas nadchodzących konferencji. Wyświetlane są informacje o nazwie warsztatu, jego cenie, liczbie miejsc zajętych i wolnych.

```
CREATE VIEW UpcomingWorkshops AS
  select conf.ConferenceName, Workshops.WorkshopName, Workshops.Price,
Workshops.ParticipantsLimit - (
      (select count (*) from WorkshopReservation as wr
      where wr.WorkshopID = Workshops.WorkshopID and
            ((GETDATE() < DueDate) or (GETDATE() > DueDate and AmountPaid >
AmountToPay)))
      +
      (select count(*) from WorkshopParticipants
        where WorkshopParticipants.WorkshopID = Workshops.WorkshopID)
      ) as 'AvailablePlaces', Workshops.StartTime, Workshops.EndTime
  from Workshops
  inner join Conferences as conf on Workshops.ConferenceID = conf.ConferenceID
  where conf.EndTime >= GETDATE();
```

UnpaidConferenceDayReservation

Widok wyświetlający nieopłacone rezerwacje, dane osobowe uczestników oraz nazwy konferencji. Rezerwacje te przeznaczone są do anulowania, jeśli nie zostaną opłacone.

```
CREATE VIEW UnpaidConferenceDayReservation AS
  select c.ConferenceName, cd.Date, P.FirstName + ' ' + P.LastName as 'Name',
  cdr.AmountPaid, cdr.AmountToPay, cdr.AmountToPay - cdr.AmountPaid as
'DifferenceToPay'
  from ConferenceDayReservation as cdr
  inner join ConferenceDays CD on cdr.ConferenceDayID = CD.ConferenceDayID
  inner join Conferences C on CD.ConferenceID = C.ConferenceID
  inner join Participants P on cdr.ParticipantID = P.ParticipantID
  where cdr.AmountPaid < cdr.AmountToPay and getdate() > cdr.DueDate;
```

UnpaidWorkshopReservations

Widok wyświetlający nieopłacone rezerwacje, dane osobowe uczestników oraz nazwy warsztatów. Rezerwacje te przeznaczone są do anulowania, jeśli nie zostaną opłacone.

```
CREATE VIEW UnpaidWorkshopReservations AS
  SELECT C.ConferenceName, CD.Date, W.WorkshopName, P.FirstName + ' ' +
P.LastName + ' ' + P.Address as 'Name'
from WorkshopReservation
inner join Workshops W on WorkshopReservation.WorkshopID = W.WorkshopID
inner join ConferenceDays CD on W.ConferenceDayID = CD.ConferenceDayID
inner join Conferences C on CD.ConferenceID = C.ConferenceID
inner join Participants P on WorkshopReservation.ParticipantID = P.ParticipantID
where WorkshopReservation.AmountPaid < WorkshopReservation.AmountToPay
AND GETDATE() > cd.Date;
```

ParticipantsOfUpcomingConferences

Widok wyświetlający uczestników nadchodzących konferencji

```
CREATE VIEW ParticipantsOfUpcomingConferences AS
SELECT ConferenceName, CD.Date, P.FirstName + ' ' + P.LastName + ' ' +
P.Address as 'Participant'
FROM Conferences
inner join ConferenceDays CD on Conferences.ConferenceID = CD.ConferenceID
```

```
inner join ConferenceParticipants CP on CD.ConferenceDayID =
CP.ConferenceDayID
inner join Participants P on CP.ParticipantID = P.ParticipantID
where  CP.Cancelled <> 1 and CD.Date > GETDATE();
```

ParticipantsOfUpcomingWorkshops

Widok wyświetlający uczestników nadchodzących warsztatów

```
CREATE VIEW ParticipantsOfUpcomingWorkshops AS
SELECT C.ConferenceName, CD.Date, P.FirstName + ' ' + P.LastName + ' ' +
P.Address as 'Participant'
FROM Workshops W
inner join WorkshopParticipants WP on W.WorkshopID = WP.WorkshopID
inner join Participants P on WP.ParticipantID = P.ParticipantID
inner join ConferenceDays CD on W.ConferenceDayID = CD.ConferenceDayID
inner join Conferences C on CD.ConferenceID = C.ConferenceID
where  WP.Cancelled <> 1 and CD.Date > GETDATE();
```

OrganizedConferencesByClients

Widok wyświetlający klientów, ich dane osobowe bądź nazwę firmy oraz ilość zorganizowanych konferencji

```
CREATE VIEW OrganizedConferencesByClients AS
  SELECT iif(I.ClientID IS NULL, Cp.CompanyName, I.FirstName + ' ' + I.LastName)
as 'Client',
       count(C.ClientID) as 'NumberOfOrganizedConferences'
from Clients
left join Individuals I on Clients.ClientID = I.ClientID
left join Companies Cp on Clients.ClientID = Cp.ClientID
left join Conferences C on Clients.ClientID = C.ClientID
group by iif(I.ClientID IS NULL, Cp.CompanyName, I.FirstName + ' ' + I.LastName)
```

ConferenceReservedPlacesByClients

Widok wyświetlający firmy oraz ilość ich rezerwacji na poszczególne konferencje

```
CREATE VIEW ConferenceReservedPlacesByClients AS
  SELECT cp.CompanyName, C.ConferenceName, CD.Date, count(*) AS
'NumberOfReservedPlaces'
from Clients
inner join  Companies Cp on Clients.ClientID = Cp.ClientID
```

```
inner join Conferences C on Clients.ClientID = C.ClientID
inner join ConferenceDays CD on C.ConferenceID = CD.ConferenceID
inner join ConferenceDayReservation CDR on CD.ConferenceDayID =
CDR.ConferenceDayID
inner join ConferenceParticipants P on CD.ConferenceDayID = P.ConferenceDayID
inner join Participants P2 on CDR.ParticipantID = P2.ParticipantID
where P2.FirstName IS NULL and P2.LastName IS NULL and P2.Address is null
and P2.Phone IS NULL
and datediff(day, GETDATE(), C.StartTime) > 14
group by cp.CompanyName, C.ConferenceName, CD.date
```

WorkshopReservedPlacesByClients

Widok wyświetlający firmy oraz ilość ich rezerwacji na poszczególne warsztaty

```
CREATE VIEW WorkshopReservedPlacesByClients AS
 SELECT cp.CompanyName, C.ConferenceName, W.WorkshopName, count(*) AS
'NumberOfReservedPlaces'
from Clients
inner join Companies Cp on Clients.ClientID = Cp.ClientID
inner join Conferences C on Clients.ClientID = C.ClientID
inner join Workshops W on C.ConferenceID = W.ConferenceID
left join WorkshopReservation WR on W.WorkshopID = WR.WorkshopID
left join Participants P2 on WR.ParticipantID = P2.ParticipantID
where P2.FirstName IS NULL and P2.LastName IS NULL and P2.Address is null
and P2.Phone IS NULL
and datediff(day, GETDATE(), C.StartTime) > 14
group by cp.CompanyName, C.ConferenceName, W.WorkshopName;
```

ResignationsByParticipants

Widok wyświetlający ilość niewykorzystanych rezerwacji (brak płatności na czas) oraz z których uczestnik zrezygnował samodzielnie dla konferencji i warsztatów

```
create view ResignationsByParticipants AS
select iif(Participants.FirstName IS NULL,
  C3.CompanyName,
  Participants.FirstName + ' ' + Participants.LastName + ' ' + Participants.Address)
as 'Participant' ,
    count(cdr.ParticipantID) + count(cp.ParticipantID) AS
'AmountOfResignationOfConferences',
    count(wp.ParticipantID) + count(wr.ParticipantID) AS
'AmountOfResignationOfWorkshops'
```

```
from Participants
left join ConferenceDayReservation CDR on Participants.ParticipantID =
CDR.ParticipantID
left join ConferenceParticipants CP on Participants.ParticipantID = CP.ParticipantID
left join WorkshopParticipants WP on CP.ParticipantID = WP.ParticipantID and
CP.ConferenceDayID = WP.ConferenceDayID
left join WorkshopReservation WR on CDR.ParticipantID = WR.ParticipantID and
CDR.ConferenceDayID = WR.ConferenceDayID
left join ConferenceDays CD on CDR.ConferenceDayID = CD.ConferenceDayID
left join Conferences C on CD.ConferenceID = C.ConferenceID
left join Clients C2 on C.ClientID = C2.ClientID
left join Companies C3 on C2.ClientID = C3.ClientID
where cdr.Cancelled = 1 or cp.Cancelled = 1 or wp.Cancelled = 1 or wr.Cancelled =
1
group by  C3.CompanyName, Participants.FirstName, Participants.LastName,
Participants.Address;
```

FUNKCJE

getConferenceDayID

Funkcja zwraca ID dnia konferencji na podstawie ID konferencji oraz daty.

```
CREATE FUNCTION getConferenceDayID (
  @dateOfDay date,
  @conferenceIdParameter int
) RETURNS int
BEGIN
  RETURN (select ConferenceDayID
    from ConferenceDays
    inner join Conferences C on ConferenceDays.ConferenceID = C.ConferenceID
    where C.ConferenceID = @conferenceIdParameter and ConferenceDays.Date
= @dateOfDay)
END
```

isStudent

Funkcja przyjmuje ID uczestnika i zwraca 10 jeśli uczestnik jest studentem albo -10 jeśli uczestnik nie jest studentem

```
create function isStudent (@ParticipantID int)
```

```
returns int
as
begin
  declare @isStudent as int
  declare @student as int
  set @student = (
     select ParticipantID
     from Students
     where ParticipantID = @ParticipantID
  )
  if @student is null
  begin
     set @isStudent = -10
  end
  else
  begin
     set @isStudent = 10
  end

  return @isStudent
end
```

getConferenceFreeSeats

Funkcja przyjmuje ID dnia konferencji i zwraca liczbę wolnych miejsc na tą konferencje

```
create function getConferenceDayFreeSeats (@ConferenceDayID int)
returns int
as
begin
  declare @Limit as int
  declare @Participants as int
  declare @Reservations as int

  set @Limit = (
    select ParticipantsLimit
    from ConferenceDays
    where ConferenceDayID = @ConferenceDayID
  )
  set @Participants = (
    select count(ParticipantID)
    from ConferenceParticipants
```

```
    where ConferenceDayID = @ConferenceDayID
  )
  set @Reservations = (
    select count(ParticipantID)
    from ConferenceDayReservation
    where ConferenceDayID = ConferenceDayID
  )

  return @Limit - @Participants - @Reservations
end
```

getWorkshopID

Funkcja przyjmuje datę oraz nazwę warsztatu i nazwę konferencji. Zwraca ID warsztatu.

```
create function getWorkshopID (@Date date, @ConferenceName varchar(40),
@WorkshopName varchar(40))
returns int
as
begin
  declare @WorkshopID as int
  declare @ConferenceDayID as int
  declare @ConferenceID as int

  set @ConferenceID = dbo.getConferenceID (@ConferenceName)
  set @ConferenceDayID = dbo.getConferenceDayID (@Date, @ConferenceID)
  set @WorkshopID = (
    select WorkshopID
    from Workshops
    where ConferenceDayID = @ConferenceDayID and WorkshopName =
@WorkshopName
  )
  return @WorkshopID
end
```

getWorkshopPrize

Funkcja zwraca cane warsztatu na podstawie jego ID.

```
create function getWorkshopPrize (@WorkshopID int)
returns money
as
```

```
begin
  return (
    select Price
    from Workshops
    where WorkshopID = @WorkshopID
  )
end

select * from Workshops
```

getConfDiscount

Funkcja zwraca zniżkę na konkretną konferencje, która aktualnie obowiązuje.

```
create function getConfDiscount (@ConferenceID int, @Date date)
returns float
as
begin
  return (
    select top 1 Discount
    from ConfDiscounts
    where  ConferenceID = @ConferenceID and @Date <= toDate
    order by ToDate asc
  )

end
```

getStudentDiscount

Funkcja zwraca zniżkę studencką dla konkretnej konferencji.

```
create function getStudentDiscount (@ConferenceID int)
returns float
as
begin
  return (
    select StudentDiscount
    from Conferences
    where ConferenceID = @ConferenceID
  )
end
```

getConferencePrice

Funkcja zwracająca cenę konferencji na podstawie jej ID.

```
create function getConferencePrice (@ConferenceID int)
returns int
as
begin
  return (
        select Price
        from Conferences
        where ConferenceID = @ConferenceID
  )
end
```

getWorkshopFreeSeats

Funkcja zwraca ilość wolnych miejsc na danym warsztacie na podstawie jego ID.

```
create function getWorkshopFreeSeats (@WorkshopID int)
returns int
begin
  declare @Participants as int
  declare @Limit as int
  declare @Reservation as int

  set @Limit = (
     select ParticipantsLimit
     from Workshops
     where WorkshopID = @WorkshopId
  )
  set @Participants = (
     select count(ParticipantID)
     from WorkshopParticipants
     where WorkshopID = @WorkshopID
  )
  set @Reservation = (
     select count(ParticipantID)
     from WorkshopReservation
     where WorkshopID = @WorkshopID
  )

  return @Limit - @Participants - @Reservation
end
```

isConferenceParticipant

Funkcja zwraca czy dany użytkownik jest uczestnikiem konkretnego dnia konferencji

```
create function isConferenceParticipant (@ConferenceDayID int, @ParticipantID int)
returns int
as
begin
  declare @isConferenceParticipant as int
  declare @result as int

  set @result = (
    select ParticipantID
    from ConferenceParticipants
    where ParticipantID = @ParticipantID and ConferenceDayID =
@ConferenceDayID
  )
  if @result is null
  begin
    set @isConferenceParticipant = -10
  end
  else
  begin
    set @isConferenceParticipant = 10
  end

  return @isConferenceParticipant
end
```

PROCEDURY

Add_Conference

Procedura dodaje konferencje.

```
CREATE PROCEDURE Add_Conference
  @ConferenceName varchar(40),
  @ClientID int,
  @StartTime date,
  @EndTime date,
  @Prize int,
```

```
  @StudentDiscount float
  as
  begin
     set nocount on;
     insert into Conferences(ConferenceName, ClientID, StartTime, EndTime, Prize,
StudentDiscount)
     values (@ConferenceName, @ClientID, @StartTime, @EndTime, @Prize,
@StudentDiscount)
  end
go
```

Add_Conference_Day

Procedura dodaje dzień konferencji.

```
create procedure Add_Conference_Day
  @ConferenceName varchar(40),
  @Date date,
  @ParticipantLimit int
  as
  begin
     set nocount on;
     Declare @ConferenceID as int
     set @ConferenceID = (
          select ConferenceID
          from Conferences
          where ConferenceName = @ConferenceName
     )
     insert into ConferenceDays(ConferenceID, Date, ParticipantsLimit)
     values (@ConferenceID, @Date, @ParticipantLimit)
  end
go
```

Add_Workshop

Procedura dodaje Warsztat.

```
CREATE PROCEDURE Add_Workshop
@ConferenceName varchar(40),
@WorkshopName varchar(40),
@Date date,
@Prise int,
@ParticipantsLimit int,
@StartTime datetime,
@EndTime datetime
AS
BEGIN
```

```
  SET NOCOUNT ON
  Declare @ConferenceID as int
  set @ConferenceID = dbo.getConferenceID (@ConferenceName)
  DECLARE @ConferenceDayID AS int
  SET @ConferenceDayID = [dbo].getConferenceDayID (@Date, @ConferenceID)
  INSERT INTO Workshops(ConferenceID, ConferenceDayID, WorkshopName,
ParticipantsLimit, StartTime, EndTime, Price)
  VALUES (@ConferenceID, @ConferenceDayID, @WorkshopName,
@ParticipantsLimit, @StartTime, @EndTime, @Prise)
END
go
```

Add_Conference_Day_Reservation

Procedura dodaje rezerwacje na konkretny dzień konferencji. Przyjmuje ID uczestnika, nazwę konferencji, oraz datę rezerwacji.

```
create procedure Add_Conference_Day_Reservation
  @ParticipantID int,
  @ConferenceName varchar(40),
  @Date date
  as
  begin
     set nocount on;
     declare @AmountToPay as float
     declare @AmountPaid as float
     declare @DueDate as date
     declare @ConfDiscount as float
     declare @StudentDiscount as float
     declare @ConferenceDayID as int
     declare @Price as float
     declare @ConferenceID as int

     set @ConferenceID = dbo.getConferenceID (@ConferenceName)
     set @ConferenceDayID = dbo.getConferenceDayID (@Date, @ConferenceID)
     set @AmountPaid = 0;
     set @Price = dbo.getConferencePrice (@ConferenceID)
     set @DueDate = dateadd(week, 1, @Date)
     set @StudentDiscount = dbo.getStudentDiscount (@ConferenceID)
     set @ConfDiscount = dbo.getConfDiscount (@ConferenceID, @Date)

     if @ConfDiscount is null
```

```
    begin
        set @ConfDiscount = 0
    end

    if dbo.isStudent(@ParticipantID) > 0
    begin
        set @AmountToPay = @Price*(1-@StudentDiscount)*(1-@ConfDiscount)
    end
    else
    begin
        set @AmountToPay = @Price*(1-@ConfDiscount)
    end
    if dbo.getConferenceDayFreeSeats (@ConferenceDayID) > 0
    begin
        insert into ConferenceDayReservation(ParticipantID, ConferenceDayID,
ReservationDate, DueDate, AmountToPay, AmountPaid)
        values (@ParticipantID, @ConferenceDayID, @Date, @DueDate,
@AmountToPay, @AmountPaid)
    end
  end
go
```

Add_Client_Individual

Procedura dodaje klienta indywidualnego na podstawie jego danych osobowych.

```
create procedure Add_Client_Individual
  @Phone int,
  @Email varchar(40),
  @Address varchar(40),
  @FirstName varchar(40),
  @LastName varchar(40)
  as
  begin
     set nocount on;
     insert into Clients(Phone, Email, Address)
     values (@Phone, @Email, @Address)
     declare @ClientID as int
     set @ClientID = (
        select ClientID
        from Clients
        where Email = @Email
```

```
    )

    insert into Individuals(ClientID, FirstName, LastName)
    values (@ClientID, @FirstName, @LastName)
  end
go
```

Add_Client_Company

Procedura dodaje klienta firmę na podstawie jego danych osobowych.

```
create procedure Add_Client_Company
  @Phone int,
  @Email varchar(40),
  @Address varchar(40),
  @CompanyName varchar(40),
  @NIP varchar(15)
  as
  begin
    set nocount on;
    insert into Clients(Phone, Email, Address)
    values (@Phone, @Email, @Address)
    declare @ClientID as int
    set @ClientID = (
      select ClientID
      from Clients
      where Email = @Email
    )

    insert into Companies(ClientID, CompanyName, NIP)
    values (@ClientID, @CompanyName, @NIP)
  end
go
```

Add_Workshop_Reservation

Procedura dodaje rezerwacje warsztatu dla pojedynczej osoby.

```
create procedure Add_Workshop_Reservation
  @ConferenceName varchar(40),
  @Date date,
  @WorkshopName varchar(40),
  @ParticipantID int
```

```
  as
  begin
     set nocount on;
     declare @WorkshopID as int
     declare @ConferenceDayID as int
     declare @DueDate as date
     declare @AmountToPay as float
     declare @AmountPaid as float

     set @WorkshopID = dbo.getWorkshopID (@Date, @ConferenceName,
@WorkshopName)
     set @ConferenceDayID = dbo.getConferenceID (@ConferenceName)
     set @DueDate = dateadd(week, 1, @Date)
     set @AmountToPay = dbo.getWorkshopPrize(@WorkshopID)
     set @AmountPaid = 0

     if dbo.getWorkshopFreeSeats (@WorkshopID) > 0
          and dbo.isConferenceParticipant (@ParticipantID, @ConferenceDayID) > 0
          and dbo.isAlreadyParticipating (@ParticipantID, @WorkshopID) < 0
     begin
        insert into WorkshopReservation(ParticipantID, WorkshopID,
ConferenceDayID, ReservationDate, DueDate, AmountToPay, AmountPaid)
        values (@ParticipantID, @WorkshopID , @ConferenceDayID, @Date,
@DueDate, @AmountToPay, @AmountPaid)
     end
  end
go
```

Add_Conference_Day_Participant

Procedura dodająca uczestnika konferencji na konkretny dzień.

```
CREATE PROCEDURE Add_Conference_Day_Participant
  @ParticipantID int,
  @ConferenceDayDate date,
  @ConferenceName varchar(40)
  as
  begin
     set nocount on;
     Declare @ConferenceID as int
     Declare @ConferenceDayID as int
     set @ConferenceID = dbo.getConferenceID (@ConferenceName)
     set @ConferenceDayID = dbo.getConferenceDayID (@ConferenceDayDate,
@ConferenceID)
```

```
    insert into ConferenceParticipants(ParticipantID, ConferenceDayID,
ConferenceID)
    values (@ParticipantID, @ConferenceDayID, @ConferenceID)
  end
go
```

Add_Workshop_Reservation

Procedura dodaje rezerwacje warsztatu.

```
create procedure Add_Workshop_Reservation
  @ConferenceName varchar(40),
  @Date date,
  @WorkshopName varchar(40),
  @ParticipantID int
  as
  begin
    set nocount on;
    declare @WorkshopID as int
    declare @ConferenceDayID as int
    declare @DueDate as date
    declare @AmountToPay as float
    declare @AmountPaid as float
    declare @ConferenceID as int

    set @ConferenceID = dbo.getConferenceID (@ConferenceName)
    set @WorkshopID = dbo.getWorkshopID (@Date, @ConferenceName,
@WorkshopName)
    set @ConferenceDayID = dbo.getConferenceDayID (@Date, @ConferenceID)
    set @DueDate = dateadd(week, 1, @Date)
    set @AmountToPay = dbo.getWorkshopPrize(@WorkshopID)
    set @AmountPaid = 0

    if dbo.getWorkshopFreeSeats (@WorkshopID) > 0
        and dbo.isConferenceParticipant (@ParticipantID, @ConferenceDayID) > 0
        and dbo.isAlreadyParticipating (@ParticipantID, @WorkshopID) < 0
    begin
      insert into WorkshopReservation(ParticipantID, WorkshopID,
ConferenceDayID, ReservationDate, DueDate, AmountToPay, AmountPaid)
      values (@ParticipantID, @WorkshopID , @ConferenceDayID, @Date,
@DueDate, @AmountToPay, @AmountPaid)
    end
  end
go
```

Add_Workshop_Participant

Procedura dodająca uczestnika warsztatu.

```
create procedure Add_Workshop_Participant
  @ConferenceName varchar(40),
  @Date date,
  @WorkshopName varchar(40),
  @ParticipantID int
  as
  begin
     set nocount on;
     declare @WorkshopID as int
     declare @ConferenceDayID as int
     declare @ConferenceID as int


     set @WorkshopID = dbo.getWorkshopID (@Date, @ConferenceName,
@WorkshopName)
     set @ConferenceID = dbo.getConferenceID (@ConferenceName)
     set @ConferenceDayID = dbo.getConferenceDayID (@Date, @ConferenceID)

     if dbo.getWorkshopFreeSeats (@WorkshopID) > 0
          and dbo.isConferenceParticipant (@ConferenceDayID, @ParticipantID) > 0
          and dbo.isAlreadyParticipating (@ParticipantID, @WorkshopID) < 0
     begin
        insert into WorkshopParticipants(WorkshopID, ParticipantID,
ConferenceDayID)
        values (@WorkshopID, @ParticipantID, @ConferenceDayID)
     end
  end
go
```

Cancel_Workshop_Participation

Procedura usuwająca uczestnictwo w warsztacie.

```
create procedure Cancel_Workshop_Participation
  @ParticipantID int,
  @Date date,
  @ConferenceName varchar(40),
  @WorkshopName varchar(40)
  as
  begin
     set nocount on;
     declare @WorkshopID as int
```

```
    set @WorkshopID = dbo.getWorkshopID (@Date, @ConferenceName,
@WorkshopName)
    update WorkshopParticipants
    set Cancelled = 1
    where WorkshopID = @WorkshopID and ParticipantID = @ParticipantID
  end
go
```

Cancel_Workshop_Reservation

Procedura usuwające rezerwacje warsztatu.

```
create procedure Cancel_Workshop_Reservation
  @ParticipantID int,
  @Date date,
  @ConferenceName varchar(40),
  @WorkshopName varchar(40)
  as
  begin
    set nocount on;
    declare @WorkshopID as int

    set @WorkshopID = dbo.getWorkshopID (@Date, @ConferenceName,
@WorkshopName)
    update WorkshopReservation
    set Cancelled = 1
    where WorkshopID = @WorkshopID and ParticipantID = @ParticipantID
  end
go
```

Cancel_Conference_Day_Participation

Procedura usuwająca uczestnictwo w konkretnym dniu konferencji.

```
create procedure Cancel_Conference_Day_Participation
  @ConferenceName varchar(40),
  @ParticipantID int,
  @Date date
  as
  begin
    set nocount on;
    declare @ConferenceID as int
    declare @ConferenceDayID as int

    set @ConferenceID = dbo.getConferenceID (@ConferenceName)
    set @ConferenceDayID = dbo.getConferenceDayID (@Date, @ConferenceID)
```

```
    update ConferenceParticipants
    set Cancelled = 1
    where ConferenceDayID = @ConferenceDayID and ParticipantID =
@ParticipantID
  end
go
```

Cancel_Conference_Day_Reservation

Procedura usuwająca rezerwacje na dzień konferencji.

```
create procedure Cancel_Conference_Day_Reservation
  @ConferenceName varchar(40),
  @ParticipantID int,
  @Date date
  as
  begin
    set nocount on;
    declare @ConferenceID as int
    declare @ConferenceDayID as int

    set @ConferenceID = dbo.getConferenceID (@ConferenceName)
    set @ConferenceDayID = dbo.getConferenceDayID (@Date, @ConferenceID)
    update ConferenceDayReservation
    set [Cancelled ] = 1
    where ConferenceDayID = @ConferenceDayID and ParticipantID =
@ParticipantID
  end
go
```

Cancel_Unpaid_Reservations

Procedura usuwająca nieopłacone w terminie rezerwacje. Jest bezparametrowa.

```
create procedure Cancel_Unpaid_Reservations
  as
  begin
    set nocount on;
    begin transaction
      update ConferenceDayReservation
      set [Cancelled ] = 1
      where GETDATE() >= DueDate

      if @@error <> 0
      begin
        raiserror ('error', 16, 1)
```

```
            rollback transaction
        end

        update WorkshopReservation
        set Cancelled = 1
        where GETDATE() >= DueDate

        if @@error <> 0
        begin
            raiserror ('error', 16, 1)
            rollback transaction
        end
    Commit transaction
  end
go
```

Workshop_Payment

Procedura aktualizująca płatność za warsztat.

```
create procedure Workshop_Payment
  @ParticipantID int,
  @Amount money,
  @WorkshopName varchar(40),
  @ConferenceName varchar(40),
  @Date date
  as
  begin
    set nocount on;
    declare @WorkshopID as int

    set @WorkshopID = dbo.getWorkshopID (@Date, @ConferenceName,
@WorkshopName)   -- chyba trigger wleci

    update WorkshopReservation
    set AmountPaid += @Amount
    where ParticipantID = @ParticipantID and WorkshopID = @WorkshopID and
Cancelled = -1
  end
go
```

Conference_Payment

Procedura uaktualniająca płatność za warsztat.

```
create procedure Conference_Payment
```

```
  @ParticipantID varchar(40),
  @ConferenceName varchar(40),
  @Date date,
  @Amount money
  as
  begin
    set nocount on;
    declare @ConferenceID as int
    declare @ConferenceDayID as int

    set @ConferenceID = dbo.getConferenceID (@ConferenceName)
    set @ConferenceDayID = dbo.getConferenceDayID (@Date, @ConferenceID)
    update ConferenceDayReservation
    set AmountPaid += @Amount
    where ConferenceDayID = @ConferenceDayID and ParticipantID =
@ParticipantID
  end
go
```

Add_Company_Employee

Procedura zaznaczająca uczestnika jako pracownika firmy.

```
create procedure Add_Company_Employee
  @CompanyName varchar(40),
  @ParticipantID int
  as
  begin
    set nocount on;
    set @ParticipantID = (
      select ParticipantID
      from Participants
      where ParticipantID = @ParticipantID
    )
    set @CompanyName = (
      select CompanyName
      from Companies
      where CompanyName = @CompanyName
    )
    insert into CompanyEmployees(ParticipantID, CompanyName)
    values (@ParticipantID, @CompanyName)
  end
go
```

Add_Student

Procedura zaznaczające uczestnika jako studenta.

```
create procedure Add_Student
  @ParticipantID int,
  @StudentCardID int
  as
  begin
    set nocount on;
    set @ParticipantID = (
      select ParticipantID
      from Participants
      where ParticipantID = @ParticipantID
    )
    insert into Students(ParticipantID, StudentCardID)
    values (@ParticipantID, @StudentCardID)
  end
go
```

Add_Group_Conference_Day_Reservation

Procedura tworząca grupową rezerwacje na dzień konferencji (bez specyfikacji danych osobowych, do uzupełnienia później)

```
create procedure Add_Group_Conference_Day_Reservation
  @NumberOfSeats int,
  @ConferenceName varchar(40),
  @Date date
  as
  begin
    set nocount on;
    declare @ConferenceID as int
    declare @ConferenceDayID as int
    declare @ParticipantID as int
    declare @CompanyName as varchar(40)

    set @CompanyName = (
      select CompanyName
      from Companies
      where ClientID in (
        select Conferences.ClientID
        from Conferences
        where ConferenceID = @ConferenceID
      )
```

```
    )
    set @ConferenceID = dbo.getConferenceID (@ConferenceName)
    set @ConferenceDayID = dbo.getConferenceDayID (@Date, @ConferenceID)

    if dbo.getConferenceDayFreeSeats (@ConferenceDayID) >= @NumberOfSeats
    begin
      while @NumberOfSeats > 0
      begin
        exec dbo.Add_Participant
        set @ParticipantID = (
          select max(ParticipantID)
          from Participants
        )
        exec dbo.Add_Conference_Day_Reservation @ParticipantID =
@ParticipantID, @ConferenceName = @ConferenceName, @Date = @Date
        exec dbo.Add_Company_Employee @CompanyName =
@CompanyName, @ParticipantID = @ParticipantID
        set @NumberOfSeats -= 1
      end
    end
  end
go
```

Add_Group_Workshop_Conference_Day_Reservation

Procedura tworząca grupową rezerwacje na konferencje oraz na warsztat na tej konferencji (Bez danych osobowych, do uzupełnienia).

```
create procedure Add_Group_Workshop_Conference_Day_Reservation
  @NumberOfSeats int,
  @ConferenceName varchar(40),
  @Date date,
  @WorkshopName int
  as
  begin
    set nocount on;
    declare @WorkshopID as int
    declare @ParticipantID as int
    declare @ConferenceID as int
    declare @ConferenceDayID as int
    declare @CompanyName as varchar(40)

    set @CompanyName = (
      select CompanyName
      from Companies
```

```
        where ClientID in (
            select Conferences.ClientID
            from Conferences
            where ConferenceID = @ConferenceID
        )
    )
    set @ConferenceID = dbo.getConferenceID (@ConferenceName)
    set @ConferenceDayID = dbo.getConferenceDayID (@Date, @ConferenceID)

    set @WorkshopID = dbo.getWorkshopID (@Date, @ConferenceName,
@WorkshopName)

    if dbo.getWorkshopFreeSeats (@WorkshopID) >= @NumberOfSeats and
dbo.getConferenceDayFreeSeats (@ConferenceDayID) >= @NumberOfSeats
    begin
        while @NumberOfSeats > 0
        begin
            exec dbo.Add_Participant
            set @ParticipantID = (
                select max(ParticipantID)
                from Participants
            )
            exec dbo.Add_Conference_Day_Reservation @ParticipantID =
@ParticipantID, @ConferenceName = @ConferenceName, @Date = @Date
            exec dbo.Add_Workshop_Reservation @ConferenceName =
@ConferenceName, @Date = @Date, @WorkshopName = @WorkshopName,
@ParticipantID = @ParticipantID
            exec dbo.Add_Company_Employee @CompanyName =
@CompanyName, @ParticipantID = @ParticipantID
            set @NumberOfSeats -= 1
        end
    end
  end
go
```

TRIGGERY

CancelWorkshopReservation

Trigger anuluje rezerwację na warsztaty jeżeli rezerwacja na dni konferencji zostanie usunięta.

```
CREATE TRIGGER CancelWorkshopReservation

    ON ConferenceDayReservation
    FOR UPDATE AS
    BEGIN
        IF UPDATE(Cancelled)
        BEGIN
            DECLARE @PID AS int
            set @PID = (SELECT ParticipantID FROM inserted)

            UPDATE WorkshopReservation
            SET Cancelled = 1
            WHERE @PID = WorkshopReservation.ParticipantID
        end
    end
```

CancelConferenceReservation

Trigger anuluje rezerwację na konferencję jeżeli uczestnik zapłacił za udział w konferencji

```
CREATE TRIGGER CancelConferenceReservation
   ON ConferenceParticipants
   FOR UPDATE AS
   BEGIN
       IF (SELECT COUNT(*) from inserted) = 1
       BEGIN
           DECLARE @PID AS int
           set @PID = (SELECT ParticipantID FROM inserted)

           UPDATE ConferenceDayReservation
           SET Cancelled = 1
           WHERE @PID = ConferenceDayReservation.ParticipantID
       end
   end
```

CancelWorkshopReservationWhenRegistrationIsCanceled

Trigger anuluje rezerwację na warsztat jeżeli uczestnik zapłacił za udział w warsztacie

```
CREATE TRIGGER CancelWorkshopReservationWhenRegistrationIsCanceled
```

```
ON WorkshopParticipants
FOR UPDATE AS
BEGIN
    IF (SELECT COUNT(*) from inserted) = 1
    BEGIN
        DECLARE @PID AS int
        set @PID = (SELECT ParticipantID FROM inserted)

        UPDATE WorkshopReservation
        SET Cancelled = 1
        WHERE @PID = WorkshopReservation.ParticipantID
    end
end
```