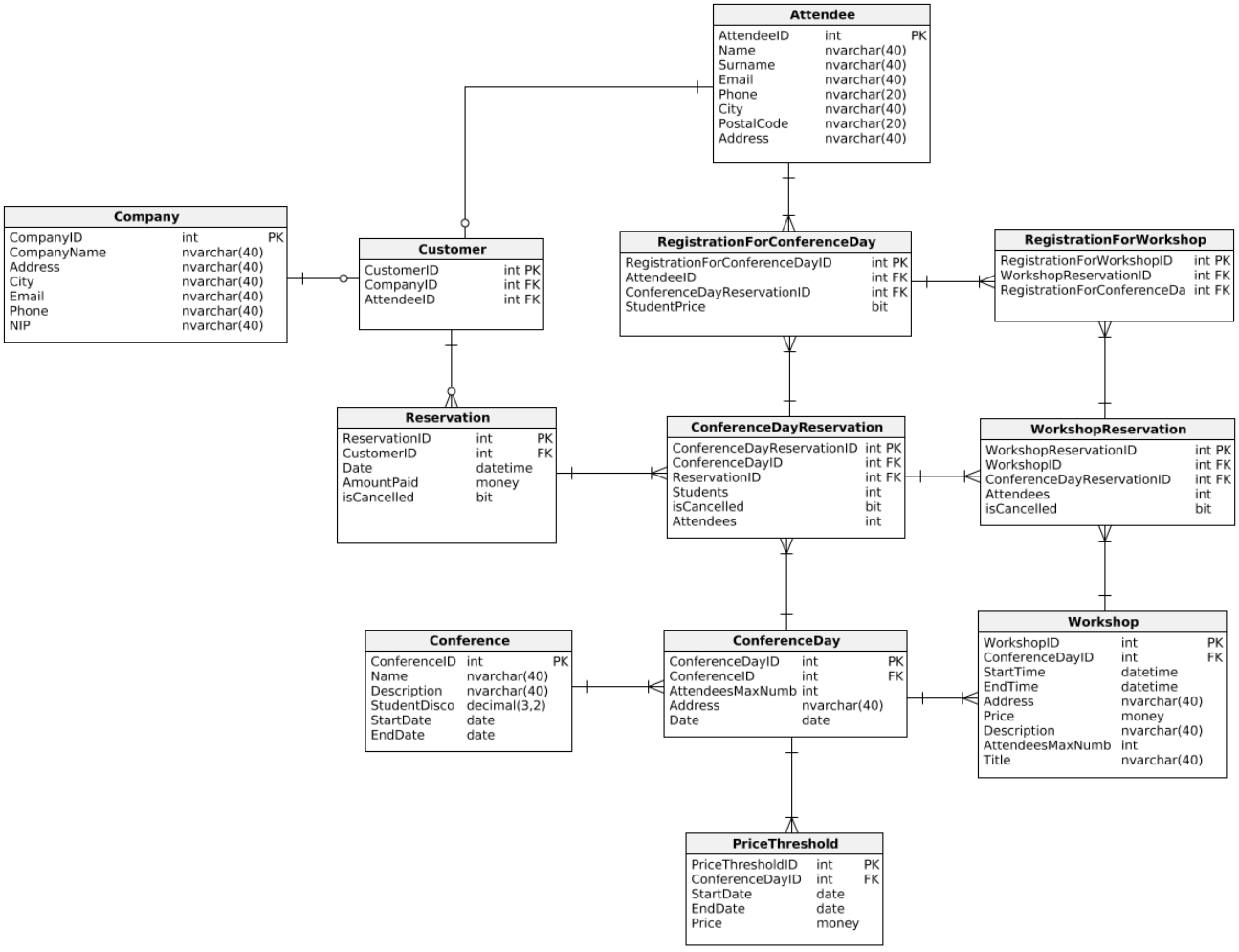
Podstawy baz danych

Projekt systemu zarządzania konferencjami

Gabriel Kępka, Andrii Trishch

1. Diagram bazy danych



2. Tabele
    1. Tabela Company

Zawiera informację nt. klientów firmowych takie jak nazwa firmy, jej adres, nr telefonu itp.

- CompanyID - Identyfikator kompanji
- CompanyName – Nazwa kompanji
- Address- Adres firmy
- City-Miasto firmy
- Email-e-mail firmy
- Phone-Numer firmy
- NIP-NIP firmy

```
CREATE TABLE Company (
    CompanyID int IDENTITY(1,1) NOT NULL,
    CompanyName nvarchar(40)  NOT NULL,
    Address nvarchar(40)  NOT NULL,
    City nvarchar(40)  NOT NULL,
    Email nvarchar(40)  NOT NULL,
    Phone nvarchar(40)  NOT NULL,
    NIP nvarchar(40)  NOT NULL,
    CONSTRAINT Company_pk PRIMARY KEY  (CompanyID)
);
                );
```

1. Tabela Customer

Określa, czy klient jest klientem indywidualnym (pole CompanyID ma wartość NULL), czy firmowym (pole AttendeeID ma wartość NULL)

- CustomerID-identyfikator klienta
- CompanyID-identyfikator firmy ktora jest klientem
- AttendeeID-identyfikator uczestnika tkory jest klientem

```
CREATE TABLE Customer (
    CustomerID int IDENTITY(1,1) NOT NULL,
    CompanyID int,
    AttendeeID int,
    CONSTRAINT Customer_pk PRIMARY KEY  (CustomerID)
);
ALTER TABLE Customer ADD CONSTRAINT Customer_Attendee
    FOREIGN KEY (AttendeeID)
    REFERENCES Attendee (AttendeeID);
ALTER TABLE Customer ADD CONSTRAINT Customer_Company
    FOREIGN KEY (CompanyID)
    REFERENCES Company (CompanyID);
ALTER TABLE Customer WITH CHECK ADD CONSTRAINT AttendeeIDOrCompanyIDEqualsNULL
    CHECK ((AttendeeID IS NULL OR CompanyID IS NULL ) AND NOT (AttendeeID IS NULL
AND CompanyID IS NULL))
ALTER TABLE Customer CHECK CONSTRAINT AttendeeIDOrCompanyIDEqualsNULL
```

1. Tabela Reservation

Zawiera informacje nt rezerwacji – ID klienta, datę dokonania rezerwacji, dotychczas wpłacone pieniądze oraz informację o anulowaniu

- ReservationID-identyfikator rezerwacji
- CustomerID-number identyfikacyjny klienta któremu należy dana rezerwacja
- AmountPaid-ilość pieniędzy którę zostali zaplacone
- isCancelled-identyfikuje czy dana rezerwacja została odwołana

```
xCREATE TABLE Reservation (
    ReservationID int IDENTITY(1,1) NOT NULL,
    CustomerID int  NOT NULL,
    Date datetime  NOT NULL,
    AmountPaid money  NOT NULL,
    isCancelled bit  NOT NULL,
    CONSTRAINT Reservation_pk PRIMARY KEY  (ReservationID)
);
```

```
ALTER TABLE Reservation ADD CONSTRAINT Reservation_Customer
    FOREIGN KEY (CustomerID)
    REFERENCES Customer (CustomerID);
```

1. Tabela ConferenceDayReservation

Zawiera informacje o rezerwacji na dany dzień konferencji – ilość zarezerwowanych miejsc łącznie (pole Attendees), ilość zarezerwowanych miejsc studenckich (pole Students), informację o anulowaniu

- ConferenceDayReservationID-identyfikator dnia rezerwacji
- ConferenceDayID-identykator dnia konferencji
- Reservation -identyfikator rezerwacji do jakiej należy dany rezerwowany dzień
- Attendees-liczba miejjc zarezerwowanych
- Students -liczba miejsc zarezerwowanych dla studentów
- IsCancelled-identyfikuje czy rezerwacja na dany dzień została odwołana

```
CREATE TABLE ConferenceDayReservation (
    ConferenceDayReservationID int IDENTITY(1,1) NOT NULL,
    ConferenceDayID int  NOT NULL,
    ReservationID int  NOT NULL,
    Attendees int NOT NULL,
    Students int  NOT NULL,
    isCancelled bit  NOT NULL,
    CONSTRAINT ConferenceDayReservation_pk PRIMARY KEY
(ConferenceDayReservationID)
);
ALTER TABLE ConferenceDayReservation WITH CHECK ADD CONSTRAINT AttendeesOver0
    CHECK (Attendees > 0)
ALTER TABLE ConferenceDayReservation CHECK CONSTRAINT AttendeesOver0
ALTER TABLE ConferenceDayReservation WITH CHECK ADD CONSTRAINT
StudentsEqualOrOver0AndLessOrEqualAttendees
    CHECK (Students >= 0 AND  Students <= Attendees)
ALTER TABLE ConferenceDayReservation CHECK CONSTRAINT
StudentsEqualOrOver0AndLessOrEqualAttendees
ALTER TABLE ConferenceDayReservation ADD CONSTRAINT
ConferenceDayReservation_ConferenceDay
    FOREIGN KEY (ConferenceDayID)
    REFERENCES ConferenceDay (ConferenceDayID);
ALTER TABLE ConferenceDayReservation ADD CONSTRAINT
ConferenceDayReservation_Reservation
    FOREIGN KEY (ReservationID)
    REFERENCES Reservation (ReservationID);
```

1. Tabela WorkshopReservation

Zawiera informacje o rezerwacji miejsc na warsztaty – liczbę zarezerwowanych miejsc, informację o anulowaniu rezerwacji

- WorkshopReservationID-identyfikator rezerwacji
- WorkshopID-identyfikator warsztatu
- ConferenceDayReservationID-identyfikator dni rezerwacji
- Attendees-liczba rezerwowanych miejsc
- isCancelled-identyfikuje czy rezerwacja na warsztat jest odwołana

```
CREATE TABLE WorkshopReservation (
    WorkshopReservationID int IDENTITY(1,1) NOT NULL,
```

```
    WorkshopID int  NOT NULL,
    ConferenceDayReservationID int  NOT NULL,
    Attendees int  NOT NULL,
    isCancelled bit  NOT NULL,
    CONSTRAINT WorkshopReservation_pk PRIMARY KEY  (WorkshopReservationID)
);
ALTER TABLE WorkshopReservation ADD CONSTRAINT
WorkshopReservation_ConferenceDayReservation
    FOREIGN KEY (ConferenceDayReservationID)
    REFERENCES ConferenceDayReservation (ConferenceDayReservationID);
ALTER TABLE WorkshopReservation ADD CONSTRAINT WorkshopReservation_Workshop
    FOREIGN KEY (WorkshopID)
    REFERENCES Workshop (WorkshopID);
ALTER TABLE WorkshopReservation WITH CHECK ADD CONSTRAINT AttendeesAboveZero
    CHECK (Attendees > 0)
ALTER TABLE WorkshopReservation CHECK CONSTRAINT AttendeesAboveZero
```

1. Tabela ConferenceDay

Zawiera informacje o dniu konferencji – łączną liczbę miejsc, adres, data

- ConferenceDayID-identyfikator dnia konferencji
- ConferenceID-identyfikator konferencji
- AttendeeMaxNumber-ilość dotępnych miejsc konferencji
- Address-miejsce prowadzenia konferencji
- Date-data konferencji

```
CREATE TABLE ConferenceDay (
    ConferenceDayID int IDENTITY(1,1) NOT NULL,
    ConferenceID int  NOT NULL,
    AttendeesMaxNumber int  NOT NULL,
    Address nvarchar(40)  NOT NULL,
    Date date  NOT NULL,
    CONSTRAINT ConferenceDay_pk PRIMARY KEY  (ConferenceDayID)
);
ALTER TABLE ConferenceDay ADD CONSTRAINT ConferenceDay_Conference
    FOREIGN KEY (ConferenceID)
    REFERENCES Conference (ConferenceID);
ALTER TABLE ConferenceDay WITH CHECK ADD CONSTRAINT AttendeesMaxNumberOver0
    CHECK (AttendeesMaxNumber > 0)
ALTER TABLE ConferenceDay CHECK CONSTRAINT AttendeesMaxNumberOver0
```

1. Tabela Workshop

Zawiera informacje o warsztatach – czas początku i końca, adres, cena za udział, opis, łączna ilość miejsc, temat

- WorkshopID-identyfikator warsztatu
- ConferenceDayId-identyfikator dnia konferencji
- StartTime-czas początku warsztatu
- Endtime-czas końca warsztatu
- Address-adres prowadzenia warsztatu
- Price -cena warsztatu
- Description-opis warsztatu
- AttendeesMaxNumber-ilość dostępnych miejsc warsztatu

- Title-nazwa warsztatu

```
CREATE TABLE Workshop (
    WorkshopID int IDENTITY(1,1) NOT NULL,
    ConferenceDayID int  NOT NULL,
    StartTime datetime  NOT NULL,
    EndTime datetime  NOT NULL,
    Address nvarchar(40)  NOT NULL,
    Price money  NOT NULL,
    Description nvarchar(40)  NOT NULL,
    AttendeesMaxNumber int  NOT NULL,
    Title nvarchar(40) NOT NULL,
    CONSTRAINT Workshop_pk PRIMARY KEY  (WorkshopID)
);
ALTER TABLE Workshop ADD CONSTRAINT Workshop_ConferenceDay
    FOREIGN KEY (ConferenceDayID)
    REFERENCES ConferenceDay (ConferenceDayID);
ALTER TABLE Workshop WITH CHECK ADD CONSTRAINT StartTimeBeforeEndTime
    CHECK (EndTime >= StartTime)
ALTER TABLE Workshop CHECK CONSTRAINT StartTimeBeforeEndTime
ALTER TABLE Workshop WITH CHECK ADD CONSTRAINT PriceNotBelowZero
    CHECK (Price >= 0)
ALTER TABLE Workshop CHECK CONSTRAINT PriceNotBelowZero
ALTER TABLE Workshop WITH CHECK ADD CONSTRAINT AttendeesMaxNumberAboveZero
    CHECK (AttendeesMaxNumber > 0)
ALTER TABLE Workshop CHECK CONSTRAINT AttendeesMaxNumberAboveZero
```

1. Tabela Conference

Zawiera informacje nt konferencji – nazwę, opis, wartość zniżki dla studentów, datę początku i końca

- ConferenceID-identyfikator konferencji
- Name-nazwa konferencji
- Description-opis konferencji
- StudentDiscount-rabat studencki konferencji
- StartDate-data początku konferencji
- EndDate-data końca konferencji

```
CREATE TABLE Conference (
    ConferenceID int IDENTITY(1,1) NOT NULL,
    Name nvarchar(40)  NOT NULL,
    Description nvarchar(40)  NOT NULL,
    StudentDiscount decimal(3,2)  NOT NULL,
    StartDate date NOT NULL,
    EndDate date NOT NULL,
    CONSTRAINT Conference_pk PRIMARY KEY  (ConferenceID)
);
ALTER TABLE Conference WITH CHECK ADD CONSTRAINT StudentDiscountBetween0and1
    CHECK (StudentDiscount <= 1 AND StudentDiscount >= 0)
ALTER TABLE Conference CHECK CONSTRAINT StudentDiscountBetween0and1
ALTER TABLE Conference WITH CHECK ADD CONSTRAINT NameLengthOver0
    CHECK (LEN(Name) > 0)
ALTER TABLE Conference CHECK CONSTRAINT NameLengthOver0
ALTER TABLE Conference WITH CHECK ADD CONSTRAINT DescriptionLengthOver0
    CHECK (LEN(Description) > 0)
ALTER TABLE Conference CHECK CONSTRAINT DescriptionLengthOver0
```

1. Tabela PriceThreshold

Zawiera listę progów cenowych powiązanych z danym dniem konferencji, zawiera cenę, datę początku obowiązywania progu i końca

```
CREATE TABLE PriceThreshold (
    PriceThresholdID int IDENTITY(1,1) NOT NULL,
    ConferenceDayID int  NOT NULL,
    StartDate date  NOT NULL,
    EndDate date NOT NULL,
    Price money  NOT NULL,
    CONSTRAINT PriceThreshold_pk PRIMARY KEY  (PriceThresholdID)
);
ALTER TABLE PriceThreshold ADD CONSTRAINT PriceThreshold_ConferenceDay
    FOREIGN KEY (ConferenceDayID)
    REFERENCES ConferenceDay (ConferenceDayID);
ALTER TABLE PriceThreshold WITH CHECK ADD CONSTRAINT StartDateBeforeEndDate
    CHECK (EndDate > StartDate)
ALTER TABLE PriceThreshold CHECK CONSTRAINT StartDateBeforeEndDate
ALTER TABLE PriceThreshold WITH CHECK ADD CONSTRAINT PriceNotBelowZero
    CHECK (Price >= 0)
ALTER TABLE PriceThreshold CHECK CONSTRAINT PriceNotBelowZero
```

2. Tabela RegistrationForConferenceDay

Zawiera listę rejestracji uczestników na dzień konferencji wraz z określeniem, czy obowiązuje cena studencka

```
CREATE TABLE RegistrationForConferenceDay (
    RegistrationForConferenceDayID int IDENTITY(1,1) NOT NULL,
    AttendeeID int  NOT NULL,
    ConferenceDayReservationID int  NOT NULL,
    StudentPrice bit NOT NULL,
    CONSTRAINT RegistrationForConferenceDay_pk PRIMARY KEY
(RegistrationForConferenceDayID)
);
ALTER TABLE RegistrationForConferenceDay ADD CONSTRAINT
RegistrationForConferenceDay_Attendee
    FOREIGN KEY (AttendeeID)
    REFERENCES Attendee (AttendeeID);
ALTER TABLE RegistrationForConferenceDay ADD CONSTRAINT
RegistrationForConferenceDay_ConferenceDayReservation
    FOREIGN KEY (ConferenceDayReservationID)
    REFERENCES ConferenceDayReservation (ConferenceDayReservationID);
```

3. Tabela RegistrationForWorkshop

Zawiera informacje o rejestracji na warsztat – każda rejestracja powiązana jest z odpowiednią rejestracją na dzień konferencji

```
CREATE TABLE RegistrationForWorkshop (
    RegistrationForWorkshopID int IDENTITY(1,1) NOT NULL,
    WorkshopReservationID int  NOT NULL,
    RegistrationForConferenceDayID int  NOT NULL,
    CONSTRAINT RegistrationForWorkshop_pk PRIMARY KEY  (RegistrationForWorkshopID)
);
ALTER TABLE RegistrationForWorkshop ADD CONSTRAINT
RegistrationForWorkshop_RegistrationForConferenceDay
    FOREIGN KEY (RegistrationForConferenceDayID)
```

```
    REFERENCES RegistrationForConferenceDay (RegistrationForConferenceDayID);
ALTER TABLE RegistrationForWorkshop ADD CONSTRAINT
RegistrationForWorkshop_WorkshopReservation
    FOREIGN KEY (WorkshopReservationID)
    REFERENCES WorkshopReservation (WorkshopReservationID);
```

4. Tabela Attendee

Zawiera informacje o uczestnikach konferencji – imię i nazwisko, dane adresowe, kontaktowe

```
CREATE TABLE Attendee (
    AttendeeID int IDENTITY(1,1) NOT NULL,
    Name nvarchar(40)  NOT NULL,
    Surname nvarchar(40)  NOT NULL,
    Email nvarchar(40)  NOT NULL,
    Phone nvarchar(20)  NOT NULL,
    City nvarchar(40)  NOT NULL,
    PostalCode nvarchar(20)  NOT NULL,
    Address nvarchar(40)  NOT NULL,
    CONSTRAINT AttendeeID PRIMARY KEY  (AttendeeID)
);
```

2. Widoki
    1. Widok MostPopularConferences

Zawiera listę 10 najpopularniejszych konferencji, uszeregowanych wg liczby uczestników

```
create view MostPopularConferences
as
    select top 10
    c.ConferenceID, c.name,sum(cdr.Attendees) as [Attendees Number] from
Conference c
inner join ConferenceDay cd on cd.ConferenceID=c.ConferenceID
inner join ConferenceDayReservation cdr on cdr.ConferenceDayID=cd.ConferenceDayID
group by c.ConferenceID, c.name
order by sum(cdr.Attendees) desc
```

2. Widok MostPopularWorkshops

Zawiera listę 10 najbardziej popularnych warsztatów, uszeregowanych wg liczby uczestników

```
create view MostPopularWorkshops
as
    select top 10
    w.WorkshopID, w.title ,sum(wr.Attendees) [Attendees Number] from Workshop w
inner join WorkshopReservation wr on wr.WorkshopID=w.WorkshopID
group by w.title, w.WorkshopID
order by sum(wr.Attendees) desc
```

3. Widok FutureConferences

Zawiera listę konferencji, które odbędą się w przyszłości

```
create view FutureConferences
as
    select c.ConferenceID, c.name,c.startdate ,c.Description
from Conference c
where c.StartDate>GETDATE()
```

4. Widok ReservationsWithPrices

Zawiera listę rezerwacji wraz z łączną potrzebną do zapłacenia kwotą

```
create view dbo.ReservationsWithPrices
as
select c.name,r.reservationid,r.date,
[dbo].get_price_by_reservation_id(r.ReservationID) as [Amount to
Pay],r.isCancelled from Reservation r
left join ConferenceDayReservation cdr on cdr.reservationID=r.ReservationID
left join ConferenceDay cd on cdr.ConferenceDayID=cd.ConferenceDayID
left join Conference c on cd.ConferenceID=c.ConferenceID
```

5. Widok CompaniesWithReservationsWithoutFullData
   Zawiera listę firm posiadających rezerwacje z niepełnymi danymi oraz identyfikatory tych rezerwacji wraz z datą ich dokonania

```
create view CompaniesWithReservationsWithoutFullData
as
    select co.CompanyID,co.CompanyName,co.Phone,co.Email,r.date as [Reservation
Date],r.ReservationID as [Reservation ID] from company co
inner join customer cu on cu.CompanyID=co.CompanyID
inner join reservation r on r.CustomerID=cu.CustomerID
inner join ConferenceDayReservation cdr on cdr.ReservationID=r.ReservationID
inner join RegistrationForConferenceDay rfcd on
rfcd.ConferenceDayReservationID=cdr.ConferenceDayReservationID
inner join WorkshopReservation WR on cdr.ConferenceDayReservationID =
WR.ConferenceDayReservationID
inner join RegistrationForWorkshop rfw on
rfw.WorkshopReservationID=wr.WorkshopReservationID

where ((select sum(cdr2.attendees) from ConferenceDayReservation cdr2 where
cdr2.ReservationID=r.ReservationID)
    >(select count(rfcd2.RegistrationForConferenceDayID)  from
RegistrationForConferenceDay rfcd2
        inner join ConferenceDayReservation cdr2 on
rfcd2.ConferenceDayReservationID = cdr2.ConferenceDayReservationID
        inner join Reservation r2 on cdr2.ReservationID = r2.ReservationID
        where cu.CustomerID=r2.CustomerID and r2.ReservationID = r.ReservationID)
    )
or ((select sum(wsr.attendees) from WorkshopReservation wsr
    inner join ConferenceDayReservation cdr3 on
wsr.ConferenceDayReservationID=cdr3.ConferenceDayReservationID
    where cdr3.ReservationID=r.ReservationID)
        > (select count(rfw2.RegistrationForWorkshopID) from
RegistrationForWorkshop rfw2
            inner join WorkshopReservation W2 on rfw2.WorkshopReservationID =
W2.WorkshopReservationID
            inner join ConferenceDayReservation C on W2.Attendees = C.Attendees
            inner join Reservation R3 on C.ReservationID = R3.ReservationID
            where R3.CustomerID=cu.CustomerID and r.ReservationID =
r3.ReservationID
        ))
```

6. Widok ReservationsForConferencesThatStartinTwoWeeksWithoutFullData

Zawiera listę jak w widoku powyżej, z tym, że wyświetlone są tylko wpisy dot. konferencji mających się zacząć za mniej niż 2 tygodnie

```
create view ReservationsForConferencesThatStartInTwoWeeksWithoutFullData
as
    select CompanyID, CompanyName, Phone, Email, [Reservation Date], [Reservation
ID] from CompaniesWithReservationsWithoutFullData r
inner join ConferenceDayReservation cdr on cdr.ReservationID=r.[Reservation ID]
inner join ConferenceDay cd on cd.ConferenceDayID=cdr.ConferenceDayID
inner join Conference c on c.ConferenceID=cd.ConferenceID
where dateadd(week,+2,GETDATE())>c.StartDate and c.StartDate > GETDATE()
```

7. Widok MostActiveCustomers

Zawiera listę 10 najbardziej aktywnych klientów, uszeregowaną wg liczby zarezerwowanych przez nich miejsc

```
create view MostActiveCustomers
as
    select top 10
co.companyname,(select count (*) from Attendee a2 where a2.AttendeeID in (select
AttendeeID from
                Customer c2 inner join
                Reservation R2 on c2.CustomerID = R2.CustomerID
                inner join ConferenceDayReservation CDR on R2.ReservationID =
CDR.ReservationID
                inner join RegistrationForConferenceDay RFCD on
CDR.ConferenceDayReservationID = RFCD.ConferenceDayReservationID
                inner join Attendee A3 on RFCD.AttendeeID = A3.AttendeeID
                where c2.customerid = c.CustomerID )) as [Attendee Number]
from company co
inner join customer c on c.CompanyID=co.CompanyID
inner join Attendee a on a.AttendeeID=c.AttendeeID
group by co.companyname, c.CustomerID
order by [Attendee Number] desc
```

8. Widok MonthIncome

Zawiera listę uzyskanych zysków wg miesięcy

```
create view MonthIncome
as
    select month(r.date) as Month,r.AmountPaid
from reservation r
group by month(r.date),r.amountpaid
```

9. Widok FutureConferencesAttendees

Wyświetla listę uczestników przyszłych konferencji wraz z danymi na ich temat

```
create view FutureConferencesAttendees
as
    select distinct A.AttendeeID, A.Name, A.Surname, A.Email, A.Phone, A.City,
A.PostalCode, A.Address
        from Conference C inner join ConferenceDay CD on C.ConferenceID =
CD.ConferenceID
        inner join ConferenceDayReservation CDR on CD.ConferenceDayID =
CDR.ConferenceDayID
        inner join RegistrationForConferenceDay RFCD on
CDR.ConferenceDayReservationID = RFCD.ConferenceDayReservationID
```

```
        inner join Attendee A on RFCD.AttendeeID = A.AttendeeID
        where C.StartDate >= GETDATE()
```

10. Widok OverlappingWorkshops

Wyświetla pary nachodzących się warsztatów

```
create view OverlappingWorkshops
as
    select W1.WorkshopID as FirstWorkshopID, W1.Title as FirstWorkshopTitle,
W1.StartTime as FirstWorkshopStartTime, W1.EndTime as FirstWorkshopEndTime,
W2.WorkshopID, W2.Title, W2.StartTime, W2.EndTime
    from Workshop W1 cross join Workshop W2
    where [dbo].are_workshops_overlapping(W1.WorkshopID, W2.WorkshopID) = 1
```

11. Widok OverlappingConferenceDays

Wyświetla pary nachodzących się dni konferencji

```
create view OverlappingConferenceDays
as
    select C1.ConferenceDayID as C1ConferenceDayID, C1.Date as C1Date, C1.Address
as C1Address, C1.AttendeesMaxNumber as C1AttendeesMaxNumber, C2.ConferenceDayID,
C2.Date, C2.Address, C2.AttendeesMaxNumber
    from ConferenceDay C1 cross join ConferenceDay C2
    where C1.Date = C2.Date
    and C1.ConferenceDayID != C2.ConferenceDayID
```

12. Widok OverlappingConferences

Wyświetla pary nachodzących się konferencji

```
create view OverlappingConferences

as
    select C1.ConferenceID as C1ConferenceID, C1.Name as C1Name, C1.StartDate as
C1StartDate, C1.EndDate as C1EndDate,C2.ConferenceID, C2.Name, C2.StartDate,
C2.EndDate
    from Conference C1 cross join Conference C2
    where [dbo].are_conferences_overlapping(C1.ConferenceID, C2.ConferenceID) = 1
```

13. Widok UndergoingConferences

Wyświetla listę aktualnie odbywających się konferencji

```
create view UndergoingConferences
as
    select ConferenceID, Name, StartDate, EndDate
    from Conference
    where Conference.StartDate <= GETDATE() and Conference.EndDate >= GETDATE()
```

14. Widok CancelledReservationsWithCustomerInfo

Wyświetla listę anulowanych rezerwacji wraz z informacjami nt klientów, którzy ich dokonali

```
create view CancelledReservationsWithCustomerInfo
as
```

```
    select ReservationID, R.CustomerID, Date, AmountPaid, CompanyName, Address,
City, Email, Phone
    from Reservation R inner join Customer C on R.CustomerID = C.CustomerID
    inner join Company C2 on C.CompanyID = C2.CompanyID
    where R.isCancelled = 1
    union
    select ReservationID, R.CustomerID, Date, AmountPaid, Name + Surname as name,
Address, City, Email, Phone
    from Reservation R inner join Customer C3 on R.CustomerID = C3.CustomerID
    inner join Attendee A on C3.AttendeeID = A.AttendeeID
    where R.isCancelled = 1
```

15. Widok FutureWorkshopsWithFreePlaces

Zawiera listę przyszłych warsztatów wraz z liczbą wolnych miejsc

```
create view FutureWorkshopsWithFreePlaces
as
    select WorkshopID, StartTime, EndTime, Address, Price, Title,
AttendeesMaxNumber - (select count(*) from
dbo.get_attendees_by_workshopid(WorkshopID)) as [Free places]
    from Workshop
    where AttendeesMaxNumber > (select count(*) from
get_attendees_by_workshopid(WorkshopID))
```

16. Widok FutureConferenceDaysWithFreePlaces

Zawiera listę przyszłych dni konferencji wraz z liczbą wolnych miejsc

```
create view FutureConferenceDaysWithFreePlaces
as
    select ConferenceDayID, ConferenceID, Address, Date, AttendeesMaxNumber -
(select count(*) from  dbo.get_attendees_by_conferencedayid(ConferenceDayID)) as
[Free places]
    from ConferenceDay
    where AttendeesMaxNumber > (select count(*) from
dbo.get_attendees_by_conferencedayid(ConferenceDayID))
```

3. Procedury
    1. Procedura add_conference

Dodaje konferencję

```
create procedure add_conference
 @name nvarchar(40),
 @description nvarchar(40),
 @StudentDiscount decimal(3,2),
 @StartDate date,
 @EndDate date
 as begin try
    insert into conference
     (Name, Description, StudentDiscount, StartDate, EndDate)
     values (@name,@description,@StudentDiscount,@StartDate,@EndDate)
end try
```

```
begin catch
    declare @errorMessage nvarchar(2048)
    set @errorMessage = 'Error occurred when adding conference: \n'
    +ERROR_MESSAGE();
    THROW 52000, @errormessage,1
end catch
```

2. Procedura add_conference_day

Dodaje dzień koferencji

```
create procedure add_conference_day
@conferenceID int,
@attendanceMaxNumber int,
@address nvarchar(40),
@date datetime
 as begin try
    if not exists
     (select * from conference
        where conferenceid=@conferenceID)
     begin
         throw 52000, 'Conference with provided id does not exist.',1
     end
     if exists
         (select * from ConferenceDay
         where ConferenceID=@conferenceID and date=@date)
     begin
         throw 52000,'Conference day with provided date already exists.',1
     end
     insert into ConferenceDay(conferenceid, attendeesmaxnumber, address, date)
     values(@conferenceID, @attendanceMaxNumber,@address,@date)
end try
begin catch
    declare @errorMessage nvarchar(2048)
        set @errorMessage = 'Error occurred when adding conference day: \n'
+ERROR_MESSAGE();
   THROW 52000, @errormessage,1
end catch
```

3. Procedura add_workshop

Dodaje warsztat

```
create procedure add_workshop
@conferenceDayID int,
@startTime datetime,
@endTime datetime,
@address nvarchar(40),
@price money,
@description nvarchar(40),
@attendanceMaxNumber int,
@title nvarchar(40)
as
    begin try
        if not exists(
            select * from ConferenceDayReservation
            where ConferenceDayID=@conferenceDayID
            )
```

```
        begin
            throw 52000,'Conference day does not exist' ,1
        end
        insert into Workshop
        (conferencedayid, starttime, endtime, address, price, description,
attendeesmaxnumber, title)

values(@conferenceDayID,@startTime,@endTime,@address,@price,@description,@attendan
ceMaxNumber,@title)
    end try
    begin catch
          declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when adding workshop: \n'
+ERROR_MESSAGE();
    THROW 52000, @errormessage,1
    end catch
```

4. Procedura add_price_threshold

Dodaje progi cenowe.

```
create procedure add_price_threshold
        @StartDate datetime,
        @EndDate datetime,
        @price money,
        @conferenceDayID int
as
    begin try
        if not exists(select * from ConferenceDay where
ConferenceDayID=@conferenceDayID)
        begin
            throw 52000, 'Conference day does not exists',1
        end
        insert into PriceThreshold
        (ConferenceDayID, StartDate, EndDate, Price)
        Values(@conferenceDayID,@StartDate,@EndDate,@price)
    end try
    begin catch
        declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when adding price threshold: \n'
+ERROR_MESSAGE();
    THROW 52000, @errormessage,1
    end catch
```

5. Procedura add_reservation_for_conference

Dodaje rezerwacje na conferencje

```
create procedure add_reservation_for_conference
    @ConferenceID int,
    @CustomerID int,
    @Date datetime
as
    begin try
        if not exists (select * from Conference where conferenceID=@ConferenceID)
        begin throw 52000 , 'Conference does not exists .' ,1
        end
        if not exists(select * from Customer  where CustomerID=@CustomerID)
        begin throw 52000 , 'Customer does not exists .' ,1
        end
        insert into Reservation
```

```
        (CustomerID, Date, AmountPaid, isCancelled)
        Values (@CustomerID,@date, 0, 0)
    end try
    begin catch
        declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when adding conference: \n'
+ERROR_MESSAGE();
    THROW 52000, @errormessage,1
    end catch
```

6. Procedura add_conference_day_reservation

Dodaje rezerwacje na dzień konferecji

```
create procedure add_conference_day_reservation
@ConferenceDayID int,
@ReservationID int,
@Students int,
@Attendees int
as
begin try
    if not exists(select * from conferenceDay where
conferenceDayID=@ConferenceDayID)
    begin throw 52000,'Conference day does not exists .' ,1
    end
    if not exists(select * from reservation where ReservationID=@ReservationID)
    begin throw 52000,'Reservation does not exists .',1
    end
   if not exists(select c.companyID from customer c
    inner join Reservation r on r.CustomerID=c.CustomerID
        where r.ReservationID = @ReservationID and CompanyID is not null)
    and @Attendees > 1
    begin
    ;throw 52000,'Individual client can book only single place in a day',1
    end
    insert into ConferenceDayReservation(conferencedayid, reservationid,
attendees, students,isCancelled)
    values(@ConferenceDayID,@ReservationID,@Attendees,@Students, 0)
end try
begin catch
    declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when adding conference day
reservation: \n' +ERROR_MESSAGE();
    THROW 52000, @errormessage,1
end catch
```

7. Procedura add_workshop_reservation

Dodaje reserwacje na warsztat

```
create procedure add_workshop_reservation
@WorkshopID int,
@attendees int,
@conferenceDayReservationID int
as
        begin try
            if not exists (select * from workshop wr where
wr.WorkshopID=@WorkshopID)
            begin throw 52000,'Workshop does not exists',1
                end
            if not exists (select * from ConferenceDayReservation where
```

```
ConferenceDayReservationID=@conferenceDayReservationID)
            begin throw 52000,'Reservation does not exists',1
                end
            if exists (select isCancelled from ConferenceDayReservation where
ConferenceDayReservationID=@conferenceDayReservationID
                and ConferenceDayReservation.isCancelled = 1)
                begin throw 52000,'Reservation for this day of conference is
already cancelled',1
                end
            declare @possibleAttendees int =
            (select Attendees from ConferenceDayReservation where
@conferenceDayReservationID=ConferenceDayReservationID)
            if @possibleAttendees < @attendees
            begin throw 52000,'Not enough place for all',1
                end
            insert into WorkshopReservation
            ( WorkshopID, ConferenceDayReservationID, Attendees, isCancelled)
            Values (@WorkshopID,@conferenceDayReservationID,@attendees, 0)
        end try
        begin catch
               declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when adding reservation workshop: \n'
+ERROR_MESSAGE();
    THROW 52000, @errormessage,1
        end catch
```

8. Procedura add_company

Dodaje firme

```
create procedure add_company
@CompanyName nvarchar(40),
@CompanyAddress nvarchar(40),
@City nvarchar(40),
@email nvarchar(40),
@phone nvarchar(40),
@NIP nvarchar(40)
as
begin try
    declare @companyID int
   select @companyID=co.companyID from customer
    inner join company co on co.CompanyID=customer.CompanyID
    where @email=co.Email and @nip=co.NIP
    if @companyId is not NULL
        begin throw 52000,'Company already exists',1
    end
    insert into Company
    (CompanyName, Address, City, Email, Phone, NIP)
    values(@companyname,@CompanyAddress,@City,@email,@phone,@nip)
end try
begin catch
    declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when adding company: \n'
+ERROR_MESSAGE();
    THROW 52000, @errormessage,1
end catch
```

9. Procedura add_customer

Dodaje klienta

```
create procedure add_customer
@CompanyID int,
@AttendeeID int
as
    begin try
        if ((@AttendeeID is null and @CompanyID is null) or (@AttendeeID is not
null and @CompanyID is not null))
        begin throw 52000,'Customer is private person or a company, those groups
are exclusive ',1
            end
        declare @customerID int
        select @customerID=customerid from Customer
        where @AttendeeID=AttendeeID and @CompanyID=CompanyID

        if @customerID is not null
        begin throw 52000,'Customer already exists',1
        end
insert into customer (CompanyID, AttendeeID)
values(@CompanyID,@AttendeeID)
    end try
    begin catch
        declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when adding customer: \n'
+ERROR_MESSAGE();
    THROW 52000, @errormessage,1
    end catch
```

10. Procedura add_attendee

Dodaje uczestnika

```
create procedure add_attendee
@Name nvarchar(40),
@Surname nvarchar(40),
@Email nvarchar(40),
@Phone nvarchar(20),
@City nvarchar(40),
@PostalCode nvarchar(40),
@Address nvarchar(40)
as
    begin try
        declare @attendeeID int
        select @attendeeID=attendeeID
        from Attendee
        where @name=name and @surname=surname and @phone=phone
        if @attendeeID is not null begin throw 52000,'Attendee already exists',1
        end
        insert into Attendee (Name, Surname, Email, Phone, City, PostalCode,
Address)
        values(@name,@surname,@email,@phone,@city,@postalcode,@address)
    end try
    begin catch
         declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when adding attendee: \n'
+ERROR_MESSAGE();
    THROW 52000, @errormessage,1
    end catch
```

11. Procedura add_conference_day_registration

Dodaje rejestracje na dzień conferencji

```
create procedure add_conference_day_registration
@conferenceDayReservationID int,
@AttendeeID int,
@studentPrice bit
as
    begin try
        if not exists(select * from ConferenceDayReservation  where
ConferenceDayReservationID=@conferenceDayReservationID)
        begin throw 52000,'No reservation for that time.',1
        end
        if not exists(select * from Attendee where AttendeeID=@AttendeeID)
        begin throw 52000,'Attendee does not exist',1
            end
        insert into
            RegistrationForConferenceDay(AttendeeID, ConferenceDayReservationID,
StudentPrice)
            values(@AttendeeID,@conferenceDayReservationID,@studentPrice)
    end try
    begin catch
          declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when doing registration for
conference day: \n' +ERROR_MESSAGE();
    THROW 52000, @errormessage,1
    end catch
```

12. Procedura add_workshop_registration

Dodaje rejestracje na warsztat

```
create procedure add_workshop_registration
@WorkshopReservationID int,
@registrationForConferenceDayID int
as
            begin try
                if not exists (select * from WorkshopReservation where
WorkshopReservationID=@WorkshopReservationID)
                begin throw 52000,'No reservation for that workshop',1
                end
                if not exists (select * from RegistrationForConferenceDay where
@registrationForConferenceDayID=RegistrationForConferenceDayID)
                begin throw 52000,'No day registration for that time',1
                end
                insert into RegistrationForWorkshop(workshopreservationid,
registrationforconferencedayid)
                values(@WorkshopReservationID,@registrationForConferenceDayID)
            end try
            begin catch
                 declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when doing registration for workshop:
\n' +ERROR_MESSAGE();
    THROW 52000, @errormessage,1
            end catch
```

13. Procedura remove_conference_day_reservation

```
create procedure remove_conference_day_reservation
@ConferenceDayReservationID int
```

```
as
    begin
        begin
            try
            if not exists (select * from ConferenceDayReservation as cdr where
cdr.ConferenceDayReservationID=@ConferenceDayReservationID)
            begin throw 52001,'Reservation for conference day  does not exist',1
                end
            delete from ConferenceDayReservation where
ConferenceDayReservationID=@ConferenceDayReservationID
        end try
        begin catch
            declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when doing  conference day
reservation removement : \n' +ERROR_MESSAGE();
    THROW 52000, @errormessage,1
        end catch
end
```

14. Procedura remove_workshop_reservation

```
create procedure remove_workshop_reservation
@WorkshopReservationID int
as
    begin
        begin
            try
            if not exists (select * from WorkshopReservation as wr where
wr.WorkshopReservationID=@WorkshopReservationID)
            begin throw 52001,'Reservation for workshop does not exist',1
        end
            delete from WorkshopReservation where
WorkshopReservationID=@WorkshopReservationID
            end try
        begin catch
            declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when doing workshop reservation
removement: \n' +ERROR_MESSAGE();
    THROW 52000, @errormessage,1
        end catch
    end
```

15. Procedura remove_reservation

Usuwa reserwacje

```
create procedure remove_reservation
@ReservationID int
as
    begin
        begin try
            if not exists (select * from Reservation r where
r.ReservationID=@ReservationID)
            begin throw 52001,'Reservation does not exist',1
            end
            delete from  Reservation where ReservationID=@ReservationID
            end try
          begin catch
            declare @errorMessage nvarchar(2048)
```

```
         set @errorMessage = 'Error occurred when doing reservation removement: \
n' +ERROR_MESSAGE();
    THROW 52000, @errormessage,1
        end catch

    end
```

16. Procedura remove_registration_for_workshop

Usuwa rejestracje dla warsztatu

```
create procedure remove_registration_for_workshop
@RegistrationForWorkshopID int
as
begin
    begin try
        if not exists (select * from RegistrationForWorkshop where
RegistrationForWorkshopID=@RegistrationForWorkshopID)
        begin throw 52001,'Registration for workshop does not exist',1
            end
        delete from RegistrationForWorkshop where
RegistrationForWorkshopID=@RegistrationForWorkshopID
    end try
    begin catch
        declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when doing registration for workshop
removement: \n' +ERROR_MESSAGE();
    THROW 52000, @errormessage,1
    end catch
end
```

17. Procedura payment_realisation

Dokonanie wplaty

```
create procedure payment_realisation
@PaymentAmount int,
@ReservationID int
as
    begin
        begin try
            if not exists (select * from Reservation where
ReservationID=@ReservationID)
            begin throw 52001,'Reservation does not exist',1
                end
            if @PaymentAmount < 0
            begin throw 52001,'Payment amount can not be below zero',1
            end
            update reservation set AmountPaid=AmountPaid+@PaymentAmount where
ReservationID=@ReservationID
            end try
        begin catch
            declare @errorMessage nvarchar(2048)
         set @errorMessage = 'Error occurred when doing payment realisation: \n'
+ERROR_MESSAGE();
    THROW 52000, @errormessage,1
        end catch
    end
```

18. Procedura unpaid_reservation_cancellation

Anuluje reserwacje co nie zostali oplacone w ciągu tygodnia

```
create procedure unpaid_reservation_cancellation
as
    update Reservation
    set reservation.isCancelled=1
    from Reservation join ReservationsWithPrices on
Reservation.ReservationID=ReservationsWithPrices.reservationid
    where not
        exists(
            select * from reservation r  where datediff(d,r.date,GETDATE()) <=7
            and ReservationsWithPrices.[Amount to Pay]=0
            )
```

4. Triggery
    1. Trigger reservation_cancelled

Anuluje wszystkie reserwacje dni konferecji reserwacja których została odwolana

```
create trigger reservation_cancelled
    on reservation
    after update
as begin
    update cdr
    set cdr.isCancelled=i.isCancelled
    from ConferenceDayReservation cdr
    inner join inserted i on i.ReservationID = cdr.ReservationID
    update wr
    set wr.isCancelled=i.isCancelled
    from WorkshopReservation wr
    inner join ConferenceDayReservation CDR2 on wr.ConferenceDayReservationID =
CDR2.ConferenceDayReservationID
    inner join inserted i on i.ReservationID = CDR2.ReservationID
    end
```

    2. Trigger too_few_places_conference_day

Sprawdza czy ilość zareserwowanych miejsc na konferencji nie jest większą od dostępnych

```
create trigger too_few_places_conference_day
  on ConferenceDay
   after update
as
    begin
if  exists(select * from inserted i where
dbo.get_conference_day_free_places_number(i.ConferenceDayID)<0)
        begin
            rollback transaction;
            throw 52001,'Too few places to reserve this many conference day
attendees.',1
    end
end
```

    3. Trigger too_few_places_workshop

Sprawdza czy ilość zareserwowanych miejsc na warsztacie nie jest większą od dostępnych

```
create trigger too_few_places_workshop
    on Workshop
   after update
as
    begin
```

```
if exists(select * from inserted i where dbo.get_workshop_free_places_number
(i.WorkshopID)<0)
begin
    rollback transaction;
    throw 52001,'Too few places to reserve this many workshop attendees',1
    end
end
```

4. Trigger conference_day_belongs_conference

Sprawdza czy data dodanego dnia konferencji należy do danej konferencji

```
create trigger conference_day_belongs_conference
    on ConferenceDay
    after insert
as
    begin
        if exists(select * from inserted i inner join conference c on
c.ConferenceID=i.ConferenceID
            where i.date not between c.startdate and c.enddate)

        begin
        rollback transaction;
        throw 52001,'Conference day must belong to conference',1
            end
    end
```

5. Trigger workshop_belongs_conference

Sprawdza czy data dodanego warsztaty należy do danego dnia konferencji

```
create trigger workshop_belongs_conference
    on Workshop
    after insert
as
    begin
        if exists(select * from inserted i inner join conferenceday cd on
cd.ConferenceDayID=i.ConferenceDayID
            and convert(date,i.starttime)!=cd.date)
        begin
            rollback transaction;
            throw 52002,'Workshop must belong to conference',1
            end
    end
```

6. Trigger start_of_one_threshold_is_end_plus_one_of_another

```
create trigger start_of_one_threshold_is_end_plus_one_of_another
    on PriceThreshold
    after insert
as
    begin
        if exists(select * from inserted i where i.startdate !=
            dateadd(day, +1 ,(select top 1 PriceThreshold.EndDate from
PriceThreshold where PriceThreshold.ConferenceDayID = i.ConferenceDayID
            order by EndDate desc)) or i.enddate>(select cd.date from
ConferenceDay cd where i.ConferenceDayID=cd.ConferenceDayID))
        begin
            rollback transaction;
            throw 52002,'Not valid date of threshold',1
```

```
        end
    end
```

Role
Aministrator dostęp do wszystkich procedur składowanych i widoków
Pracownik firmy
Klient
Uczestnik