Dokumentacja Projektu Bazy Danych

“Konferencje”

Autorzy:
Gabriel Nowak
Bartosz Drzewicki

Spis treści:

15. CancelWorkshopReservation
16. CancelOrderAfterWeek
17. ChangeCustomerData
18. ChangeConfDayCapacity
19. ChangeWorkshopCapacity

Funkcje:

1. ConferenceDayPlacesLeft
2. WorkshopPlacesLeft
3. ConferenceDayParticipantsList
4. WorkshopParticipantsList
5. ConferenceDays
6. WorkshopsOfConferenceList
7. ConferencePriceTresholds
8. OrderValue

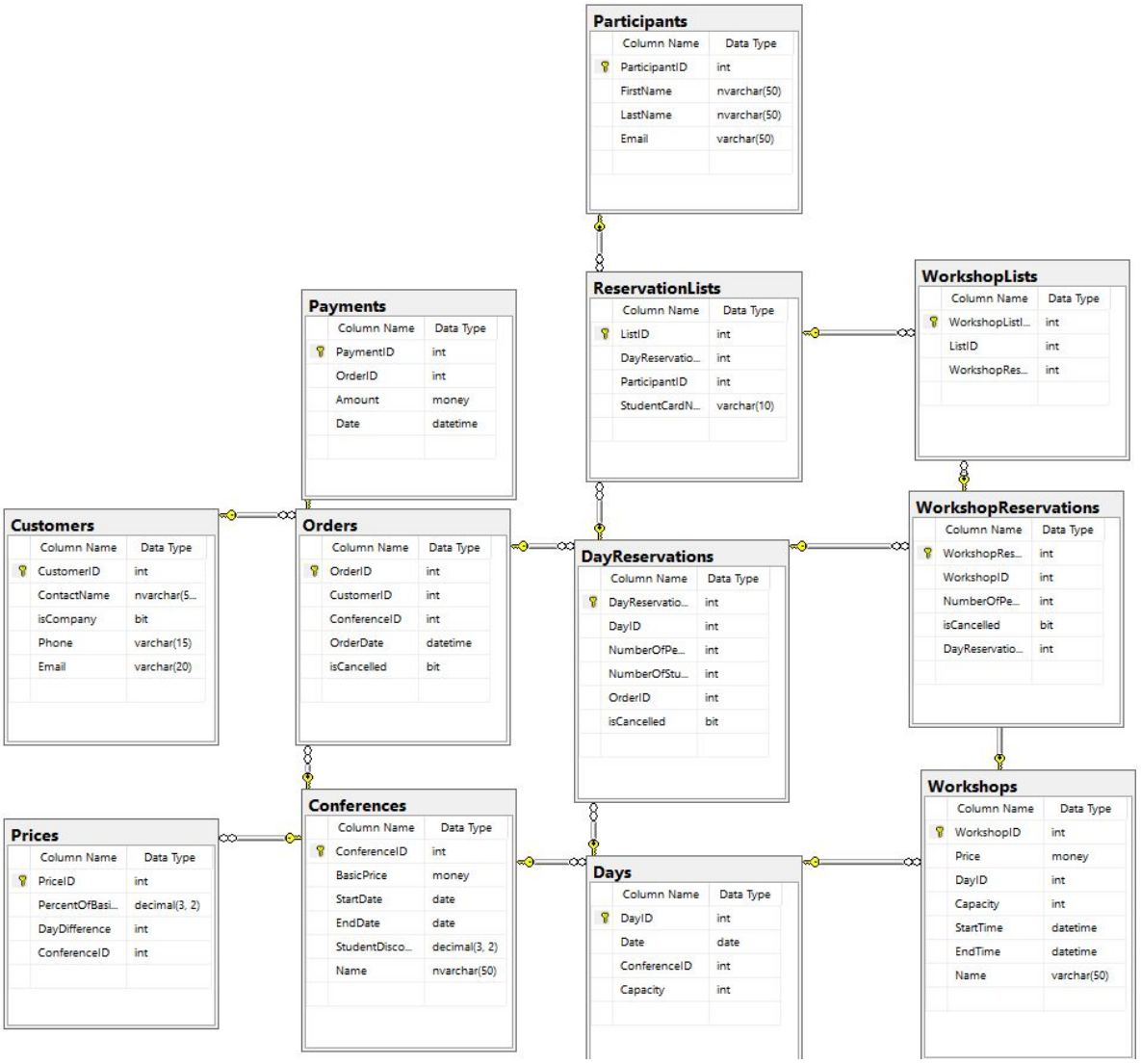
Triggery:

1. CancelOrderTrigger
2. CancelDayReservation
3. UpdateReservationListTrigger
4. InsertReservationListTrigger
5. AddReservationListRecordTrigger
6. WorkshopsParticipantsLevelTrigger

Role:

1. Administrator
2. Menedżer ds konferencji
3. Księgowy
4. Obsługa konferencji

Schemat:


Participants
Column Name Data Type
ParticipantID int
FirstName nvarchar(50)
LastName nvarchar(50)
Email varchar(50)
ReservationLists
Column Name Data Type
ListID int
DayReservatio... int
ParticipantID int
StudentCardN... varchar(10)
WorkshopLists
Column Name Data Type
WorkshopListI... int
ListID int
WorkshopRes... int
Payments
Column Name Data Type
PaymentID int
OrderID int
Amount money
Date datetime
Customers
Column Name Data Type
CustomerID int
ContactName nvarchar(5...
isCompany bit
Phone varchar(15)
Email varchar(20)
Orders
Column Name Data Type
OrderID int
CustomerID int
ConferenceID int
OrderDate datetime
isCancelled bit
DayReservations
Column Name Data Type
DayReservatio... int
DayID int
NumberOfPe... int
NumberOfStu... int
OrderID int
isCancelled bit
WorkshopReservations
Column Name Data Type
WorkshopRes... int
WorkshopID int
NumberOfPe... int
isCancelled bit
DayReservatio... int
Prices
Column Name Data Type
PriceID int
PercentOfBasi... decimal(3, 2)
DayDifference int
ConferenceID int
Conferences
Column Name Data Type
ConferenceID int
BasicPrice money
StartDate date
EndDate date
StudentDisco... decimal(3, 2)
Name nvarchar(50)
Days
Column Name Data Type
DayID int
Date date
ConferenceID int
Capacity int
Workshops
Column Name Data Type
WorkshopID int
Price money
DayID int
Capacity int
StartTime datetime
EndTime datetime
Name varchar(50)

Tabele:

1. Customers

Tabela Customers przechowuje wszystkie niezbędne dane na temat klientów firmy, takie jak indywidualne CustomerID, nazwę kontaktową (ContactName), flagę isCompany, numer telefonu i adres email.

```
CREATE TABLE [dbo].[Customers](
          [CustomerID] [int] IDENTITY(1,1) NOT NULL,
          [ContactName] [nvarchar](50) NOT NULL,
          [isCompany] [bit] NOT NULL,
          [Phone] [varchar](15) NULL,
          [Email] [varchar](20) NOT NULL,
 CONSTRAINT [PK_Customers] PRIMARY KEY CLUSTERED
(
          [CustomerID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY],
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]
GO

ALTER TABLE [dbo].[Customers]  WITH CHECK ADD  CONSTRAINT [CK_Customers_1] CHECK  ((NOT [Phone] like '%[^0-9]%'))
GO

ALTER TABLE [dbo].[Customers] CHECK CONSTRAINT [CK_Customers_1]
GO
```

2. Orders

Tabela Orders przechowuje informacje na temat złożonych zamówień, takie jak indywidualne OrderID, ID klienta, ID zamawianej konferencji, datę zamówienia czy flagę o anulacji zamówienia.

```
CREATE TABLE [dbo].[Orders](
          [OrderID] [int] IDENTITY(1,1) NOT NULL,
          [CustomerID] [int] NOT NULL,
          [ConferenceID] [int] NOT NULL,
          [OrderDate] [datetime] NOT NULL,
          [isCancelled] [bit] NOT NULL,
 CONSTRAINT [PK_Orders] PRIMARY KEY CLUSTERED
(
          [OrderID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Orders]  WITH CHECK ADD  CONSTRAINT [FK_Orders_Conferences] FOREIGN KEY([ConferenceID])
REFERENCES [dbo].[Conferences] ([ConferenceID])
GO

ALTER TABLE [dbo].[Orders] CHECK CONSTRAINT [FK_Orders_Conferences]
GO

ALTER TABLE [dbo].[Orders]  WITH CHECK ADD  CONSTRAINT [FK_Orders_Customers] FOREIGN KEY([CustomerID])
REFERENCES [dbo].[Customers] ([CustomerID])
GO

ALTER TABLE [dbo].[Orders] CHECK CONSTRAINT [FK_Orders_Customers]
GO
```

3. Payments

Tabela Payments przechowuje wszystkie płatności, a w szczególności informacje o OrderID dla jakiego wpłata jest przypisana, wartość płatności oraz jej data.

```
CREATE TABLE [dbo].[Payments](
          [PaymentID] [int] IDENTITY(1,1) NOT NULL,
          [OrderID] [int] NOT NULL,
          [Amount] [money] NOT NULL,
          [Date] [datetime] NOT NULL,
 CONSTRAINT [PK_Payments] PRIMARY KEY CLUSTERED
(
          [PaymentID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Payments]  WITH CHECK ADD  CONSTRAINT [FK_Payments_Orders] FOREIGN KEY([OrderID])
REFERENCES [dbo].[Orders] ([OrderID])
GO

ALTER TABLE [dbo].[Payments] CHECK CONSTRAINT [FK_Payments_Orders]
GO

ALTER TABLE [dbo].[Payments]  WITH CHECK ADD  CONSTRAINT [CK_Payments] CHECK  (([amount]>(0)))
GO

ALTER TABLE [dbo].[Payments] CHECK CONSTRAINT [CK_Payments]
GO
```

4. Conferences

Przechowuje informacje o konferencjach jakie wykonywała bądź będzie wykonywać firma, a w szczególności ID konferencji, cenę podstawową, datę początku/końca konferencji, zniżkę studencką czy nazwę konferencji.

```
CREATE TABLE [dbo].[Conferences](
          [ConferenceID] [int] IDENTITY(1,1) NOT NULL,
          [BasicPrice] [money] NOT NULL,
          [StartDate] [date] NOT NULL,
          [EndDate] [date] NOT NULL,
          [StudentDiscount] [decimal](3, 2) NOT NULL,
          [Name] [nvarchar](50) 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]
GO

ALTER TABLE [dbo].[Conferences]  WITH CHECK ADD  CONSTRAINT [CK_Conferences] CHECK  (([basicprice]>(0)))
GO

ALTER TABLE [dbo].[Conferences] CHECK CONSTRAINT [CK_Conferences]
GO

ALTER TABLE [dbo].[Conferences]  WITH CHECK ADD  CONSTRAINT [CK_Conferences_1] CHECK  (([startdate]<=[enddate]))
GO

ALTER TABLE [dbo].[Conferences] CHECK CONSTRAINT [CK_Conferences_1]
GO

ALTER TABLE [dbo].[Conferences]  WITH CHECK ADD  CONSTRAINT [CK_Conferences_2] CHECK  (([studentdiscount]>=(0)))
GO

ALTER TABLE [dbo].[Conferences] CHECK CONSTRAINT [CK_Conferences_2]
GO
```

5. Prices

Tabela Prices przechowuje informacje o progach cenowych wszystkich konferencj.

```
CREATE TABLE [dbo].[Prices](
          [PriceID] [int] IDENTITY(1,1) NOT NULL,
          [PercentOfBasicPrice] [decimal](3, 2) NOT NULL,
          [DayDifference] [int] NOT NULL,
          [ConferenceID] [int] NOT NULL,
 CONSTRAINT [PK_Prices] PRIMARY KEY CLUSTERED
(
          [PriceID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Prices]  WITH CHECK ADD  CONSTRAINT [FK_Prices_Conferences] FOREIGN KEY([ConferenceID])
REFERENCES [dbo].[Conferences] ([ConferenceID])
GO

ALTER TABLE [dbo].[Prices] CHECK CONSTRAINT [FK_Prices_Conferences]
GO

ALTER TABLE [dbo].[Prices]  WITH CHECK ADD  CONSTRAINT [CK_Prices] CHECK  (([PercentOfBasicPrice]<(1) AND
[PercentOfBasicPrice]>=(0)))
GO

ALTER TABLE [dbo].[Prices] CHECK CONSTRAINT [CK_Prices]
GO

ALTER TABLE [dbo].[Prices]  WITH CHECK ADD  CONSTRAINT [CK_Prices_1] CHECK  (([daydifference]>(0)))
GO

ALTER TABLE [dbo].[Prices] CHECK CONSTRAINT [CK_Prices_1]
GO
```

6. Days

Tabela przechowuje informacje o konkretnym dniu konferencji. Zawiera identyfikator dnia(DayID), datę(Date),identyfikator konferencji(ConferenceID) oraz limit uczestników danego dnia(Capacity)

```
CREATE TABLE [dbo].[Days](
          [DayID] [int] IDENTITY(1,1) NOT NULL,
          [Date] [date] NOT NULL,
          [ConferenceID] [int] NOT NULL,
          [Capacity] [int] NOT NULL,
 CONSTRAINT [PK_Days] PRIMARY KEY CLUSTERED
(
          [DayID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Days]  WITH CHECK ADD  CONSTRAINT [FK_Days_Conferences] FOREIGN KEY([ConferenceID])
REFERENCES [dbo].[Conferences] ([ConferenceID])
GO

ALTER TABLE [dbo].[Days] CHECK CONSTRAINT [FK_Days_Conferences]
GO

ALTER TABLE [dbo].[Days]  WITH CHECK ADD  CONSTRAINT [CK_Days] CHECK  (([capacity]>(0)))
GO

ALTER TABLE [dbo].[Days] CHECK CONSTRAINT [CK_Days]
GO
```

7. Workshops

Tabela przechowuje informacje o warsztatach jakie odbywają się w czasie trwania konferencji. Zawiera identyfikator warsztatu(WorkshopID), Cenę(Price), identyfikator dnia konferencji w któym odbywa się warsztat(DayID), limit uczestników(Capacity), datę rozpoczęcia(StartTime), datę zakończenia (EndTime) oraz nazwę warsztatu(Name).

```
CREATE TABLE [dbo].[Workshops](
          [WorkshopID] [int] IDENTITY(1,1) NOT NULL,
          [Price] [money] NULL,
          [DayID] [int] NOT NULL,
          [Capacity] [int] NOT NULL,
          [StartTime] [datetime] NOT NULL,
          [EndTime] [datetime] NOT NULL,
          [Name] [varchar](50) 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]
GO

ALTER TABLE [dbo].[Workshops]  WITH CHECK ADD  CONSTRAINT [FK_Workshops_Days] FOREIGN KEY([DayID])
REFERENCES [dbo].[Days] ([DayID])
GO
ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [FK_Workshops_Days]
GO

ALTER TABLE [dbo].[Workshops]  WITH CHECK ADD  CONSTRAINT [CK_Workshops] CHECK  (([Capacity]>(0)))
GO
ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [CK_Workshops]
GO

ALTER TABLE [dbo].[Workshops]  WITH CHECK ADD  CONSTRAINT [CK_Workshops_1] CHECK  (([starttime]<[endtime]))
GO
ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [CK_Workshops_1]
GO
```

8. DayReservations

Tabela przechowuje informacje o rezerwacjach na konkretny dzień konferencji. Zawiera identyfikator rezerwacji(DayReservationID),identyfikator dnia konferencji(DayID),ilość osób(NumberOfPeople),ilość osób będących studentami(NumberOfStudents),identyfikator zamówienia(OrderID) oraz flagę czy rezerwacja została anulowana(isCancelled).

```
CREATE TABLE [dbo].[DayReservations](
          [DayReservationID] [int] IDENTITY(1,1) NOT NULL,
          [DayID] [int] NOT NULL,
          [NumberOfPeople] [int] NOT NULL,
          [NumberOfStudents] [int] NOT NULL,
          [OrderID] [int] NOT NULL,
          [isCancelled] [bit] NOT NULL,
 CONSTRAINT [PK_DayReservations] PRIMARY KEY CLUSTERED
(
          [DayReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
 ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[DayReservations]  WITH CHECK ADD  CONSTRAINT [FK_DayReservations_Days] FOREIGN KEY([DayID])
REFERENCES [dbo].[Days] ([DayID])
GO
ALTER TABLE [dbo].[DayReservations] CHECK CONSTRAINT [FK_DayReservations_Days]
GO

ALTER TABLE [dbo].[DayReservations]  WITH CHECK ADD  CONSTRAINT [FK_DayReservations_Orders] FOREIGN KEY([OrderID])
REFERENCES [dbo].[Orders] ([OrderID])
GO
ALTER TABLE [dbo].[DayReservations] CHECK CONSTRAINT [FK_DayReservations_Orders]
GO

ALTER TABLE [dbo].[DayReservations]  WITH CHECK ADD  CONSTRAINT [CK_DayReservations] CHECK  (([NumberOfPeople]>(0)))
GO
ALTER TABLE [dbo].[DayReservations] CHECK CONSTRAINT [CK_DayReservations]
GO

ALTER TABLE [dbo].[DayReservations]  WITH CHECK ADD  CONSTRAINT [CK_DayReservations_1] CHECK
(([NumberOfPeople]>=[NumberOfStudents]))
GO
ALTER TABLE [dbo].[DayReservations] CHECK CONSTRAINT [CK_DayReservations_1]
GO

ALTER TABLE [dbo].[DayReservations]  WITH CHECK ADD  CONSTRAINT [CK_DayReservations_2] CHECK  (([NumberOfStudents]>=(0)))
GO
ALTER TABLE [dbo].[DayReservations] CHECK CONSTRAINT [CK_DayReservations_2]
GO
```

9. WorkshopReservation

Tabela przechowuje informacje o rezerwacjach na warsztaty. Zawiera identyfikator rezerwacji(WorshopReservationID), identyfikator warsztatu(WorkshopID),ilość osób(NumberOfPeople), flagę czy rezerwacja została anulowana(isCancelled) oraz identyfikator rezerwacji na dzień konferencji w którym odbywa się warsztat(DayReservationID).

```
CREATE TABLE [dbo].[WorkshopReservations](
          [WorkshopReservationID] [int] IDENTITY(1,1) NOT NULL,
          [WorkshopID] [int] NOT NULL,
          [NumberOfPeople] [int] NOT NULL,
          [isCancelled] [bit] NOT NULL,
          [DayReservationID] [int] NOT NULL,
 CONSTRAINT [PK_WorkshopReservations] PRIMARY KEY CLUSTERED
(
          [WorkshopReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
 ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[WorkshopReservations]  WITH CHECK ADD  CONSTRAINT [FK_WorkshopReservations_DayReservations] FOREIGN
KEY([DayReservationID])
REFERENCES [dbo].[DayReservations] ([DayReservationID])
GO
ALTER TABLE [dbo].[WorkshopReservations] CHECK CONSTRAINT [FK_WorkshopReservations_DayReservations]
GO

ALTER TABLE [dbo].[WorkshopReservations]  WITH CHECK ADD  CONSTRAINT [FK_WorkshopReservations_Workshops] FOREIGN
KEY([WorkshopID])
REFERENCES [dbo].[Workshops] ([WorkshopID])
GO
ALTER TABLE [dbo].[WorkshopReservations] CHECK CONSTRAINT [FK_WorkshopReservations_Workshops]
GO

ALTER TABLE [dbo].[WorkshopReservations]  WITH CHECK ADD  CONSTRAINT [CK_WorkshopReservations] CHECK
(([NumberOfPeople]>(0)))
GO
ALTER TABLE [dbo].[WorkshopReservations] CHECK CONSTRAINT [CK_WorkshopReservations]
GO
```

10. WorkshopLists

Tabela przechowuje liste uczestnikow warsztatów. Zawiera identyfikator(WorkshopListID), identyfikator listy uczestnika na dany dzień konferencji(ListID) oraz identyfikator rezerwacji(WorkshopReservationID).

```
CREATE TABLE [dbo].[WorkshopLists](
          [WorkshopListID] [int] IDENTITY(1,1) NOT NULL,
          [ListID] [int] NOT NULL,
          [WorkshopReservationID] [int] NOT NULL,
 CONSTRAINT [PK_WorkshopLists] PRIMARY KEY CLUSTERED
(
          [WorkshopListID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[WorkshopLists]  WITH CHECK ADD  CONSTRAINT [FK_WorkshopLists_ReservationLists] FOREIGN KEY([ListID])
REFERENCES [dbo].[ReservationLists] ([ListID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[WorkshopLists] CHECK CONSTRAINT [FK_WorkshopLists_ReservationLists]
GO

ALTER TABLE [dbo].[WorkshopLists]  WITH CHECK ADD  CONSTRAINT [FK_WorkshopLists_WorkshopReservations] FOREIGN
KEY([WorkshopReservationID])
REFERENCES [dbo].[WorkshopReservations] ([WorkshopReservationID])
GO
ALTER TABLE [dbo].[WorkshopLists] CHECK CONSTRAINT [FK_WorkshopLists_WorkshopReservations]
GO
```

11. ReservationLists

Tabela przechowuje liste uczestnikow konferencji. Zawiera identyfikator(ListID), identyfikator powiązanej rezerwacji(DayReserationID),identyfikator uczestnika(ParticipantID) oraz opcjonalnie numer legitymacji studenckiej.

```
CREATE TABLE [dbo].[ReservationLists](
          [ListID] [int] IDENTITY(1,1) NOT NULL,
          [DayReservationID] [int] NOT NULL,
          [ParticipantID] [int] NOT NULL,
          [StudentCardNumber] [varchar](10) NULL,
 CONSTRAINT [PK_ReservationLists] PRIMARY KEY CLUSTERED
(
          [ListID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
 ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[ReservationLists]  WITH CHECK ADD  CONSTRAINT [FK_ReservationLists_Participants] FOREIGN
KEY([ParticipantID])
REFERENCES [dbo].[Participants] ([ParticipantID])
GO
ALTER TABLE [dbo].[ReservationLists] CHECK CONSTRAINT [FK_ReservationLists_Participants]
GO

ALTER TABLE [dbo].[ReservationLists]  WITH CHECK ADD  CONSTRAINT [FK_ReservationLists_ReservationLists] FOREIGN
KEY([DayReservationID])
REFERENCES [dbo].[DayReservations] ([DayReservationID])
GO
ALTER TABLE [dbo].[ReservationLists] CHECK CONSTRAINT [FK_ReservationLists_ReservationLists]
GO
```

12. Participants

Tabela Participants przechowuje informacje o wszystkich uczestnikach konferencji i warsztatów, takie jak imię, nazwisko, unikalne ID oraz email.

```
CREATE TABLE [dbo].[Participants](
          [ParticipantID] [int] IDENTITY(1,1) NOT NULL,
          [FirstName] [nvarchar](50) NOT NULL,
          [LastName] [nvarchar](50) NOT NULL,
          [Email] [varchar](50) NULL,
 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]
GO
```

Widoki:

1. ConferencesNotStartedYet

Wyświetla informacje o konferencjach które się jeszcze nie rozpoczęły.

```
CREATE VIEW [dbo].[ConferencesNotStartedYet]
AS
SELECT        ConferenceID, BasicPrice, StartDate, EndDate, StudentDiscount, Name, DATEDIFF(dd, GETDATE(), StartDate) AS
DaysToStart
FROM            dbo.Conferences
WHERE        (DATEDIFF(dd, GETDATE(), StartDate) > 0)
GO
```

2. AvailableWorkshops

Widok zwraca listę warsztatów, które odbędą się w przyszłości lub trwają obecnie wraz z nazwą, datą, maksymalną liczbą uczestników oraz zajętymi miejscami.

```
CREATE VIEW [dbo].[AvailableWorkshops]
AS
SELECT w.Name, w.StartTime, w.EndTime, w.Capacity AS 'Participants limit',
          (SELECT isNULL(sum(wr.NumberOfPeople),0)
          FROM Workshops AS WORK
          JOIN WorkshopReservations AS wr ON wr.WorkshopID = WORK .WorkshopID
    WHERE wr.isCancelled = 0 AND WORK .EndTime >= GETDATE()
          AND wr.WorkshopID = w.WorkshopID) AS 'Participants '
FROM Workshops AS w
GO
```

3. ParticipantsOfFutureConferences

Wyświetla listę uczestników, którzy są zapisani na jedną z nadchodzących konferencji.

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

select  d.conferenceid,dr.dayid,rl.participantid,rl.studentcardnumber, p.firstname,p.lastname,p.email
from reservationlists as rl
          inner join participants as p on rl.participantid = p.participantid
          inner join dayreservations as dr on rl.dayreservationid = dr.dayreservationid
          inner join days as d on dr.dayid = d.dayid
where GETDATE()<d.date
GO
```

4. WorkshopsParticipantsList

Widok zwraca listę uczestników w przyszłych i obecnie trwających warsztatach.

```
create view [dbo].[WorkshopsParticipantsList] as

          select distinct p.ParticipantID,p.FirstName,p.LastName,w.Name
                    from WorkshopReservations as wr
                    join WorkshopLists as wl on wl.WorkshopReservationID=wr.WorkshopReservationID
                    join ReservationLists as rl on rl.ListID=wl.ListID
                    join Participants as p on p.ParticipantID=rl.ParticipantID
                    join Workshops as w on w.WorkshopID=wr.WorkshopID
                    where w.EndTime>=GETDATE()
GO
```

5. ConferencePopularity

Widok zwraca listę odbytych konferencji z ilością uczestników jaka brała w niej udział, maksymalną liczbe uczestników oraz procent zapełnienia.

```
CREATE view [dbo].[ConferencePopularity] as
          select con.Name ,
                    (select isNULL(sum(dr.NumberOfPeople),0)
                    from Conferences as c
                    join Days as d on d.ConferenceID=c.ConferenceID
                    join DayReservations as dr on dr.DayID=d.DayID
                    where dr.isCancelled=0 and c.ConferenceID=con.ConferenceID
                    group by dr.DayReservationID)
                    as Participants ,

                    (select sum(d.Capacity)
                     from Days as d
                    where d.ConferenceID=con.ConferenceID)
                    as ParticipantsLimit ,

                    ((select isNULL(sum(dr.NumberOfPeople),0)
                    from Conferences as c
                    join Days as d on d.ConferenceID=c.ConferenceID
                    join DayReservations as dr on dr.DayID=d.DayID
                    where dr.isCancelled=0 and c.ConferenceID=con.ConferenceID
                    group by dr.DayReservationID)*100/
                              (select isNULL(sum(d.Capacity),0)
                              from Days as d
                              where d.ConferenceID=con.ConferenceID))
                    as 'Fill'

          from Conferences as con
          join Days as d on d.ConferenceID=con.ConferenceID
          where con.EndDate < GETDATE()
          group by con.ConferenceID,con.Name
GO
```

6. WorkshopsStatistic

Wyświetla statystyki warsztatów, między innymi ich procentowe obłożenie.

```
CREATE VIEW [dbo].[WorkshopsStatistics]
AS
SELECT        WorkshopID, Name,
                            (SELECT        ISNULL(SUM(NumberOfPeople), 0) AS Expr1
                              FROM            dbo.WorkshopReservations
                              WHERE        (WorkshopID = w.WorkshopID) AND (isCancelled = 0)) AS Participants, Capacity AS
'Participants limit',
                            (SELECT        ISNULL(SUM(NumberOfPeople), 0) AS Expr1
                              FROM            dbo.WorkshopReservations AS WorkshopReservations_1
                              WHERE        (WorkshopID = w.WorkshopID) AND (isCancelled = 0)) * 100 / Capacity AS Fill
FROM            dbo.Workshops AS w
GO
```

7. OrdersNotPayedYet

Wyświetla zamówienia, które nie zostały jeszcze opłacone.

```
create view [dbo].[OrdersNotPayedYet]
as
          select * from OrdersPaymentStatus
          where Payed < [Order Value]
GO
```

8. ReservationsWithoutFilledParticipants

Wyświetla informacje o rezerwacjach dla których nie zostały wypełnione dane dot. uczestników danego dnia konferencji.

```
create view [dbo].[ReservationsWithoutFilledParticipants] as
          select dr.dayreservationid, dr.numberofpeople, dr.numberofstudents,
          (select count(*) from reservationlists where dr.dayreservationid = reservationlists.dayreservationid) as
PeopleFilled ,
          (select count(*) from reservationlists where dr.dayreservationid = reservationlists.dayreservationid
                    and reservationlists.StudentCardNumber is not null) as StudentsFilled
          from DayReservations as dr
          where (select count(*) from reservationlists where dr.dayreservationid = reservationlists.dayreservationid) !=
dr.numberofpeople
                    or (select count(*) from reservationlists where dr.dayreservationid = reservationlists.dayreservationid
                    and reservationlists.StudentCardNumber is not null) != dr.numberofstudents
GO
```

9. ClientRanking

Widok zwraca listę klientów z informacją o liczbie konferencji, dni konferencji, warsztatów w jakich brali udział.

```
create view [dbo].[ClientRanking] as
select c.CustomerID,c.ContactName,

          (select distinct count(o.ConferenceID)
                    from Orders as o
                    where o.CustomerID=c.CustomerID and o.isCancelled=0)
          as BookedConferences,

          (select distinct count(dr.DayID)
                    from Orders as o
                    join DayReservations as dr on dr.OrderID=o.OrderID
                    where o.CustomerID=c.CustomerID and dr.isCancelled=0 and o.isCancelled=0)
          as BookedConferenceDays,

          (select distinct count(wr.WorkshopID)
                    from Orders as o
                    join DayReservations as dr on dr.OrderID=o.OrderID
                    join WorkshopReservations as wr on wr.DayReservationID=dr.DayReservationID
                    where o.CustomerID=c.CustomerID and wr.isCancelled=0
                              and dr.isCancelled=0 and o.isCancelled=0)
          as BookedWorkshops
from Customers as c
GO
```

10. ParticipantsStatistics

Wyświetla informacje o aktywności uczestników, w tym o ilości przebytych konferencji i warsztatów.

```
create view [dbo].[ParticipantsStatistics] as
          select p.participantid,p.firstname, p.lastname ,
          (select count(*) from reservationlists as r where r.participantid = p.participantid ) as [conferences joined] ,
          (select count(*) from workshoplists as w where w.listid = rl.listid) as [workshop joined]
          from participants as p
                    inner join reservationlists as rl on p.participantid = rl.listid
GO
```

11. OrdersPaymentStatus

Widok zwraca id zamówienia, kwotę jaką wpłacono oraz kwotę należną.

```
CREATE VIEW [dbo].[OrdersPaymentStatus]
AS
          SELECT dbo.Orders.OrderID, SUM(dbo.Payments.Amount) AS Payed,
                    dbo.OrderValue(dbo.Orders.OrderID) AS 'Order Value'
                    FROM  dbo.Orders
                    join dbo.Payments ON dbo.Orders.OrderID = dbo.Payments.OrderID
                    where dbo.Orders.isCancelled=0
                    GROUP BY dbo.Orders.OrderID
GO
```

12. OrdersOverPaid

Wyświetla informacje o zamówieniach które zostały przepłacone.

```
CREATE VIEW [dbo].[OrdersOverPaid]
AS
SELECT        ops.OrderID, c.ContactName, c.Email, ops.Payed - ops.[Order Value] AS 'OverPaid'
FROM            dbo.OrdersPaymentStatus AS ops INNER JOIN
                         dbo.Orders AS o ON o.OrderID = ops.OrderID INNER JOIN
                         dbo.Customers AS c ON c.CustomerID = o.CustomerID
WHERE        (ops.Payed > ops.[Order Value])
GO
```

13. ReservationsFillParticipantStatus

Wyświetla wszystkie rezerwacje oraz ilość zarezerwowanych miejsc i ilość wypełnionych danych dot. uczestników (w tym informacji i numerach albumów studenckich).

```
create view [dbo].[ReservationsFillParticipantStatus] as
          select dr.dayreservationid, dr.numberofpeople, dr.numberofstudents,
          (select count(*) from reservationlists where dr.dayreservationid = reservationlists.dayreservationid) as
PeopleFilled ,
          (select count(*) from reservationlists where dr.dayreservationid = reservationlists.dayreservationid
                    and reservationlists.StudentCardNumber is not null) as StudentsFilled
          from DayReservations as dr
GO
```

Procedury

1. AddConference

Procedura dodająca konferencję. Jako argument przyjmuje nazwę(Name), datę rozpoczęcia(StartDate), datę zakończenia (EndDate), cenę bazową (BasicPrice) oraz zniżkę dla studentów(StudentDiscount).

```
CREATE PROCEDURE [dbo].[AddConference]
          -- Add the parameters for the stored procedure here
          @Name nvarchar(50) ,
          @StartDate date,
          @EndDate date,
          @BasicPrice money,
          @StudentDiscount decimal(3,2)

AS
BEGIN
          if exists (select * from Conferences where Name=@Name and StartDate=@StartDate and EndDate=@EndDate)
          begin
                    THROW 52000, 'This conference already exists' , 1
          end
          else if (@EndDate < GETDATE())
          begin
                    THROW 52000, 'Can not add conference to the past',1
          end
          else
          begin
                    insert into Conferences (BasicPrice,StartDate,EndDate,StudentDiscount,Name)
                                                        values(@BasicPrice,@StartDate,@EndDate,@StudentDiscount,@Name)
          end
END
GO
```

2. AddDay

Procedura umożliwiająca dodanie dnia do danej konferencji i o danej pojemności przekazanych jako parametry.

```
CREATE PROCEDURE [dbo].[AddDay]
          @Date date,
          @ConferenceID int,
          @Capacity int
AS
BEGIN

          SET NOCOUNT ON;

          if not exists(select * from Conferences where ConferenceID = @ConferenceID)
          begin raiserror ('Conference witch such ID does not exist', -1, -1)
          end

          else if @Date not between (select StartDate from Conferences as c where c.conferenceID = @ConferenceID) AND (select
EndDate from Conferences as c where c.conferenceID = @ConferenceID)
          begin raiserror ('Date is not between start date and end date of this conference',-1,-1)
          end

          else if GETDATE() > (select c.EndDate from conferences as c where c.conferenceid = @conferenceID)
          begin raiserror ('You cannot add day of conference which is already done',-1,-1)
          end

          else if exists (select date from days where date = @date and ConferenceID=@ConferenceID)
          begin raiserror ('This day already exists',-1,-1)
          end

else begin
insert into Days (Date, ConferenceID, Capacity)
values (@Date, @ConferenceID, @Capacity)
end
END
```

3. AddWorkshop

Procedura dodająca warsztat. Jako argument przyjmuje identyfiaktor dnia konferencji w którym odbywa się warsztat(DayID), nazwę(Name), datę rozpoczęcia(StartDate), datę zakończenia (EndDate), cenę (Price) oraz maksymalną liczbę uczestników (Capacity).

```
CREATE PROCEDURE [dbo].[AddWorkshop]
          -- Add the parameters for the stored procedure here
          @DayID int,
          @Name varchar(50),
          @StartTime datetime,
          @EndTime datetime,
          @Capacity int,
          @Price money
AS
BEGIN
          declare @Day date
          set @Day = (select Date from Days where DayID=@DayID)
          declare @StartDay date
          declare @EndDay date
          set @StartDay = CONVERT(date, @StartTime)
          set @EndDay = CONVERT(date, @EndTime)

          if(@StartDay <> @Day or @EndDay <> @Day )
          begin
                    raiserror('Invalid start or end date',-1,-1)
          end
          else
          begin
                    insert into Workshops (Price,DayID,Capacity,StartTime,EndTime,Name)
                    values (@Price,@DayID,@Capacity,@StartTime,@EndTime,@Name)
          end
END
GO
```

4. AddCustomer

Procedura umożliwiająca dodanie nowego klienta do bazy o podanych jako parametr danych.

```
CREATE PROCEDURE [dbo].[AddCustomer]
          @ContactName nvarchar(50),
          @isCompany bit,
          @Phone varchar(15),
          @Email varchar(20)

AS
BEGIN

          SET NOCOUNT ON;

    -- Insert statements for procedure here
          insert into Customers (ContactName, isCompany, Phone, Email)
          values (@ContactName,@isCompany,@Phone,@Email)
END
```

5. AddOrder

Procedura dodająca zamówienie na konferencje. Jako argument przyjmuje identyfikator klienta (CustomerID) oraz identyfikator konferencji (ConferenceID).

```
CREATE PROCEDURE [dbo].[AddOrder]
          @CustomerID int,
          @ConferenceID int
AS
BEGIN
          if not exists (select * from Customers where CustomerID=@CustomerID)
          begin
                    raiserror('Customer does not exists' , -1 ,-1)
          end
          else if not exists (select * from Conferences where ConferenceID=@ConferenceID)
          begin
                    raiserror('Conference does not exists' , -1 ,-1)
          end
          else if (GETDATE() > (select StartDate from Conferences where ConferenceID=@ConferenceID))
          begin
                    raiserror('Can not book past conference',-1,-1)
          end
          else
          begin
                    insert into Orders (CustomerID,ConferenceID,OrderDate,isCancelled)
                    values (@CustomerID,@ConferenceID,GETDATE(),0)
          end
END
GO
```

6. AddPayment

Procedura umożliwiająca dodanie płatności do danego ID zamówienia przekazanego jako parametr.

```
CREATE PROCEDURE [dbo].[AddPayment]
          @OrderID int,
          @Amount money
AS
BEGIN

          SET NOCOUNT ON;

          if not exists (select orderid from orders where orderid = @orderid)
          begin raiserror('Order with such ID does not exist',-1,-1)
          end

          else if @Amount < 0
          begin raiserror ('Amount must be more than zero',-1,-1)
          end

          else if (select isCancelled from orders where orderid = @OrderID) = 1
          begin raiserror('You cannot add payment to order which is already cancelled',-1,-1)
          end

          else
          insert into payments (orderid, amount, date)
          values(@OrderID, @Amount, GETDATE())

END
GO
```

7. AddNewParticipant

Procedura dodająca do bazy nowego uczestnika. Jako argument przyjmuje imię(FirstName), nazwisko (LastName) oraz opcjonalnie adres email(Email).

```
CREATE PROCEDURE [dbo].[AddNewParticipant]
          @FirstName nvarchar(50),
          @LastName nvarchar(50),
          @Email varchar(50)=null
AS
BEGIN
          insert into Participants (FirstName,LastName,Email)
          values (@FirstName,@LastName,@Email)
END
GO
```

8. AddPrice

Procedura umożliwiająca dodanie progu cenowego.

```
CREATE PROCEDURE [dbo].[AddPrice]
          @PercentofBasicPrice decimal (3,2),
          @DayDifference int,
          @ConferenceID int
AS
BEGIN
          if not exists (select * from conferences where conferenceid = @ConferenceID)
          begin raiserror('Conference witch such ID does not exist',-1,-1)
          end
          else if (@DayDifference <= 0)
          begin
                    raiserror('DayDifference must be higher than 0',-1,-1)
          end
          else
          insert into prices (PercentOfBasicPrice, DayDifference, ConferenceID)
          values (@PercentofBasicPrice,@DayDifference,@ConferenceID)
END
GO
```

9. AddDayReservation

Procedura dodająca rezerwację na wybrany dzień konferencji. Jako argument przyjmuje identyfikator dnia(DayID), ilość osób (NumberOfPeople), ilość studentów wśród wszystkich osób (NumberOfStudents) oraz identyfikator zamówienia(OrderID).

```
CREATE PROCEDURE [dbo].[AddDayReservation]
          @DayID int,
          @NumberOfPeople int,
          @NumberOfStudents int=0,
          @OrderID int
AS
BEGIN
          if(select ConferenceID from Orders where OrderID=@OrderID) <>
                    (select ConferenceID from Days where DayID=@DayID)
           begin
                    raiserror('You are trying to book conference day which you havent ordered.',-1,-1)
          end
          else if not exists (select * from Orders where OrderID=@OrderID)
          begin
                    raiserror('Order does not exists',-1,-1)
          end
    else if (dbo.ConferenceDayPlacesLeft(@DayID) >= @NumberOfPeople)
          begin
                    insert into DayReservations (DayID,NumberOfPeople,NumberOfStudents,OrderID,isCancelled)
                    values (@DayID,@NumberOfPeople,@NumberOfStudents,@OrderID,0)
          end
          else
          begin
                    raiserror('There are not enough free places',-1,-1)
          end
END
GO
```

10. AddWorkshopReservation

Procedura dodająca rezerwację danego warsztatu, której jako parametry przekazujemy ID warsztatu na który chcemy złożyć zamówienie, ilość miejsc jakie chcemy zabukować oraz ID dnia rezerwacji.

```
CREATE PROCEDURE [dbo].[AddWorkshopReservation]
          @WorkshopID int,
          @NumberOfPeople int,
          @DayReservationID int
AS
BEGIN

          SET NOCOUNT ON;

          if (select dbo.WorkshopPlacesLeft(@WorkshopID)) - @NumberOfPeople < 0
          begin raiserror('Capacity of this workshop is smaller than Number of People you want to book',-1,-1)
          end

          else if not exists (select * from workshops where workshopid = @WorkshopID)
          begin raiserror('Wrong WorkshopID',-1,-1)
          end

          else if not exists(select * from dayreservations where dayreservationid = @DayReservationID)
          begin raiserror('Wrong DayReservationID',-1,-1)
          end

          else if (select NumberOfPeople from dayreservations where dayreservationid = @DayReservationID) -
          (select count (*) from workshopreservations where dayreservationid = @DayReservationID and workshopID =
@WorkshopID) - @NumberOfPeople<0
          begin raiserror('You want to add more people to this workshop than is reserved in this DayReservation',-1,-1)
          end
          else if ((select DayID from DayReservations where DayReservationID=@DayReservationID) <> (select DayId from
Workshops where WorkshopID=@WorkshopID))
          begin
                    raiserror('There is no such workshop during your conference day',-1,-1)
          end
          else
          insert into workshopreservations (workshopid,numberofpeople,iscancelled,dayreservationid)
          values (@WorkshopID,@NumberOfPeople,0,@DayReservationID)

END
GO
```

11. AddParticipantToConferenceDay

Procedura przypisująca uczestnika do rezerwacji dnia konferencji. Jako argument przyjmuje identyfikator uczestnika(ParticipantID), identyfikator rezerwacji dnia konferencji (DayReservationID) oraz numer legitymacji studenckiej jeśli uczestnik jest studentem(StudentCardNumber).

```
CREATE PROCEDURE [dbo].[AddParticipantToConferenceDay]
          @ParticipantID int,
          @DayReservationID int,
          @StudentCardNumber varchar(10)=null
AS
BEGIN
    if not exists (select * from Participants where ParticipantID=@ParticipantID)
          begin
                    raiserror('Participant does not exists',-1,-1)
          end
          else if not exists (select * from DayReservations where DayReservationID=@DayReservationID)
          begin
                    raiserror('Day reservation does not exists',-1,-1)
          end
          else if exists (select * from ReservationLists where DayReservationID=@DayReservationID and
ParticipantID=@ParticipantID)
          begin
                    raiserror('Participant is on the list',-1,-1)
          end
          else
          begin
                    insert into ReservationLists (DayReservationID,ParticipantID,StudentCardNumber)
                    values (@DayReservationID,@ParticipantID,@StudentCardNumber)
          end
END
GO
```

12. AddWorkshopList

Procedura dodająca nowy rekord w tabeli przechowującej listy warsztatów.

```
CREATE PROCEDURE [dbo].[AddWorkshopList]
          @ListID int,
          @WorkshopReservationID int
AS
BEGIN

          SET NOCOUNT ON;
          if not exists(select * from reservationLists where listid = @ListID)
          begin raiserror('Reservation List witch such ID does not exist',-1,-1)
          end
          else if not exists(select * from WorkshopReservations where WorkshopReservationID = @WorkshopReservationID)
          begin raiserror('Workshop Reservation witch such ID does not exist',-1,-1)
          end
          else if exists (select * from WorkshopLists where listID = @ListID)
          begin raiserror('Duplicate',-1,-1)
          end

          else
          insert into WorkshopLists (listid, workshopreservationid)
          values (@ListID,@WorkshopReservationID)

END
GO
```

13. CancelOrder

Procedura anulująca zamówienie o konkretnym identyfikatorze(OrderID).

```
CREATE PROCEDURE [dbo].[CancelOrder]
          @OrderId int
AS
BEGIN
    if not exists (select * from Orders where OrderID=@OrderId)
          begin
                    raiserror('Order does not exists',-1,-1)
          end
          else
          begin
                    update Orders set isCancelled=1 where OrderID=@OrderId
          end
END
GO
```

14. CancelDayReservation

Procedura umożliwiająca anulowanie rezerwacji dnia poprzez ustawienie flagi.

```
CREATE PROCEDURE [dbo].[CancelDayReservation]
          @DayReservationID int
AS
BEGIN
          SET NOCOUNT ON;

    update DayReservations set isCancelled = 1 where DayreservationID = @DayReservationID
END
```

15. CancelWorkshopReservation

Procedura anulująca rezerwację na warsztat o konkretnym identyfikatorze (WorkshopReservationID).

```
CREATE PROCEDURE [dbo].[CancelWorkshopReservation]
          @WorkshopReservationID int
AS
BEGIN
    if not exists (select * from WorkshopReservations where WorkshopReservationID=@WorkshopReservationID)
          begin
                    raiserror('Reservation does not exists',-1,-1)
          end
          else
          begin
                    update WorkshopReservations set isCancelled=1 where WorkshopReservationID=@WorkshopReservationID
          end
END
GO
```

16. CancelOrderAfterWeek

Procedura anulująca zamówienie jeśli nie zostało opłacone w ciągu 7 dni. Należy uruchamiać ją cyklicznie np. raz dziennie.

```
CREATE PROCEDURE [dbo].[CancelOrderAfterWeek]
AS
BEGIN
          update Orders set isCancelled=1
          where OrderID in
                    (select odp.OrderID
                    from OrdersPaymentStatus as odp
                    join Orders as o on o.OrderID=odp.OrderID
                    where odp.Payed < odp.[Order Value] and DATEDIFF(DAY,o.OrderDate,GETDATE())>7)
END
GO
```

17. ChangeCustomerData

Procedura umożliwiająca zmianę danych klienta.

```
CREATE PROCEDURE [dbo].[ChangeCustomerData]
          @CustomerID int,
          @ContactName nvarchar(50),
          @isCompany bit,
          @Phone varchar(15),
          @Email varchar(20)
AS
BEGIN

          SET NOCOUNT ON;
          update Customers set ContactName = @ContactName, isCompany = @isCompany , Phone = @Phone, Email = @Email where
customerID = @CustomerID
END
GO
```

18. ChangeConfDayCapacity

Procedura umożliwiająca zmianę maksymalnej liczby osób na dany dzień konferencji. Jako argument przyjmuje identyfikator dnia(DayID) oraz nowy limit uczestników(NewCapacity)

```
CREATE PROCEDURE [dbo].[ChangeConfDayCapacity]
          @DayID int,
          @NewCapacity int
AS
BEGIN
    declare @CurrentlyBooked int
          set @CurrentlyBooked =  (select isNULL(sum(NumberOfPeople),0)
                                                      from DayReservations as dr
                                                      where dr.DayID=@DayID and dr.isCancelled=0)

          if (@NewCapacity >= @CurrentlyBooked)
          begin
                    update Days set Capacity=@NewCapacity where DayID=@DayID
          end
          else
          begin
                    raiserror('New capacity must be higher or equal currently booked places',-1,-1)
          end
END
GO
```

19. ChangeWorkshopCapacity

Procedura umożliwiająca zmianę pojemności warsztatu podanego jako parametr @WorkshopID na nową wartość @Capacity.

```
CREATE PROCEDURE [dbo].[ChangeWorkshopCapacity]
          @WorkshopID int,
          @Capacity int
AS
BEGIN
          -- SET NOCOUNT ON added to prevent extra result sets from
          -- interfering with SELECT statements.
          SET NOCOUNT ON;
          if @Capacity < (select count(*) from dbo.WorkshopParticipantList(@WorkshopID))
           and @Capacity < (select capacity from workshops where workshopID =@WorkshopID)
           begin raiserror('Number of already signed participants is higher than capacity you want to change', -1,-1)
           end
           else
          update workshops set capacity = @Capacity where workshopID = @WorkshopID
END
```

Funkcje:

1. ConferenceDayPlacesLeft

Funkcja zwraca liczbę dostępnych miejsc w danym dniu konferencji.

```
CREATE FUNCTION [dbo].[ConferenceDayPlacesLeft]
(
          @DayID int
)
RETURNS int
AS
BEGIN
          DECLARE @Result int
          declare @AlreadyTaken int
          set @AlreadyTaken = (select isNUll(sum(NumberOfPeople),0)
                                                      from DayReservations as dr
                                                      where dr.DayID=@DayID and dr.isCancelled=0)
          SELECT @Result = (select d.Capacity from Days as d where d.DayID=@DayID) - @AlreadyTaken
          RETURN @Result
END
GO
```

2. WorkshopPlacesLeft

Zwraca ilość miejsc, jakie zostały na dany warsztat którego ID podajemy jako parametr.

```
CREATE FUNCTION [dbo].[WorkshopPlacesLeft]
(
	@WorkshopID int
)
RETURNS int
AS
BEGIN
	-- Declare the return variable here
	DECLARE  @Result int
	DECLARE @AlreadyTaken int
	set @AlreadyTaken = (select isNULL(sum(NumberOfPeople),0) from WorkshopReservations where WorkshopID = @WorkshopID
and iscancelled = 0)

	SELECT @Result = (select Capacity from workshops where workshopid = @WorkshopID) - @AlreadyTaken
	-- Return the result of the function
	RETURN @Result

END
```

3. ConferenceDayParticipantsList

Funkcja zwraca listę uczestników zapisanych na dany dzień konferencji.

```
CREATE FUNCTION [dbo].[ConferenceDayParticipantsList]
(
	@DayID int
)
RETURNS TABLE
AS
RETURN
(
	SELECT distinct p.ParticipantID,p.FirstName,p.LastName
		from  ReservationLists as rl
		join Participants as p on p.ParticipantID=rl.ParticipantID
		join DayReservations as dr on dr.DayReservationID=rl.DayReservationID
		where dr.DayID = @DayID
)
GO
```

4. WorkshopParticipantsList

Funkcja zwraca tabelę z listą uczestników danego warsztatu.

```
CREATE FUNCTION [dbo].[WorkshopParticipantList]
(
	@WorkshopID int
)
RETURNS TABLE
AS
RETURN
(
	select wr.workshopid, p.firstname, p.lastname, rl.studentcardnumber from participants as p
		inner join reservationlists as rl on p.participantid = rl.participantid
		inner join workshoplists as wl on rl.listid = wl.listid
		inner join workshopreservations as wr on wl.workshopreservationid = wr.WorkshopReservationID
	where wr.workshopid = @WorkshopID
)
END
```

5. ConferenceDays

Funkcja zwraca dni w jakich odbywa się dana konferencja.

```
CREATE FUNCTION [dbo].[ConferenceDays]
(
	@ConferenceID int
)
RETURNS TABLE
AS
RETURN
(
	select d.DayID,d.Date,d.Capacity
	from Days as d
	where d.ConferenceID=@ConferenceID
)
GO
```

6. WorkshopsOfConferenceList

Zwraca tabelę zawierającą listę warsztatów konferencji której ID zostało podane jako parametr.

```
CREATE FUNCTION [dbo].[WorkshopsOfConferenceList]
(
	@ConferenceID int
)
RETURNS TABLE
AS
RETURN
(
	select d.conferenceid, c.name as conferencename ,w.name, d.dayid, d.date, w.starttime, w.endtime, w.price
	from conferences as c
		inner join days as d on c.ConferenceID = d.ConferenceID
		inner join workshops as w on d.DayID = w.DayID
	where c.conferenceID = @ConferenceID
)
END
```

7. ConferencePriceTresholds

Funkcja zwraca progi cenowe danej konferencji

```
CREATE FUNCTION [dbo].[ConferencePriceTresholds]
(
	@ConferenceID int
)
RETURNS TABLE
AS
RETURN
(
	select * from Prices where ConferenceID=@ConferenceID
)
GO
```

8. OrderValue

Funkcja zwraca wartość danego zamówienia.

```
CREATE FUNCTION [dbo].[OrderValue]
(
	@OrderID int
)
RETURNS money
AS
BEGIN
	-- Declare the return variable here
	DECLARE @Result money
	DECLARE @PercentOfBasicPrice float
	DECLARE @DayDifference int
	DECLARE @OrderDate date
	DECLARE @ConferenceID int
	set @ConferenceID = (select ConferenceID from Orders where OrderID=@OrderID)
	set @OrderDate = (select OrderDate from Orders where OrderID=@OrderID)
	DECLARE @ConferenceStartDate date
	set @ConferenceStartDate = (select StartDate from Conferences where ConferenceID= @ConferenceID)

	set @DayDifference = DATEDIFF(DAY, @OrderDate, @ConferenceStartDate)

	set @PercentOfBasicPrice = isNULL((select top 1 PercentOfBasicPrice from
dbo.ConferencePriceTresholds(@ConferenceID) where @DayDifference >= DayDifference and ConferenceID=@ConferenceID order by
DayDifference DESC),1)

	Declare @BasicPrice money
	set @BasicPrice = (select BasicPrice from Conferences where ConferenceID=@ConferenceID)

	Declare @Price money
	set @Price = @BasicPrice * @PercentOfBasicPrice

	Declare @StudentDiscount float
	set @StudentDiscount = (select StudentDiscount from Conferences where ConferenceID=@ConferenceID)

	Declare @StudentPrice money
	set @StudentPrice = @Price * (1-@StudentDiscount)

	Declare @NumberOfPeople int
	set @NumberOfPeople = (select ISnULL(sum(NumberOfPeople),0) from DayReservations where OrderID=@OrderID)

	Declare @NumberOfStudents int
	set @NumberOfStudents = (select ISnull(sum(NumberOfStudents),0) from DayReservations where OrderID=@OrderID)

	Declare @WorkshopsValue money
	set @WorkshopsValue = (select isNULL(sum(wynik.Value),0) from (select wr2.WorkshopReservationID , (select
wr.NumberOfPeople*w.Price
		from WorkshopReservations as wr
		join Workshops as w on w.WorkshopID=wr.WorkshopID
		where wr.isCancelled=0 and wr.WorkshopReservationID=WorkshopReservationID) as 'Value' from
DayReservations
		join WorkshopReservations as wr2 on wr2.DayReservationID=DayReservations.DayReservationID
		where DayReservations.OrderID=@OrderID) as wynik)

	set @Result = @WorkshopsValue + (@NumberOfPeople-@NumberOfStudents)*@Price + @NumberOfStudents*@StudentPrice
	RETURN @Result
END
GO
```

Triggery:

1. CancelOrderTrigger

Triger unieważnia rezerwacje na dni konferencji oraz warsztaty, gdy unieważniamy zamówenie.

```
CREATE TRIGGER [dbo].[CancelOrderTrigger] ON [dbo].[Orders]
          FOR UPDATE
AS
BEGIN
          IF UPDATE(isCancelled)
          begin
                    if (select isCancelled from inserted)=1
                    begin
                              declare @OrderID int
                              set @OrderID = (select OrderID from inserted)

                              update DayReservations set isCancelled=1 where OrderID=@OrderID

                              declare @DayReservationID int
                              set @DayReservationID = (select DayReservationID from DayReservations where OrderID=@OrderID)
                              update WorkshopReservations set isCancelled=1 where DayReservationID= @DayReservationID
                    end
          end
END
GO
```

2. CancelDayReservationTrigger

Triger unieważnia zarezerwowane warsztaty w dniu , w którym unieważniamy rezerwacje na dzien konferencji.

```
CREATE TRIGGER [dbo].[CancelDayReservationTrigger] ON [dbo].[DayReservations]
          FOR UPDATE
AS
BEGIN
  if UPDATE(isCancelled)
  begin
            if (select isCancelled from inserted)=1
            begin
                    declare @DayReservationID int
                    set @DayReservationID = (select DayReservationID from inserted)
                    update WorkshopReservations set isCancelled=1 where DayReservationID=@DayReservationID
                    delete ReservationLists where DayReservationID=@DayReservationID
                    end
  end
END
GO
```

3. UpdateReservationListTrigger

Trigger sprawdza, czy przy robieniu apdejta listy rezerwacji liczba uczestników na liście danej rezerwacji dnia nie przekracza NumberOfPeople oraz czy ilość rekordów z wypełnionym StudentCard nie przekracza wartości NumberOfStudent.

```
CREATE TRIGGER [dbo].[UpdateReservationListTrigger]
   ON  [dbo].[ReservationLists]
   FOR update
AS
BEGIN
                    if(select numberofpeople-peoplefilled from dbo.reservationsfillparticipantstatus
                    where dayreservationid = (select dayreservationid from inserted )) <0
                    begin
                              raiserror ('You are trying to add more people than is reserved',-1,-1)
                              rollback transaction
                              end
                    else if(select numberofstudents-studentsfilled from reservationsfillparticipantstatus
                    where dayreservationid = (select dayreservationid from inserted )) <0
                    begin
                    raiserror ('You are trying to add more students than is reserved',-1,-1)
                              rollback transaction
                    end
END
```

4. InsertReservationListTrigger

To samo co wyżej tylko przy insercie.

```
CREATE TRIGGER [dbo].[InsertReservationListTrigger]
   ON  [dbo].[ReservationLists]
   FOR insert
AS
BEGIN
                    if(select numberofpeople-peoplefilled from reservationsfillparticipantstatus
                    where dayreservationid = (select dayreservationid from inserted )) <0
                    begin
                              raiserror ('You are trying to add more people than is reserved',-1,-1)
                              rollback transaction
                              end
                    else if(select numberofstudents-studentsfilled from reservationsfillparticipantstatus
                    where dayreservationid = (select dayreservationid from inserted )) <0
                    begin
                    raiserror ('You are trying to add more students than is reserved',-1,-1)
                              rollback transaction
                    end
END
```

5. WorkshopsParticipantsLevelTrigger

Triger sprawdza czy nie przekraczamy listy uczestników na daną rezerwację warsztatu.

```
CREATE TRIGGER [dbo].[WorkshopParticipantsLevelTrigger] ON [dbo].[WorkshopLists]
          FOR insert
AS
BEGIN
  declare @WorkshopReservationID int
  set @WorkshopReservationID = (select WorkshopReservationID from inserted)

  declare @ParticipantsAmount int
  set @ParticipantsAmount = (select count(*) from WorkshopLists where WorkshopReservationID=@WorkshopReservationID)

  declare @ParticipantsReserved int
  set @ParticipantsReserved = (select NumberOfPeople from WorkshopReservations where
WorkshopReservationID=@WorkshopReservationID)

  if (@ParticipantsReserved < @ParticipantsAmount )
          begin
          raiserror('Number of participants are higher than reserved number',-1,-1);
          rollback transaction
          end
END
GO
```

6. WorkshopsParticipantsLevelTrigger

Sprawdza czy przy rezerwacji Warsztatu rezerwujemy liczbę miejsc która jest jeszcze dostępna.

```
CREATE TRIGGER [dbo].[WorkshopParticipantsLevelTrigger] ON [dbo].[WorkshopLists]
          FOR insert,update
AS
BEGIN
  declare @WorkshopReservationID int
  set @WorkshopReservationID = (select WorkshopReservationID from inserted)

  declare @ParticipantsAmount int
  set @ParticipantsAmount = (select count(*) from WorkshopLists where WorkshopReservationID=@WorkshopReservationID)

  declare @ParticipantsReserved int
  set @ParticipantsReserved = (select NumberOfPeople from WorkshopReservations where
WorkshopReservationID=@WorkshopReservationID)

  if (@ParticipantsReserved < @ParticipantsAmount )
          begin
          raiserror('Number of participants are higher than reserved number',-1,-1);
          rollback transaction
          end
END
```

Indeksy:

Indeksy w bazie danych zostały utworzone na wszystkich kluczach obcych w każdej tabeli. Umożliwia to przyspieszenie operacji wstawiania, która jest dominująca w naszym projekcie.

Role:

Administrator - osoba odpowiedzialna za całość działania systemu, w związku z czym ma dostęp do wszystkich składowych systemu, a także jest w stanie nadawać innym uprawnienia do wyświetlania odpowiednich widoków, używanie odpowiednich funkcji etc. Jest również w stanie dokonywać modyfikacji już istniejących funkcjonalności oraz dodawać nowe rozszerzenia.

Menedzer ds konferencji- osoba odpowiedzialna za dodawanie konferencji oraz zarządzanie rezerwacjami. Ma dostęp do wszystkich składowych.

Księgowy- osoba odpowiedzialna za zarządzanie płatnościami konferencji. Ma dostęp do tabeli Payments, widoku OrdersNotPayedYet, OrdersPaymentStatus oraz OrdersOverPaid.

Obsługa konferencji - osoba weryfikująca czy dana osoba może wejść na konferencję/warsztat. Posiada uprawnienia do widoku ParticipantsOfFutureConferences, WorkshopParticipantsList, funkcji ConferenceDayParticipantsList oraz WorkshopParticipantList.