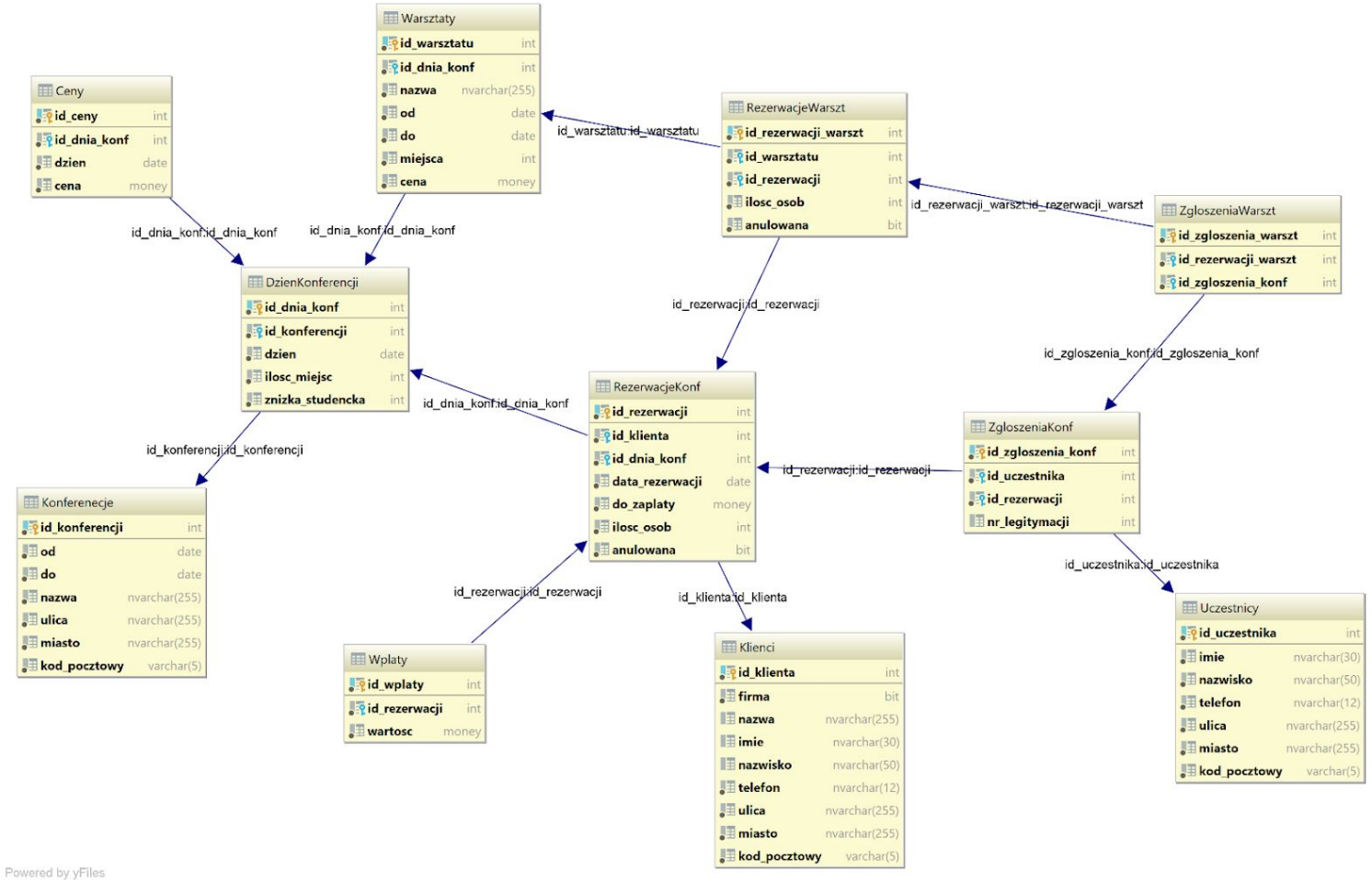
DOKUMENTACJA BAZY DANYCH KONFERENCJI

JAKUB RÓG

ALEKSANDER STAŃDO

Podstawy Baz Danych 2018/19

Schemat bazy danych



Użytkownicy oraz przypadki użycia

Klient prywatny/Firma

- Rejestracja w systemie
- Wybór terminu konferencji
- Edycja swoich danych
- Rezerwacja miejsca na wybrany termin konferencji
- Anulowanie swojej rezerwacji na konferencje
- Rejestracja na warsztaty odbywające się w tym samym terminie co wybrana konferencja
- Rezerwacja miejsca na wybrane warsztaty
- Wyświetlanie kwoty należności za udział w konferencji

Firma

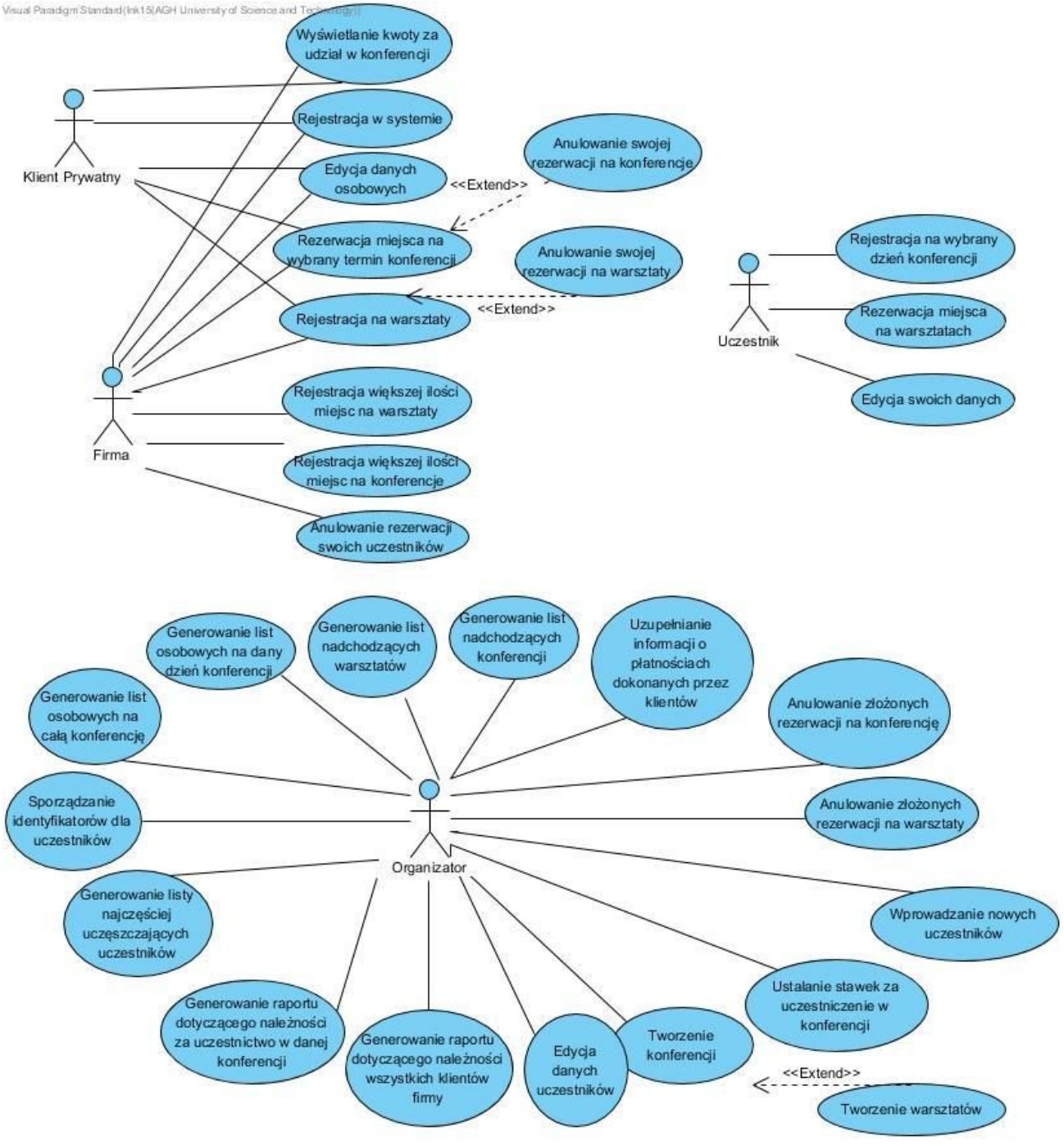
- Możliwość rezerwacji odpowiedniej ilości miejsc, bez konieczności podawania danych uczestników

Uczestnik

- Edycja swoich danych
- Rezerwacja miejsca na warsztatach
- Rejestracja na wybrany dzień konferencji

Organizator

- Tworzenie konferencji
- Tworzenie warsztatów
- Ustalanie stawek za uczestniczenie w konferencji
- Edycja danych uczestników
- Wprowadzanie nowych uczestników
- Anulowanie złożonych rezerwacji na konferencję
- Anulowanie złożonych rezerwacji na warsztaty
- Rezerwacja miejsc
- Uzupełnianie informacji o płatnościach dokonanych przez klientów
- Generowanie list nadchodzących konferencji
- Generowanie list nadchodzących warsztatów
- Generowanie list osobowych na dany dzień konferencji
- Sporządzanie identyfikatorów dla uczestników
- Generowanie listy najaktywniejszych klientów
- Generowanie raportu dotyczącego należności za uczestnictwo w danej konferencji
- Generowanie raportu dotyczącego należności wszystkich klientów firmy


Visual Paradigm Standard(Ink15(AGH University of Science and Technology))
Wyświetlanie kwoty za udział w konferencji
Rejestracja w systemie
Edycja danych osobowych
Rezerwacja miejsca na wybrany termin konferencji
Rejestracja na warsztaty
Rejestracja większej ilości miejsc na warsztaty
Rejestracja większej ilości miejsc na konferencje
Anulowanie rezerwacji swoich uczestników
Klient Prywatny
Firma
Anulowanie swojej rezerwacji na konferencje
Anulowanie swojej rezerwacji na warsztaty
<<Extend>>
<<Extend>>
Uczestnik
Rejestracja na wybrany dzień konferencji
Rezerwacja miejsca na warsztatach
Edycja swoich danych
Generowanie list osobowych na dany dzień konferencji
Generowanie list nadchodzących warsztatów
Generowanie list nadchodzących konferencji
Uzupełnianie informacji o płatnościach dokonanych przez klientów
Generowanie list osobowych na całą konferencję
Anulowanie złożonych rezerwacji na konferencję
Sporządzanie identyfikatorów dla uczestników
Anulowanie złożonych rezerwacji na warsztaty
Organizator
Generowanie listy najczęściej uczęszczających uczestników
Wprowadzanie nowych uczestników
Generowanie raportu dotyczącego należności za uczestnictwo w danej konferencji
Generowanie raportu dotyczącego należności wszystkich klientów firmy
Edycja danych uczestników
Tworzenie konferencji
Ustalanie stawek za uczestniczenie w konferencji
<<Extend>>
Tworzenie warsztatów

TABELE

Ceny

Tabela przechowuje informację o cenie konferencji, zawiera informację o identyfikatorze danej ceny (*id_ceny*), o identyfikatorze dnia konferencji którego dana cena dotyczy (*id_dnia_konferencji*). Ze względu na możliwość ustalania specjalnych promocji na poszczególne dni, np. na ostatnie miejsca dzień przed rozpoczęciem konferencji, tabela zawiera informację o cenie obowiązującej od wybranego dnia (*dzien*). Zawarta jest również wartość pieniężna ceny (*cena*).

```
CREATE TABLE [dbo].[Ceny]
(
  [id_ceny] [int] IDENTITY (1,1) NOT NULL,
  [id_dnia_konf] [int] NOT NULL,
  [dzien] [date] NOT NULL,
  [cena] [money] NOT NULL,
  CONSTRAINT [PK_ceny] PRIMARY KEY CLUSTERED
     (
      [id_ceny] ASC
     ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
       ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].Ceny WITH CHECK ADD CONSTRAINT [FK_ceny] FOREIGN KEY
(id_dnia_konf)
REFERENCES [dbo].[DzienKonferencji](id_dnia_konf)
alter table Ceny add check (cena >= 0)
alter table Ceny add default  getdate() for dzien
```

Warsztaty

Tabela przechowuje informacje o przeprowadzanych warsztatach w ramach dnia konferencji, zawiera identyfikator warsztatu (*id_warsztatu*), identyfikator dnia konferencji w ramach którego przeprowadzane są warsztaty (*id_dnia_konferencji*), nazwę warsztatu (*nazwa*), godzinę rozpoczęcia i zakończenia warsztatu (*od, do*), ilość wolnych miejsc (*miejsca*), oraz cenę (*cena*).

```
CREATE TABLE [dbo].[Warsztaty]
(
  [id_warsztatu] [int] IDENTITY (1,1) NOT NULL,
  [id_dnia_konf] [int] NOT NULL,
  [nazwa] [nvarchar] (255) NOT NULL,
  [od] [time] NOT NULL,
```

```
  [do] [time] NOT NULL,
  [miejsca] [int] NOT NULL,
  [cena] [money] NOT NULL,

  CONSTRAINT [PK_warsztatu] PRIMARY KEY CLUSTERED
     (
      [id_warsztatu] ASC
     ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
       ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].Warsztaty WITH CHECK ADD CONSTRAINT [FK_dzienKonf]
FOREIGN KEY (id_dnia_konf)
REFERENCES [dbo].DzienKonferencji(id_dnia_konf)
alter table Warsztaty add check (cena >= 0)
alter table Warsztaty add check (miejsca > 0)
alter table Warsztaty add check (do >= od )
```

DzienKonferencji

Tabela przechowuje informację o danym dniu konferencji, zawarte są w niej informacje o identyfikatorze dnia konferencji (id_dnia_konf), o identyfikatorze, konferencji w której dany dzień występuje (*id_konferencji*). Zawarte są również podstawowe informacje o dacie danego dnia (*dzien)*, ilości dostępnych miejsc (*ilosc_miejsc)*, oraz obowiązującej zniżce studenckiej przechowywanej w całkowitoliczbowej wartości procentowej zniżki.

```
CREATE TABLE [dbo].[DzienKonferencji]
(
  [id_dnia_konf]   [int] IDENTITY (1,1) NOT NULL,
  [id_konferencji] [int]              NOT NULL,
  [dzien]          [date]            NOT NULL,
  [ilosc_miejsc]   [int]              NOT NULL,
  [znizka_studencka] [int]       NOT NULL,
  CONSTRAINT [PK_dnia_konf] PRIMARY KEY CLUSTERED
     (
      [id_dnia_konf] ASC
       ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
       ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].DzienKonferencji WITH CHECK ADD CONSTRAINT [FK_konferencje]
FOREIGN KEY (id_konferencji)
REFERENCES [dbo].[Konferencje](id_konferencji)
alter table DzienKonferencji add default 0 for znizka_studencka

alter table DzienKonferencji add check (znizka_studencka between 0 and 100)

alter table DzienKonferencji add check (ilosc_miejsc > 0)
```

Konferencje

Tabela przechowuje dane o konferencjach, zawarty jest w niej identyfikator konferencji (*id_konferencji*), datę rozpoczęcia i zakończenia konferencji (*od, do*), jej nazwę (*nazwa*) oraz dokładny adres (*ulica, miasto, kod_pocztowy*).

```
CREATE TABLE [dbo].[Konferencje]
(
 [id_konferencji] [int] IDENTITY (1,1) NOT NULL,
 [od]          [date]            NOT NULL,
 [do]          [date]            NOT NULL,
 [nazwa]       [nvarchar](255)    NOT NULL,
 [ulica]       [nvarchar](255)    NOT NULL,
 [miasto]      [nvarchar](50)     NOT NULL,
 [kod_pocztowy]  [varchar](5)      NOT NULL,
 CONSTRAINT [PK_konferencji] PRIMARY KEY CLUSTERED
  (
   id_konferencji ASC
    ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
    ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

alter table Konferencje add check (od <= do)
alter table Konferencje add check (kod_pocztowy LIKE '[0-9][0-9][0-9][0-9][0-9]')
```

RezerwacjeKonf

Tabela przechowuje informacje o rezerwacjach na poszczególne dni konferencji, zawiera informacje o identyfikatorze rezerwacji (*id_rezerwacji*), identyfikator klienta składającego rezerwację (*id_klienta*), identyfikator dnia konferencji, którego dana rezerwacja dotyczy (*id_dnia_konf*), datę złożenia rezerwacji (*data_rezerwacji*), wartość pieniężną należną za udział w konferencji (*do_zaplaty*), zadeklarowaną ilość uczestników na dany dzień konferencji (*ilosc_osob*), oraz informację czy dana rezerwacja nie została anulowana (*anulowana*).

```
CREATE TABLE [dbo].[RezerwacjeKonf]
(
 [id_rezerwacji]    [int] IDENTITY (1,1),
 [id_klienta]       [int]  NOT NULL,
 [id_dnia_konf]     [int]  NOT NULL,
 [data_rezerwacji] [date] NOT NULL,
 [do_zaplaty]       [money]  NULL,
 [ilosc_osob]       [int] NOT NULL,
 [anulowana]        [bit] NOT NULL,
  CONSTRAINT [PK_rezerwacji] PRIMARY KEY CLUSTERED
    (
     [id_rezerwacji] ASC
      ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
```

```
      ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].RezerwacjeKonf WITH CHECK ADD CONSTRAINT
[FK_dzienKonferencji] FOREIGN KEY (id_dnia_konf)
REFERENCES [dbo].DzienKonferencji(id_dnia_konf)

ALTER TABLE [dbo].RezerwacjeKonf WITH CHECK ADD CONSTRAINT [FK_klient]
FOREIGN KEY (id_klienta)
REFERENCES [dbo].Klienci(id_klienta)

alter table RezerwacjeKonf add check (do_zaplaty >= 0)

alter table RezerwacjeKonf add default  0 for anulowana

alter table RezerwacjeKonf add check (ilosc_osob > 0)

alter table RezerwacjeKonf add check (data_rezerwacji <= getdate())

alter table RezerwacjeKonf add default getdate() for data_rezerwacji
```

Wplaty

Tabela przechowuje informacje o wpłatach dokonanych na rzecz danej rezerwacji, zawiera identyfikator wpłaty (*id_wplaty*), identyfikator rezerwacji do której odnosi się dana wpłata (*id_rezerwacji*) oraz wartość otrzymanej wpłaty (*wartosc*).

```
CREATE TABLE [dbo].[Wplaty]
(
  [id_wplaty] [int] IDENTITY (1,1) NOT NULL,
  [id_rezerwacji] [int] NOT NULL,
  [wartosc] [money] NOT NULL,
  CONSTRAINT [PK_wplaty] PRIMARY KEY CLUSTERED
     (
      [id_wplaty] ASC
     ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
       ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].Wplaty WITH CHECK ADD CONSTRAINT [FK rezerw] FOREIGN KEY
(id_rezerwacji)
REFERENCES [dbo].RezerwacjeKonf(id_rezerwacji)
```

RezerwacjeWarszt

Tabela przechowuje informacje o rezerwacjach na poszczególne warsztaty, zawiera informację o identyfikatorze rezerwacji danego warsztatu (*id_rezerwacji_warszt*), identyfikator warsztatu, którego dana rezerwacja dotyczy (*id_warsztatu*). Identyfikator rezerwacji dnia konferencji, w ramach którego przeprowadzane są warsztaty (*id_rezerwacji*), ilość osób zgłoszonej do udziału w danych warsztatach, oraz informację czy dana rezerwacja nie została anulowana (*anulowana*).

```
CREATE TABLE [dbo].[RezerwacjeWarszt]
(
  [id_rezerwacji_warszt]     [int] IDENTITY (1,1) NOT NULL,
  [id_warsztatu]        [int]  NOT NULL,
  [id_rezerwacji]     [int]  NOT NULL,
  [ilosc_osob]      [int] NOT NULL,
  [anulowana]       [bit] NOT NULL,
  CONSTRAINT [PK_rezerwacji_warszt] PRIMARY KEY CLUSTERED
     (
      [id_rezerwacji_warszt] ASC
       ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
       ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].RezerwacjeWarszt WITH CHECK ADD CONSTRAINT [FK_warsztaty]
FOREIGN KEY (id_warsztatu)
REFERENCES [dbo].Warsztaty(id_warsztatu)

ALTER TABLE [dbo].RezerwacjeWarszt WITH CHECK ADD CONSTRAINT
[FK_rezerwacjeK] FOREIGN KEY (id_rezerwacji)
REFERENCES [dbo].RezerwacjeKonf(id_rezerwacji)

alter table RezerwacjeWarszt add check (ilosc_osob > 0)

alter table RezerwacjeWarszt add default 0 for anulowana
```

ZgloszeniaWarszt

Tabela przechowuje informacje o zgłoszeniach na poszczególne warsztaty, zawarte są w niej identyfikator zgłoszenia na dany dzień warsztatów (id_zgloszenia_warszt)**,** identyfikator rezerwacji danego warsztatu (*id_rezerwacji_warszt*) oraz identyfikator zgłoszenia na konferencję powiązaną z danym warsztatem (*id_zgloszenia_konf*).

```
CREATE TABLE [dbo].[ZgloszeniaWarszt]
(
  [id_zgloszenia_warszt] [int] IDENTITY (1,1) NOT NULL,
  [id_rezerwacji_warszt]  [int]              NOT NULL,
  [id_zgloszenia_konf]   [int]             NOT NULL,
  CONSTRAINT [PK_zglosznia_warszt] PRIMARY KEY CLUSTERED
```

```
    (
     [ID_zgloszenia_warszt] ASC
      ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
      ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].ZgloszeniaWarszt WITH CHECK ADD CONSTRAINT
[FK_zgloszenieKonf] FOREIGN KEY (id_zgloszenia_konf)
REFERENCES [dbo].ZgloszeniaKonf(id_zgloszenia_konf)

ALTER TABLE [dbo].ZgloszeniaWarszt WITH CHECK ADD CONSTRAINT
[FK_rezerwWarsz] FOREIGN KEY (id_rezerwacji_warszt)
REFERENCES [dbo].RezerwacjeWarszt(id_rezerwacji_warszt)
```

ZgloszeniaKonf

Tabela przechowuje informację o zgłoszonych na daną konferencję uczestnikach, zawiera identyfikator zgłoszenia uczestnika na konferencję (*id_zgloszenia_konf*), identyfikator uczestnika (*id_uczestnika*), identyfikator rezerwacji na dzień konferencji, na który zgłaszany jest uczestnik (*id_rezerwacji*), oraz opcjonalny numer legitymacji studenckiej w ramach, którego uczestnik może otrzymać zniżkę (*nr_legitymacji*).

```
CREATE TABLE [dbo].[ZgloszeniaKonf]
(
  [id_zgloszenia_konf] [int] IDENTITY (1,1) NOT NULL,
  [id_uczestnika]        [int]              NOT NULL,
  [id_rezerwacji]      [int]              NOT NULL,
  [nr_legitymacji]     [int]              NULL,
  CONSTRAINT [PK_zglosznia] PRIMARY KEY CLUSTERED
    (
     [id_zgloszenia_konf] ASC
      ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
      ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

ALTER TABLE [dbo].ZgloszeniaKonf WITH CHECK ADD CONSTRAINT [FK_uczestnik]
FOREIGN KEY (id_uczestnika)
REFERENCES [dbo].[Uczestnicy](id_uczestnika)

ALTER TABLE [dbo].ZgloszeniaKonf WITH CHECK ADD CONSTRAINT [FK_rezerw]
FOREIGN KEY (id_rezerwacji)
REFERENCES [dbo].[RezerwacjeKonf](id_rezerwacji)

ALTER TABLE [dbo].ZgloszeniaKonf WITH CHECK ADD CONSTRAINT [FK_rezerwacje]
FOREIGN KEY (id_rezerwacji)
REFERENCES [dbo].[RezerwacjeKonf](id_rezerwacji)

ALTER TABLE [dbo].ZgloszeniaKonf ADD CHECK (nr_legitymacji > 0)
```

Uczestnicy

Tabela przechowuje dane o uczestnikach wszelkich konferencji, zawiera identyfikator uczestnika (*id_uczestnika*), imię oraz nazwisko (*imię, nazwisko*), telefon kontaktowy (*telefon*) oraz adres korespondencyjny (*ulica, miasto, kod_pocztowy*).

```
CREATE TABLE [dbo].[Uczestnicy]
(
  [id_uczestnika] [int] IDENTITY (1,1) NOT NULL,
  [imie]          [nvarchar](30)       NOT NULL,
  [nazwisko]      [nvarchar](50)       NOT NULL,
  [telefon]       [nvarchar](12)       NOT NULL UNIQUE,
  [ulica]         [nvarchar](255)      NOT NULL,
  [miasto]        [nvarchar](50)       NOT NULL,
  [kod_pocztowy]  [varchar](5)         NOT NULL,
  CONSTRAINT [PK_uczestnika] PRIMARY KEY CLUSTERED
    (
     [ID_uczestnika] ASC
     ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
      ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
alter table Uczestnicy add check (kod_pocztowy LIKE '[0-9][0-9][0-9][0-9][0-9]')
```

Klienci

Tabela przechowuje informacje o osobach składających rezerwację na konferencje, zawiera identyfikator klienta (*id_klienta*), informację czy dany klient jest firmą czy nie (*firma*), imię i nazwisko klienta lub nazwę firmy (*firma*), telefon kontaktowy (*telefon*) oraz adres (*ulica, miasto, kod_pocztowy*).

```
CREATE TABLE [dbo].[Klienci]
(
  [id_klienta]   [int] IDENTITY (1,1) NOT NULL,
  [firma]        [bit]                NOT NULL,
  [nazwa]        [nvarchar](255)      NULL,
  [imie]         [nvarchar](30)       NULL,
  [nazwisko]     [nvarchar](50)       NULL,
  [telefon]      [nvarchar](12)       NOT NULL,
  [ulica]        [nvarchar](255)      NOT NULL,
  [miasto]       [nvarchar](50)       NOT NULL,
  [kod_pocztowy] [varchar](5)         NOT NULL,
  CONSTRAINT [PK_klienta] PRIMARY KEY CLUSTERED
    (
     [id_klienta] ASC
     ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
      ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

alter table Klienci add check (kod_pocztowy LIKE '[0-9][0-9][0-9][0-9][0-9]')
```

WIDOKI

Nadchodzące i trwające konferencje

Widok wyświetla informację o nadchodzących i trwających konferencjach

```
CREATE VIEW NadchodzaceKonferencje AS

select nazwa as 'Nazwa',od, do, ulica + ', ' + kod_pocztowy + ' ' + miasto as Lokalizacja
from Konferencje
where do >= GETDATE()
```

Nadchodzące i trwające warsztaty

Widok wyświetla informację o nadchodzących i trwających warsztatach

```
CREATE VIEW NadchodzaceWarsztaty AS

select A.nazwa as 'Nazwa', ulica + ', ' + kod_pocztowy + ' ' + miasto as Lokalizacja,
Konferencje.nazwa as 'W ramach konferencji', Konferencje.od, Konferencje.do
from Warsztaty as A
left outer join DzienKonferencji
on A.id_dnia_konf = DzienKonferencji.id_dnia_konf
left outer join Konferencje
on DzienKonferencji.id_konferencji = Konferencje.id_konferencji
where DzienKonferencji.dzien >= getdate()
```

Ilość wolnych miejsc na nadchodzące dni konferencji

Widok wyświetla informację o nazwie konferencji, dacie dnia konferencji, ilości miejsc i ilości wolnych miejsc dla nadchodzących konferencji.

```
CREATE VIEW WolneMiejscaNaKonferencje AS

select Nazwa, dzien as Data, ilosc_miejsc as 'Ilosc miejsc', ilosc_miejsc - (select
ISNULL(sum(ilosc_osob),0) from RezerwacjeKonf where anulowana = 0 and id_dnia_konf =
Dzien.id_dnia_konf) as 'Ilosc wolnych miejsc'
from  Konferencje as Konf
inner join DzienKonferencji as Dzien
on Dzien.id_konferencji = Konf.id_konferencji
where dzien > GETDATE()
```

Ilość wolnych miejsc na nadchodzące dni warsztatów

Widok przedstawia n=informację o nazwie warsztatu, dacie, lokalizacji, ilości miejsc i ilości wolnych miejsc dla nadchodzących warsztatów

```
CREATE VIEW IloscMiejscNaWarsztaty AS

select A.nazwa,  dzien, A.od, A.do,ulica + ', ' + kod_pocztowy + ' ' + kod_pocztowy as
'Lokalizacja', ilosc_miejsc as 'Miejsca', ilosc_miejsc - isnull((select SUM(ilosc_osob) from
RezerwacjeWarszt where id_warsztatu = A.id_warsztatu),  0) as 'Ilosc wolnych miejsc'
from Warsztaty as A
inner join DzienKonferencji
on DzienKonferencji.id_dnia_konf = A.id_dnia_konf
inner join Konferencje
on Konferencje.id_konferencji = DzienKonferencji.id_konferencji
where DzienKonferencji.dzien >= GETDATE()
```

Uczestnicy nadchodzących i trwających warsztatów

Widok przedstawia dane osobowe uczestników trwających i nadchodzących warsztatów.

```
CREATE VIEW UczestnicyWarsztatów AS

select nazwa,Dzien.dzien as ‘Dzien’, od, do, imie + ' ' + nazwisko as 'imię i nazwisko', telefon
from Warsztaty inner join RezerwacjeWarszt
on Warsztaty.id_warsztatu = RezerwacjeWarszt.id_warsztatu
inner join ZgloszeniaWarszt
on ZgloszeniaWarszt.id_rezerwacji_warszt = RezerwacjeWarszt.id_rezerwacji_warszt
inner join ZgloszeniaKonf
on ZgloszeniaKonf.id_zgloszenia_konf = ZgloszeniaWarszt.id_zgloszenia_konf
inner join Uczestnicy
on Uczestnicy.id_uczestnika = ZgloszeniaKonf.id_uczestnika
inner join DzienKonferencji as Dzien
on Warsztaty.id_dnia_konf = Dzien.id_dnia_konf
where anulowana = 0 and Dzien.dzien >= getdate()
```

Uczestnicy nadchodzących i trwających konferencji

Widok przedstawia dane osobowe uczestników trwających i nadchodzących konferencji.

```
CREATE VIEW UczestnicyKonferencji as
select DzienKonferencji.dzien, nazwa, imie, nazwisko, nr_legitymacji
from Uczestnicy
inner join ZgloszeniaKonf
on Uczestnicy.id_uczestnika = ZgloszeniaKonf.id_uczestnika
inner join RezerwacjeKonf
```

```
on RezerwacjeKonf.id_rezerwacji = ZgloszeniaKonf.id_rezerwacji
inner join DzienKonferencji
on DzienKonferencji.id_dnia_konf = RezerwacjeKonf.id_dnia_konf
inner join Konferencje
on Konferencje.id_konferencji = DzienKonferencji.id_konferencji
where anulowana = 0 and DzienKonferencji.dzien >= getdate()
```

Kto ile ma do zapłaty i ile zapłacił

Widok przedstawia informacje o tym ile dany klient musi łącznie zapłacić za rezerwację dnia konferencji (wliczając w to zarezerwowane warsztaty w danym dniu) oraz ile już zapłacił. Nazwą w widoku jest Nazwa firmy dla firm lub imię i nazwisko dla klientów prywatnych.

```
CREATE VIEW Zaplaty AS

SELECT dbo.RezerwacjeKonf.id_rezerwacji,dbo.Klienci.Nazwa, dbo.Klienci.telefon,
dbo.RezerwacjeKonf.do_zaplaty AS'Lacznie do zaplaty',
(SELECT sum(ISNULL(W.Wartosc, 0))
        FROM RezerwacjeKonf as R LEFT JOIN
Wplaty AS W ON R.id_rezerwacji = W.id_rezerwacji
 WHERE R.id_rezerwacji = dbo.RezerwacjeKonf.id_rezerwacji AND
        dbo.RezerwacjeKonf.anulowana = 0) AS 'Zaplacone',
dbo.Konferencje.nazwa AS 'Nazwa koncefernji',
dbo.DzienKonferencji.dzien as 'Dzien konferencji'
FROM dbo.RezerwacjeKonf LEFT JOIN
 dbo.Klienci ON dbo.RezerwacjeKonf.id_klienta = dbo.Klienci.id_klienta LEFT JOIN
 dbo.DzienKonferencji ON dbo.RezerwacjeKonf.id_dnia_konf =
dbo.DzienKonferencji.id_dnia_konf
LEFT JOIN
 dbo.Konferencje ON dbo.Konferencje.id_konferencji = dbo.DzienKonferencji.id_konferencji
WHERE (dbo.Klienci.firma=1) AND (dbo.RezerwacjeKonf.anulowana = 0)
AND datediff(d,dbo.DzienKonferencji.dzien,getdate()) <= 14

UNION

SELECT dbo.RezerwacjeKonf.id_rezerwacji,dbo.Klienci.imie + ' ' + dbo.Klienci.nazwisko as
'Nazwa', dbo.Klienci.telefon,
dbo.RezerwacjeKonf.do_zaplaty AS'Lacznie do zaplaty',
(SELECT sum(ISNULL(W.Wartosc, 0))
        FROM RezerwacjeKonf as R LEFT JOIN
        Wplaty AS W ON R.id_rezerwacji = W.id_rezerwacji
         WHERE R.id_rezerwacji = dbo.RezerwacjeKonf.id_rezerwacji AND
         dbo.RezerwacjeKonf.anulowana = 0) AS 'Zaplacone',
dbo.Konferencje.nazwa AS 'Nazwa koncefernji', dbo.DzienKonferencji.dzien as 'Dzien
konferencji'
FROM dbo.RezerwacjeKonf LEFT JOIN
 dbo.Klienci ON dbo.RezerwacjeKonf.id_klienta = dbo.Klienci.id_klienta LEFT JOIN
 dbo.DzienKonferencji ON dbo.RezerwacjeKonf.id_dnia_konf =
dbo.DzienKonferencji.id_dnia_konf
```

```
LEFT JOIN
 dbo.Konferencje ON dbo.Konferencje.id_konferencji = dbo.DzienKonferencji.id_konferencji
WHERE (dbo.Klienci.firma=0) AND (dbo.RezerwacjeKonf.anulowana = 0)
AND datediff(d,dbo.DzienKonferencji.dzien,getdate()) <= 14
```

Rezerwacje z brakującymi uczestnikami

Widok przedstawia informacje o rezerwacjach na konferencje, które nie mają zgłoszonych wszystkich uczestników, wraz z informacjami ile uczestników zadeklarowali i ile osób brakuje.

```
CREATE VIEW brakujace_zgloszenia AS

SELECT RezerwacjeKonf.id_rezerwacji, Klienci.nazwa as 'Nazwa', Klienci.telefon,
 Konferencje.nazwa as 'Nazwa Konferencji', RezerwacjeKonf.ilosc_osob as 'Ile zgloszono',
 RezerwacjeKonf.ilosc_osob - (SELECT count(*)
                                                FROM ZgloszeniaKonf AS Z
                                                 WHERE Z.id_rezerwacji =
RezerwacjeKonf.id_rezerwacji) AS 'Ilu brakuje'
FROM RezerwacjeKonf LEFT JOIN
          Klienci ON RezerwacjeKonf.id_klienta = Klienci.id_klienta LEFT JOIN
    DzienKonferencji on RezerwacjeKonf.id_dnia_konf = DzienKonferencji.id_dnia_konf LEFT
JOIN
    Konferencje ON Konferencje.id_konferencji = DzienKonferencji.id_konferencji
WHERE (Klienci.firma=1) AND (RezerwacjeKonf.anulowana = 0) AND
  RezerwacjeKonf.ilosc_osob - (SELECT count(*)
  FROM ZgloszeniaKonf AS Z
  WHERE Z.id_rezerwacji = RezerwacjeKonf.id_rezerwacji) > 0 AND
  datediff(d,DzienKonferencji.dzien,getdate()) <= 14

UNION

SELECT RezerwacjeKonf.id_rezerwacji, Klienci.imie + ' ' + Klienci.nazwisko as 'Nazwa',
Klienci.telefon,
 Konferencje.nazwa as 'Nazwa Konferencji', RezerwacjeKonf.ilosc_osob as 'Ile zgloszono',
 RezerwacjeKonf.ilosc_osob - (SELECT count(*)
                                                FROM ZgloszeniaKonf AS Z
                                                WHERE Z.id_rezerwacji =
RezerwacjeKonf.id_rezerwacji) AS 'Ilu brakuje'
FROM RezerwacjeKonf LEFT JOIN
          Klienci ON RezerwacjeKonf.id_klienta = Klienci.id_klienta LEFT JOIN
    DzienKonferencji on RezerwacjeKonf.id_dnia_konf = DzienKonferencji.id_dnia_konf LEFT
JOIN
    Konferencje ON Konferencje.id_konferencji = DzienKonferencji.id_konferencji
WHERE (Klienci.firma = 0) AND (RezerwacjeKonf.anulowana = 0) AND
  RezerwacjeKonf.ilosc_osob - (SELECT count(*)
  FROM ZgloszeniaKonf AS Z
  WHERE Z.id_rezerwacji = RezerwacjeKonf.id_rezerwacji) > 0 AND
```

```
  datediff(d,DzienKonferencji.dzien,getdate()) <= 14
```

Odbyte konferencje

Widok przedstawia dany o konferencjach, które odbyły się w przeszłości

```
CREATE VIEW Odbyte_konferencje AS
SELECT nazwa as 'Nazwa', od, do, ulica + ', ' + kod_pocztowy + ' ' + miasto as Lokalizacja
from Konferencje
where do < GETDATE()
```

Odbyte warsztaty

Widok przedstawia dany o warsztatach, które odbyły się w przeszłości

```
CREATE VIEW Odbyte_Warsztaty AS

SELECT Warsztaty.nazwa AS 'Nazwa warsztatu', DzienKonferencji.dzien AS 'Dzien',
Warsztaty.od, Warsztaty.do, Konferencje.nazwa AS 'W ramach konferencji',
(SELECT isnull(sum(ilosc_osob),0)
 FROM RezerwacjeWarszt
 WHERE RezerwacjeWarszt.id_warsztatu = Warsztaty.id_warsztatu
 AND RezerwacjeWarszt.anulowana=0) AS 'Ilosc uczestnikow'
FROM Warsztaty LEFT JOIN DzienKonferencji ON
 Warsztaty.id_dnia_konf = DzienKonferencji.id_dnia_konf LEFT JOIN
 Konferencje ON Konferencje.id_konferencji = DzienKonferencji.id_konferencji
 WHERE DzienKonferencji.dzien < GETDATE()
```

Firmy najczęściej dokonujący rezerwacji na dni konferencji i warsztaty

Widok przedstawia 10 firm, które dokonały najwięcej nieanulowanych rezerwacji na dni konferencji i warsztaty wraz z liczbą dokonanych przez nie nieanulowanych rezerwacji.

```
CREATE VIEW Najaktywniejsze_firmy AS

SELECT TOP 10 Klienci.nazwa AS 'Nazwa firmy',
 (SELECT count(*) FROM RezerwacjeKonf
  WHERE RezerwacjeKonf.id_klienta = Klienci.id_klienta AND
RezerwacjeKonf.anulowana=0) +
 (SELECT count(*) FROM RezerwacjeWarszt INNER JOIN
  RezerwacjeKonf ON RezerwacjeWarszt.id_rezerwacji = RezerwacjeKonf.id_rezerwacji
  WHERE Klienci.id_klienta = RezerwacjeKonf.id_klienta
  AND RezerwacjeWarszt.anulowana = 0) AS 'Ilosc nieanulowanych rezerwacji'
FROM Klienci
WHERE Klienci.firma = 1
Order BY 2 DESC
```

PROCEDURY

Generowanie identyfikatorów dla podanego dnia konferencji

Widok z procedurą, która dla podanego dnia konferencji poda nazwę konferencji, datę oraz imię, nazwisko i telefon wszystkich uczestników danego dnia konferencji tak, aby można było na tej podstawie stworzyć dla nich identyfikatory.

```
CREATE PROCEDURE OsobyNadanymDniuKonferencji
   @idDniaKonferencji int
AS
BEGIN
   SET NOCOUNT ON;
   SELECT Konferencje.nazwa, DzienKonferencji.dzien, Uczestnicy.imie,
Uczestnicy.nazwisko, Uczestnicy.telefon
   FROM DzienKonferencji INNER JOIN Konferencje ON
   DzienKonferencji.id_konferencji = Konferencje.id_konferencji
   INNER JOIN RezerwacjeKonf ON RezerwacjeKonf.id_dnia_konf = @idDniaKonferencji
   INNER JOIN ZgloszeniaKonf ON ZgloszeniaKonf.id_rezerwacji =
RezerwacjeKonf.id_rezerwacji
   INNER JOIN Uczestnicy ON Uczestnicy.id_uczestnika = ZgloszeniaKonf.id_uczestnika
   WHERE RezerwacjeKonf.anulowana = 0

END
```

Dodawanie nowej konferencji

Procedura tworzenia nowej konferencji

```
Create procedure DodajKonferencje
@Od date,
@Do date,
@Nazwa nvarchar(255),
@Ulica nvarchar(255),
@Miasto nvarchar(50),
@KodPocztowy varchar(5)

AS
BEGIN
        SET NOCOUNT ON;
        IF(@Od <= @Do)
                BEGIN
                INSERT INTO Konferencje(od, do, nazwa, ulica, miasto, kod_pocztowy)
                VALUES (@Od, @Do, @Nazwa, @Ulica, @Miasto, @KodPocztowy)
                END
        ELSE
                BEGIN
```

```
		RAISERROR('Zła data',-1,-1)
		END
END
```

Dodawanie dni w ramach dodanej konferencji

Procedura tworzenia dnia konferencji w ramach istniejącej konferencji

```
CREATE PROCEDURE DodajDzienKonferencji
@IDKonferencji int,
@Dzien date,
@IloscMiejsc int,
@ZnizkaStudencka int

AS
BEGIN
	SET NOCOUNT ON;
	IF((Select COUNT(*) from Konferencje where Konferencje.id_konferencji =
@IDKonferencji group by id_konferencji) <> 0)
		BEGIN
		DECLARE @Od AS date
		DECLARE @Do AS date
		SET @Od = (select od from Konferencje where Konferencje.id_konferencji =
@IDKonferencji)
		SET @DO = (select do from Konferencje where Konferencje.id_konferencji =
@IDKonferencji)
		IF(@Dzien >= @Od and @Dzien <= @Do)
			BEGIN
			IF(@ZnizkaStudencka >=0 and @ZnizkaStudencka <= 100 and
@IloscMiejsc >=0 )
				BEGIN
				INSERT INTO DzienKonferencji(id_konferencji, dzien,
ilosc_miejsc, znizka_studencka)
				VALUES (@IDKonferencji, @Dzien, @IloscMiejsc,
@ZnizkaStudencka)
				END
			ELSE
				BEGIN
				RAISERROR('Błędna wartość zniżki studenckiej [0;100] lub
błędna wartość ilości wolnych miejsc (>=0).', -1,-1)
				END
			END
		ELSE
			BEGIN
			RAISERROR('Data nie mieści się w przedziale dat konferencji.', -1, -1)
			END
```

```
		END
	ELSE
		BEGIN
		RAISERROR('Nie znaleziono konferencji o danym IDKonferencji', -1, -1)
		END
END
```

Dodawanie nowego warsztatu

Procedura dodawania nowego warsztatu.

```
CREATE PROCEDURE DodajWarsztat
@IDDniaKonferencji int,
@Nazwa nvarchar(255),
@Od time,
@Do time,
@Miejsca int,
@Cena money

AS
BEGIN
SET NOCOUNT ON;

	IF((Select COUNT(*) from DzienKonferencji where id_dnia_konf =
@IDDniaKonferencji group by id_dnia_konf) <> 0)
		BEGIN
		DECLARE @DataDniaKonf as date
		SET @DataDniaKonf = (select dzien from DzienKonferencji where
id_dnia_konf = @IDDniaKonferencji)
		IF(@Od < @Do)
			BEGIN
			IF(@Miejsca>=0 and @Cena >= 0)
				BEGIN
				INSERT INTO Warsztaty(id_dnia_konf, nazwa, od, do, miejsca,
cena)
				VALUES (@IDDniaKonferencji, @Nazwa, @Od, @Do,
@Miejsca, @Cena)
				END
			ELSE
				BEGIN
				RAISERROR('Wprowadzono nie dozwoloną cenę lub liczbę
miejsc.', -1, -1)
				END
			END
		ELSE
			BEGIN
			RAISERROR('Wprowadzono błędne godziny, zakończenie konferencji
musi się odbyć po jej rozpoczęciu.',-1,-1)
			END
		END
```

```
        ELSE
                BEGIN
                RAISERROR('Brak dnia konferencji o podanym id',-1,-1)
                END
END
```

Dodawanie nowego klienta

Procedura tworzenia nowego klienta, Firma = 0 jeśli jest to klient prywatny, 1 jeśli jest to firma.

```
CREATE PROCEDURE DodajKlienta
@Firma bit,
@Nazwa nvarchar(255) = null,
@Imie nvarchar(30) = null,
@Nazwisko nvarchar(30) = null,
@Telefon nvarchar(12),
@Ulica nvarchar(255),
@Miasto nvarchar(50),
@kod_pocztowy nvarchar(5)

AS
BEGIN
        if( (@Firma = 1 and @Nazwa is not null) or
                (@Firma = 0 and @Imie is not null and @Nazwisko is not null))
                BEGIN
                INSERT INTO Klienci(firma, nazwa, imie, nazwisko, telefon, ulica, miasto,
kod_pocztowy)
                VALUES (@Firma, @Nazwa, @Imie, @Nazwisko, @Telefon, @Ulica,
@Miasto, @kod_pocztowy)
                END
        ELSE
                BEGIN
                RAISERROR('Firma musi posiadać nazwę a klient indywidualny imię i
nazwisko.', -1, -1)
                END
END
```

Dodawanie nowego uczestnika

Procedura dodawania nowego uczestnika.

```
CREATE PROCEDURE DodajUczestnika
@Imie nvarchar(30),
@Nazwisko nvarchar(50),
@Telefon nvarchar(12),
@Ulica nvarchar(255),
@Miasto nvarchar(50),
```

```
@KodPocztowy varchar(5)
AS
BEGIN

        INSERT INTO Uczestnicy(imie, nazwisko, telefon, ulica, miasto, kod_pocztowy)
        VALUES (@Imie, @Nazwisko, @Telefon, @Ulica, @Miasto, @KodPocztowy)

END
```

Ustalanie ceny

Procedura dodania ceny do danego dnia konferencji.

```
CREATE PROCEDURE DodajCene
@IDDniaKonferencji int,
@Dzien date,
@Cena money

AS
BEGIN

        IF( exists (select * from DzienKonferencji where id_dnia_konf = @IDDniaKonferencji))
                BEGIN
                DECLARE @DataDniaKonferencji as Date
                SET @DataDniaKonferencji = (Select dzien from DzienKonferencji where
id_dnia_konf = @IDDniaKonferencji)
                IF(@Dzien <= @DataDniaKonferencji and @Cena >= 0 )
                        BEGIN
                        INSERT INTO Ceny(id_dnia_konf, dzien, cena)
                        VALUES (@IDDniaKonferencji, @Dzien, @Cena)
                        END
                ELSE
                        BEGIN
                        RAISERROR('Błędna data lub cena', -1, -1)
                        END
                END
        ELSE
                BEGIN
                RAISERROR('Nie znaleziono dnia konferencji o takim numerze.', -1, -1)
                END
END
```

Dodawanie nowej wpłaty

Procedura dodania wpłaty do danej rezerwacji

```
CREATE PROCEDURE DodajNowaPlatnosc
@IDRezerwacji as int,
```

```
@Wartosc as money

AS
BEGIN

	SET NOCOUNT ON;
	IF(SELECT anulowana FROM RezerwacjeKonf WHERE id_rezerwacji =
@IDRezerwacji)=0
		BEGIN
		INSERT INTO Wplaty(id_rezerwacji, wartosc)
		VALUES (@IDRezerwacji, @Wartosc)
		END
	ELSE
		BEGIN
		RAISERROR('Dla anulowanej rejestracji nie można dokonać wpłaty',-1,-1)
		END

END
```

Anulowanie rezerwacji

Procedura anulowania rezerwacji (dokonuje się poprzez zmianę flagi “anulowana” na 1)

```
CREATE PROCEDURE AnulujRezerwacjeKonf
@IDRezerwacji int
AS
BEGIN
	UPDATE RezerwacjeKonf SET anulowana = 1 where @IDRezerwacji = id_rezerwacji
END
```

Zmiana danych klienta

Procedura zmiany danych klienta, zmienia tylko te dane, które nie są nullem.

```
CREATE PROCEDURE ZmienDaneKlienta
@IDKlienta int,
@Firma bit = null,
@Nazwa nvarchar(255) = null,
@Imie nvarchar(30) = null,
@Nazwisko nvarchar(30) = null,
@Telefon nvarchar(12) = null,
@Ulica nvarchar(255) = null,
@Miasto nvarchar(50) = null,
@kod_pocztowy nvarchar(5) = null

AS
BEGIN
	IF @Firma is not null
		BEGIN
```

```
		UPDATE Klienci SET firma = @Firma where id_klienta = @IDKlienta
		END
	IF @Nazwa is not null
		BEGIN
		UPDATE Klienci SET nazwa = @Nazwa where id_klienta = @IDKlienta
		END
	IF @Imie is not null
		BEGIN
		UPDATE Klienci SET imie = @Imie where id_klienta = @IDKlienta
		END
	IF @Nazwisko is not null
		BEGIN
		UPDATE Klienci SET nazwisko = @Nazwisko where id_klienta = @IDKlienta
		END
	IF @Telefon is not null
		BEGIN
		UPDATE Klienci SET telefon = @Telefon where id_klienta = @IDKlienta
		END
	IF @Ulica is not null
		BEGIN
		UPDATE Klienci SET ulica = @Ulica where id_klienta = @IDKlienta
		END
	IF @Miasto is not null
		BEGIN
		UPDATE Klienci SET miasto = @Miasto where id_klienta = @IDKlienta
		END
	IF @kod_pocztowy is not null
		BEGIN
		UPDATE Klienci SET kod_pocztowy = @kod_pocztowy where id_klienta =
@IDKlienta
		END
END
```

Dodanie rezerwacji na dzień konferencji

Procedura umożliwia dodanie rezerwacji na dany dzień konferencji. Wartość w polu “do_zaplaty” będzie wyliczana w triggerze.

```
Create procedure DodajRezerwacjeKonferencji
@idDniaKonf int,
@idKlienta int,
@iloscosob int

AS
BEGIN
 SET NOCOUNT ON;
 INSERT INTO  RezerwacjeKonf(id_klienta,id_dnia_konf,data_rezerwacji,ilosc_osob)
 VALUES (@idKlienta, @idDniaKonf, getdate(), @iloscosob)
END
```

Dodanie rezerwacji na warsztat

Procedura umożliwia dodanie rezerwacji na dany warsztat. Po dodaniu warsztatu trigger będzie aktualizować wartość “do_zapłaty” w rekordzie rezerwacji dnia konferencji powiązanym z tworzoną rezerwacją warsztatu.

```
CREATE PROCEDURE DodajRezerwacjeWarsztatu
@idWarsztatu int,
@idRezerwacji int,
@iloscOsob int

AS
BEGIN
SET NOCOUNT ON;
INSERT INTO RezerwacjeWarszt(id_warsztatu,id_rezerwacji,ilosc_osob)
VALUES (@idWarsztatu, @idRezerwacji, @iloscOsob)
END
```

Zgłoś uczestnika na konferencje

Procedura tworzy tworzy nowe zgłoszenie i dodaje na nie uczestnika. Jeśli Uczestnik już istnieje w bazie (co stwierdzamy przez imię, nazwisko i telefon) to łączymy zgłoszenie z istniejącym uczestnikiem. W przeciwnym wypadku tworzymy nowego uczestnika. Jeśli uczestnik jest studentem, to należy podać numer legitymacji w celu uwzględnienia zniżki studenckiej.

```
CREATE PROCEDURE ZglosuczestnikaNaKonferencje
@imie NVARCHAR(30),
@nazwisko NVARCHAR(50),
@telefon NVARCHAR(12),
@ulica NVARCHAR(255),
@miasto NVARCHAR(50),
@kod_pocztowy VARCHAR(5),
@idRezerwacjiKonf int,
@nrLegitymacji int=NULL
AS
BEGIN
   SET NOCOUNT ON;
   DECLARE @idUczestnika int

   SET @idUczestnika = (SELECT Uczestnicy.id_uczestnika FROM Uczestnicy
                                        WHERE  Uczestnicy.imie = @imie AND
                                        Uczestnicy.nazwisko = @nazwisko AND
                                        Uczestnicy.telefon = @telefon)
   IF @idUczestnika IS NULL
         BEGIN
               EXEC dbo.DodajUczestnika
                     @imie,
                     @nazwisko,
                     @telefon,
```

```
                    @ulica,
                    @miasto,
                    @kod_pocztowy

                    SET @idUczestnika = (SELECT Uczestnicy.id_uczestnika FROM
Uczestnicy
                                        WHERE  Uczestnicy.imie = @imie AND
                                        Uczestnicy.nazwisko = @nazwisko AND
                                        Uczestnicy.telefon = @telefon)
        END
   INSERT INTO ZgloszeniaKonf(id_uczestnika,id_rezerwacji, nr_legitymacji)
   VALUES (@idUczestnika, @idRezerwacjiKonf, @nrLegitymacji)
END
```

Zgłoś uczestnika na warsztat

Procedura umożliwia zgłoszenie uczestnika na warsztat (uczestnik musi być wcześniej zgłoszony na dany dzień konferencji)

```
CREATE PROCEDURE ZglosUczestnikaNaWarsztat
@idRezerwacjiWarsztatu int,
@idZgloszenia int
AS
BEGIN
   SET NOCOUNT ON;

   INSERT INTO ZgloszeniaWarszt(id_rezerwacji_warszt, id_zgloszenia_konf)
   VALUES(@idRezerwacjiWarsztatu, @idZgloszenia);
END
```

Zmiana maksymalnej ilości osób na dzień konferencji

Procedura zmienia maksymalną ilość osób na wybrany dzień konferencji (możliwość takiej zmiany jest sprawdzana w triggerze)

```
CREATE PROCEDURE ZmianaIlosciOsobNaDzienKonferencji
@idDniaKonferencji int,
@nowaIloscosob int

AS
BEGIN
        SET NOCOUNT ON;

        UPDATE DzienKonferencji
        SET ilosc_miejsc = @nowaIloscosob
        WHERE id_dnia_konf = @idDniaKonferencji

END
```

Zmiana maksymalnej ilości osób na warsztacie

Procedura zmienia maksymalną ilość osób na wybrany warsztat (możliwość takiej zmiany jest sprawdzana w triggerze)

```
CREATE PROCEDURE ZmianaIlosciOsobNaWarsztat
@idWarsztatu int,
@nowaIloscosob int

AS
BEGIN
   SET NOCOUNT ON;

   UPDATE Warsztaty
   SET miejsca = @nowaIloscosob
   WHERE id_warsztatu = @idWarsztatu

END
```

Zmiana danych osobowych uczestnika

Procedura umożliwia zmianę danych osobowych użytkownika. Zmienione zostaną tylko te dane, które są podane (nie są nullami).

```
CREATE PROCEDURE ZmianaDanychUczestnika
@idUczestnika int,
@imie VARCHAR(30)=NULL,
@nazwisko VARCHAR(50)=NULL,
@ulica VARCHAR(255)=NULL,
@miasto VARCHAR(50)=NULL,
@kod VARCHAR(5)=NULL,
@tel VARCHAR(12)=NULL
AS
BEGIN

   SET NOCOUNT ON;

   IF @imie IS NOT NULL
         BEGIN
         UPDATE Uczestnicy
         SET imie = @imie WHERE  id_uczestnika = @idUczestnika
         END
   IF @nazwisko IS NOT NULL
         BEGIN
         UPDATE Uczestnicy
         SET nazwisko = @nazwisko WHERE  id_uczestnika = @idUczestnika
         END
   IF @ulica IS NOT NULL
         BEGIN
         UPDATE Uczestnicy
```

```
        SET ulica = @ulica WHERE  id_uczestnika = @idUczestnika
        END
    IF @miasto IS NOT NULL
        BEGIN
        UPDATE Uczestnicy
        SET miasto = @miasto WHERE  id_uczestnika = @idUczestnika
        END
    IF @kod IS NOT NULL
        BEGIN
        UPDATE Uczestnicy
        SET kod_pocztowy = @kod WHERE  id_uczestnika = @idUczestnika
        END
    IF @tel IS NOT NULL
        BEGIN
        UPDATE Uczestnicy
        SET telefon = @tel WHERE  id_uczestnika = @idUczestnika
        END
END
```

Anulowanie rezerwacji warsztatu

Procedura umożliwia anulowanie rezerwacji warsztatu. Cena do_zapłaty w rezerwacji dnia konferencji powiązanym z tą rezerwacją warsztatu jest zmieniana w triggerze.

```
CREATE PROCEDURE AnulujRezerwacjeWarsztatu
@IDRezerwacji int
AS
BEGIN
        IF (select anulowana from RezerwacjeWarszt where id_rezerwacji_warszt =
@IDRezerwacji) = 0
        BEGIN
        UPDATE RezerwacjeWarszt SET anulowana = 1 where @IDRezerwacji =
id_warsztatu
        END
END
```

TRIGGERY

Anulowanie warsztatu przy anulowaniu rezerwacji dnia konferencji

Jeśli anulujemy rezerwację dnia konferencji, to anulujemy też rezerwację na warsztaty w danym dniu konferencji

```
CREATE TRIGGER AnulowanieWarsztatow ON RezerwacjeKonf FOR UPDATE AS
BEGIN
	IF UPDATE(anulowana)
		BEGIN
		IF ((select anulowana from inserted) = 1 )
			BEGIN
			UPDATE RezerwacjeWarszt SET anulowana = 1 WHERE
id_rezerwacji = (select id_rezerwacji from inserted)
			END
		END
END
```

Sprawdzanie czy ilość uczestników konferencji nie przekracza ich maksymalnej ilości

Trigger sprawdza, czy można dodać nową rezerwację dnia konferencji. Jeśli po rejestracji ilość osób zarejestrowanych przekroczy ilość maksymalnych miejsc to transakcja ta jest odrzucana.

```
CREATE TRIGGER PrzekroczenieLiczbyUczestnikow ON RezerwacjeKonf FOR INSERT AS
BEGIN
	DECLARE @LiczbaObecnychUczestnikow as int
	DECLARE @MaksymalnaIloscUczestnikow as int
	DECLARE @IDDniaKonf as int

	SET @IDDniaKonf = (select id_dnia_konf from inserted)

	SET @LiczbaObecnychUczestnikow = (select isnull(SUM(ilosc_osob),0) from
RezerwacjeKonf where id_dnia_konf = @IDDniaKonf AND anulowana = 0)

	SET @MaksymalnaIloscUczestnikow = (select ilosc_miejsc from inserted INNER
JOIN DzienKonferencji ON
                                                    inserted.id_dnia_konf =
DzienKonferencji.id_dnia_konf)


	IF ( @LiczbaObecnychUczestnikow > @MaksymalnaIloscUczestnikow )
	BEGIN
	RAISERROR('Przekroczono liczbe wolnych miejsc',-1,-1)
```

```
        ROLLBACK TRANSACTION
        END

END
```

Sprawdzanie, czy ilość uczestników konferencji nie przekracza ich maksymalnej ilości (przy zmianie ilości miejsc konferencji)

Trigger sprawdza, czy można zmnienić ilość miejsc na dniu konferencji (w szczególności zmniejszyć). Jeśli zbyt dużo osób zarejestrowało się na dany dzień konferencji, to transakcja zmiany ilości miejsc jest odrzucana.

```
CREATE TRIGGER [PrzekraczenieLiczbyUczestnikowDwa ON DzienKonferencji
 AFTER UPDATE AS
 BEGIN
        DECLARE @LiczbaObecnychUczestnikow as int
        DECLARE @MaksymalnaIloscUczestnikow as int
        DECLARE @IDDniaKonf as int

        SET @IDDniaKonf = (select id_dnia_konf from inserted)

        SET @MaksymalnaIloscUczestnikow = (select ilosc_miejsc FROM inserted)

        SET @LiczbaObecnychUczestnikow = (select isnull(sum(ilosc_osob),0)
       from RezerwacjeKonf
       where id_dnia_konf = @IDDniaKonf AND anulowana = 0)
        iF (@LiczbaObecnychUczestnikow > @MaksymalnaIloscUczestnikow)
                BEGIN
                        RAISERROR('Nie mozna zmniejszyc ilosci miejsc, poniewaz zapisalo
sie zbyt duzo osob',-1,-1)
                        ROLLBACK TRANSACTION
                END
 END
```

Sprawdzanie czy ilość uczestników warsztatu nie przekracza ich maksymalnej ilości

Trigger sprawdza, czy można dodać nową rezerwację warsztatu. Jeśli po rejestracji ilość osób zarejestrowanych przekroczy ilość maksymalnych miejsc to transakcja ta jest odrzucana.

```
CREATE TRIGGER PrzekroczenieLiczbyUczestnikowWarszt ON RezerwacjeWarszt FOR
INSERT AS
BEGIN
        DECLARE @LiczbaObecnychUczestnikow as int
        DECLARE @MaksymalnaIloscUczestnikow as int
        DECLARE @IDWarsztatu as int
```

```
	SET @IDWarsztatu = (select id_warsztatu from inserted)

	SET @LiczbaObecnychUczestnikow = (select isnull(SUM(ilosc_osob),0) from
RezerwacjeWarszt where id_warsztatu = @IDWarsztatu AND anulowana = 0)

	SET @MaksymalnaIloscUczestnikow = (select miejsca from inserted INNER JOIN
Warsztaty ON Warsztaty.id_warsztatu = @IdWarsztatu)


	IF ( @LiczbaObecnychUczestnikow > @MaksymalnaIloscUczestnikow )
	BEGIN
	RAISERROR('Przekroczono liczbe wolnych miejsc', -1, -1)
	ROLLBACK TRANSACTION
	END

END
```

Sprawdzanie czy ilość uczestników warsztatu nie przekracza ich maksymalnej ilości (przy zmianie ilości miejsc warsztatu)

Trigger sprawdza, czy można zmnienić ilość miejsc na warsztacie (w szczególności zmniejszyć). Jeśli zbyt dużo osób zarejestrowało się na dany warsztat, to transakcja zmiany ilości miejsc jest odrzucana.

```
CREATE TRIGGER  PrzekroczenieLiczbyUczestnikowDwa ON Warsztaty
AFTER UPDATE AS
BEGIN
	DECLARE @LiczbaObecnychUczestnikow as int
	DECLARE @MaksymalnaLiczbaUczystnikow as int
	DECLARE @IDWarsztatu as int

	SET @IDWarsztatu = (select id_warsztatu FROM inserted)
	SET @LiczbaObecnychUczestnikow = (select isnull(sum(ilosc_osob),0) FROM
		RezerwacjeWarszt WHERE id_warsztatu = @IDWarsztatu AND anulowana
= 0)
	SET @MaksymalnaLiczbaUczystnikow = (select miejsca FROM inserted)

	IF (@LiczbaObecnychUczestnikow > @MaksymalnaLiczbaUczystnikow)
		BEGIN
			RAISERROR('Nie mozna zmiejszyc ilosci miejsc, poniewaz zapisalo
sie zbyt duzo osob', -1, -1)
			ROLLBACK TRANSACTION
		END
END
```

Wyliczanie ceny przy dodaniu rezerwacji

Trigger wylicza kwotę “do_zaplaty” przy dodaniu rezerwacji, poprzez znalezienie odpowiedniej dla danej rezerwacji ceny, a następnie pomnożenie jej przez ilosc osob, dla których dokonuje się rezerwacji.

```
CREATE TRIGGER Policz_cene ON RezerwacjeKonf
AFTER insert AS
BEGIN
   DECLARE @SumaDoZaplaty as money
   DECLARE @IloscOsob as int
   DECLARE @Cena as money
   DECLARE @IdRezerwacji as int

   SET @IdRezerwacji = ( SELECT id_rezerwacji FROM inserted)

   SET @cena = dbo.znajdz_cene( @IdRezerwacji )

   SET @IloscOsob = (select ilosc_osob FROM inserted)

   SET @SumaDoZaplaty = @cena * @IloscOsob

   UPDATE RezerwacjeKonf
   SET RezerwacjeKonf.do_zaplaty = @sumaDoZaplaty
   FROM RezerwacjeKonf INNER JOIN inserted ON RezerwacjeKonf.id_rezerwacji =
inserted.id_rezerwacji

END
```

Aktualizacja ceny przy zgłoszeniu uczestnika z legitymacją

Ponieważ obliczenie kwoty “do_zaplaty” w rezerwacji dnia konferencji zakłada, że żadna z osób, dla których złożono rezerwację nie jest studentem, to gdy okaże się, że zgłoszona osoba jest studentem należy pomniejszyć cenę w “do_zapłaty” tak, aby odpowiadała kwocie ze studentem.

```
CREATE TRIGGER Zaktualizuj_cene_legitymacja ON ZgloszeniaKonf
AFTER insert AS
BEGIN
   IF((select nr_legitymacji FROM inserted ) IS NOT NULL)
         BEGIN
         DECLARE @IdRezerwacji int
         DECLARE @cena money
         DECLARE @znizka_studencka int

         SET @IdRezerwacji = (SELECT @IdRezerwacji FROM inserted)
         SET @cena = dbo.znajdz_cene(@IdRezerwacji)
         SET @znizka_studencka = (SELECT DzienKonferencji.znizka_studencka FROM
                  DzienKonferencji INNER JOIN RezerwacjeKonf
```

```
                    ON DzienKonferencji.id_dnia_konf = RezerwacjeKonf.id_dnia_konf
                     WHERE RezerwacjeKonf.id_rezerwacji = @IdRezerwacji)

        UPDATE RezerwacjeKonf
        SET do_zaplaty -= @znizka_studencka * @cena / 100 -- od kwoty do zapłaty
odejmujemy
        --cenę i dodajemy cenę ze zniżką (cena * (100-znizka)/100), czyli realnie
odejmujemy
 -- podaną wyżej kwotę
        WHERE RezerwacjeKonf.id_rezerwacji = @IdRezerwacji
        END
END
```

Aktualizacja do zapłaty przy dodaniu warsztatu

W przypadku dodaniu warsztatu cena “do_zapłaty” w rezerwacji dnia konferencji zostanie powiększona o cenę rezerwacji warsztatu.

```
CREATE TRIGGER Zaktualizuj_cene_dodanie_warsztatu ON RezerwacjeWarszt
AFTER insert AS
BEGIN
   DECLARE @ilosc_osob int
   DECLARE @cena money
   DECLARE @id_rezerwacji int

   SET @ilosc_osob = (SELECT ilosc_osob FROM inserted)
   SET @cena = (SELECT cena FROM
                        Warsztaty INNER JOIN inserted ON
                        inserted.id_warsztatu = Warsztaty.id_warsztatu)
   SET @id_rezerwacji = (SELECT id_rezerwacji FROm inserted)

   UPDATE RezerwacjeKonf
   SET RezerwacjeKonf.do_Zaplaty += @ilosc_Osob * @cena
   WHERE RezerwacjeKonf.id_rezerwacji = @id_rezerwacji
END
```

Aktualizacja ceny po anulowanie warsztatu

W przypadku anulowaniu warsztatu cena “do_zapłaty” w rezerwacji dnia konferencji zostanie pomniejszona o cenę rezerwacji warsztatu.

```
CREATE TRIGGER Zaktualizuj_cene_anulowanie_warsztatu ON RezerwacjeWarszt
AFTER UPDATE AS
BEGIN
   DECLARE @ilosc_osob int
   DECLARE @cena money
   DECLARE @id_rezerwacji int
```

```
   SET @ilosc_osob = (SELECT ilosc_osob FROM inserted)
   SET @cena = (SELECT cena FROM
                        Warsztaty INNER JOIN inserted ON
                        inserted.id_warsztatu = Warsztaty.id_warsztatu)
   SET @id_rezerwacji = (SELECT id_rezerwacji FROm inserted)
   IF (SELECT anulowana FROM inserted) = 1
         BEGIN
         UPDATE RezerwacjeKonf
         SET RezerwacjeKonf.do_Zaplaty -= @ilosc_Osob * @cena
         WHERE RezerwacjeKonf.id_rezerwacji = @id_rezerwacji
   END
END
```

Sprawdzenie, czy zgłoszenie warsztatu dla danej osoby nie koliduje z innymi zgłoszonymi warsztatami

Trigger sprawdza, czy dodane zgłoszenie warsztatu dla danej osoby koliduje z innymi zgłoszeniami warsztatów tej osoby w danym dniu i jeśli tak, to wycofuje transakcję. Jednocześnie trigger uniemożliwia dwukrotny zapis na ten sam warsztat.

```
CREATE TRIGGER Sprawdz_kolizje_warsztatow ON ZgloszeniaWarszt
AFTER insert AS
BEGIN
   DECLARE @od_zgloszenie time
   DECLARE @do_zgloszenie time
   DECLARE @id_zgloszenia_konf int

   SET @id_zgloszenia_konf = (SELECT id_zgloszenia_konf FROM inserted)

   SET @od_zgloszenie = (SELECT W.od FROM RezerwacjeWarszt AS R
                                          INNER JOIN inserted AS Z
                                          ON Z.id_rezerwacji_warszt = R.id_rezerwacji_warszt
                                          INNER JOIN Warsztaty as W
                                          ON W.id_warsztatu = R.id_warsztatu)

   sET @do_zgloszenie = (SELECT W.do FROM RezerwacjeWarszt AS R
                                          INNER JOIN inserted AS Z
                                          ON Z.id_rezerwacji_warszt = R.id_rezerwacji_warszt
                                          INNER JOIN Warsztaty as W
                                          ON W.id_warsztatu = R.id_warsztatu)
   IF ((SELECT count(*) FROM ZgloszeniaWarszt as ZW
         INNER JOIN RezerwacjeWarszt as RW ON
  RW.id_rezerwacji_warszt = ZW.id_rezerwacji_warszt
         INNER JOIN Warsztaty as W ON W.id_warsztatu = RW.id_warsztatu
         WHERE ZW.id_zgloszenia_konf = @id_zgloszenia_konf AND
         ((W.od <= @od_zgloszenie AND W.do >= @od_zgloszenie) OR
          (W.od <= @do_zgloszenie AND W.do >= @do_zgloszenie))
         ) > 1)
         BEGIN
```

```
        RAISERROR('Nie można dodać warsztatu, koliduje on z innym warsztatem ', -1, -1)
        ROLLBACK
        END
END
```

Sprawdzenie, czy ilość osób, dla których zarezerwowany jest warsztat nie jest większa niż ilość osób, dla których zarezerwowany jest dzień konferencji

Trigger sprawdza, czy przy próbie zarezerwowania warsztatu nie próbujemy zarezerwować więcej miejsc niż wynika to z rezerwacji dnia konferencji powiązanej z wprowadzoną rezerwacją warsztatu.

```
CREATE TRIGGER LiczbaRzerwowanychMiejsc ON RezerwacjeWarszt
FOR INSERT
AS
BEGIN
   DECLARE @LiczbaOsobDzienKonferencji as int
   DECLARE @LiczbaOsobWarsztat as int

   SET @LiczbaOsobDzienKonferencji = (SELECT  RezerwacjeKonf.ilosc_osob
                        FROM RezerwacjeWarszt INNER JOIN RezerwacjeKonf
                ON RezerwacjeWarszt.id_rezerwacji = RezerwacjeKonf.id_rezerwacji
                WHERE RezerwacjeWarszt.id_rezerwacji_warszt =
                (SELECT id_rezerwacji_warszt FROM inserted)
                )
   SET @LiczbaOsobWarsztat = (SELECT ilosc_osob FROM inserted)

   IF (@LiczbaOsobWarsztat > @LiczbaOsobDzienKonferencji)
         BEGIN
                RAISERROR('Liczba osób zgłoszonych na warsztat nie może być większa
niż liczba osób zgłoszona na konferencje',-1,-1)
                ROLLBACK TRANSACTION
         END
END
```

FUNKCJE

Znajdź cenę

Funkcja dla podanego id Rezerwacji odnajdzie odpowiednią cenę na ten dzień rezerwacji (tj. cenę o “największej” dacie, która ma datę przed datą złożenia rejestracji)

```
CREATE FUNCTION znajdz_cene
(
   @idRezerwacji int
)
RETURNS money
AS
BEGIN
    DECLARE @cena money

    SET @cena = ISNULL((SELECT top 1 Ceny.Cena FROM RezerwacjeKonf
                                INNER JOIN DzienKonferencji ON
                                DzienKonferencji.id_dnia_konf = RezerwacjeKonf.id_dnia_konf
                                INNER JOIN Ceny ON
                                DzienKonferencji.id_dnia_konf = Ceny.id_dnia_konf
                                WHERE RezerwacjeKonf.id_rezerwacji = @idRezerwacji AND
                                DzienKonferencji.dzien > Ceny.dzien
                                ORDER BY Ceny.Dzien DESC),0)
        RETURN @cena
END
```

INDEKSY

Indeksy nieklastrowe stworzono dla wszystkich kluczy obcych w tabeli. Ponieważ nasza baza danych nie przewiduje częstego usuwania wpisów (zamiast tego stosujemy flagę “anulowana”) to takie rozwiązanie jest optymalne.

```
CREATE NONCLUSTERED INDEX [id_konferencji_index_DzienKonferencji]
ON DzienKonferencji(id_konferencji)

CREATE NONCLUSTERED INDEX [id_dnia_konf_index_Ceny]
ON Ceny(id_dnia_konf)

CREATE NONCLUSTERED INDEX [id_dnia_konf_index_Warsztaty]
ON Warsztaty(id_dnia_konf)

CREATE NONCLUSTERED INDEX [id_klienta_index_RezerwacjeKonf]
ON RezerwacjeKonf(id_klienta)
```

```
CREATE NONCLUSTERED INDEX [id_dnia_konf_index_RezerwacjeKonf]
ON RezerwacjeKonf(id_dnia_konf)

CREATE NONCLUSTERED INDEX [id_rezerwacji_index_Wplaty]
ON Wplaty(id_rezerwacji)

CREATE NONCLUSTERED INDEX [id_warsztatu_index_RezerwacjeWarszt]
ON RezerwacjeWarszt(id_warsztatu)

CREATE NONCLUSTERED INDEX [id_rezerwacji_index_RezerwacjeWarszt]
ON RezerwacjeWarszt(id_rezerwacji)

CREATE NONCLUSTERED INDEX [id_uczestnika_index_ZgloszeniaKonf]
ON ZgloszeniaKonf(id_uczestnika)

CREATE NONCLUSTERED INDEX [id_rezerwacji_index_ZgloszeniaKonf]
ON ZgloszeniaKonf(id_rezerwacji)

CREATE NONCLUSTERED INDEX [id_rezerwacji_warszt_index_ZgloszeniaWarszt]
ON ZgloszeniaWarszt(id_rezerwacji_warszt)

CREATE NONCLUSTERED INDEX [id_zgloszenia_konf_index_ZgloszeniaWarszt]
ON ZgloszeniaWarszt(id_zgloszenia_konf)
```

GENERATOR DANYCH

Dane zostały wygenerowane za pomocą programu SQL Data Generator 4, podczas zapełniania bazy danych wyłączone zostały triggery, ponieważ nie udało się ustawić programu tak aby suma zarezerwowanych miejsc na daną konferencję lub warsztat była mniejsza od sumy miejsc dozwolonych. Stąd też w widokach pojawiają się ujemne wartości przy ilości wolnych miejsc na konferencje i na warsztaty. Problemem w generowanych danych okazały się również zapłaty, które zostały wygenerowane zdecydowanie większe niż kwota do zapłaty.