

Podstawy Baz Danych

Projekt: System zarządzania konferencjami

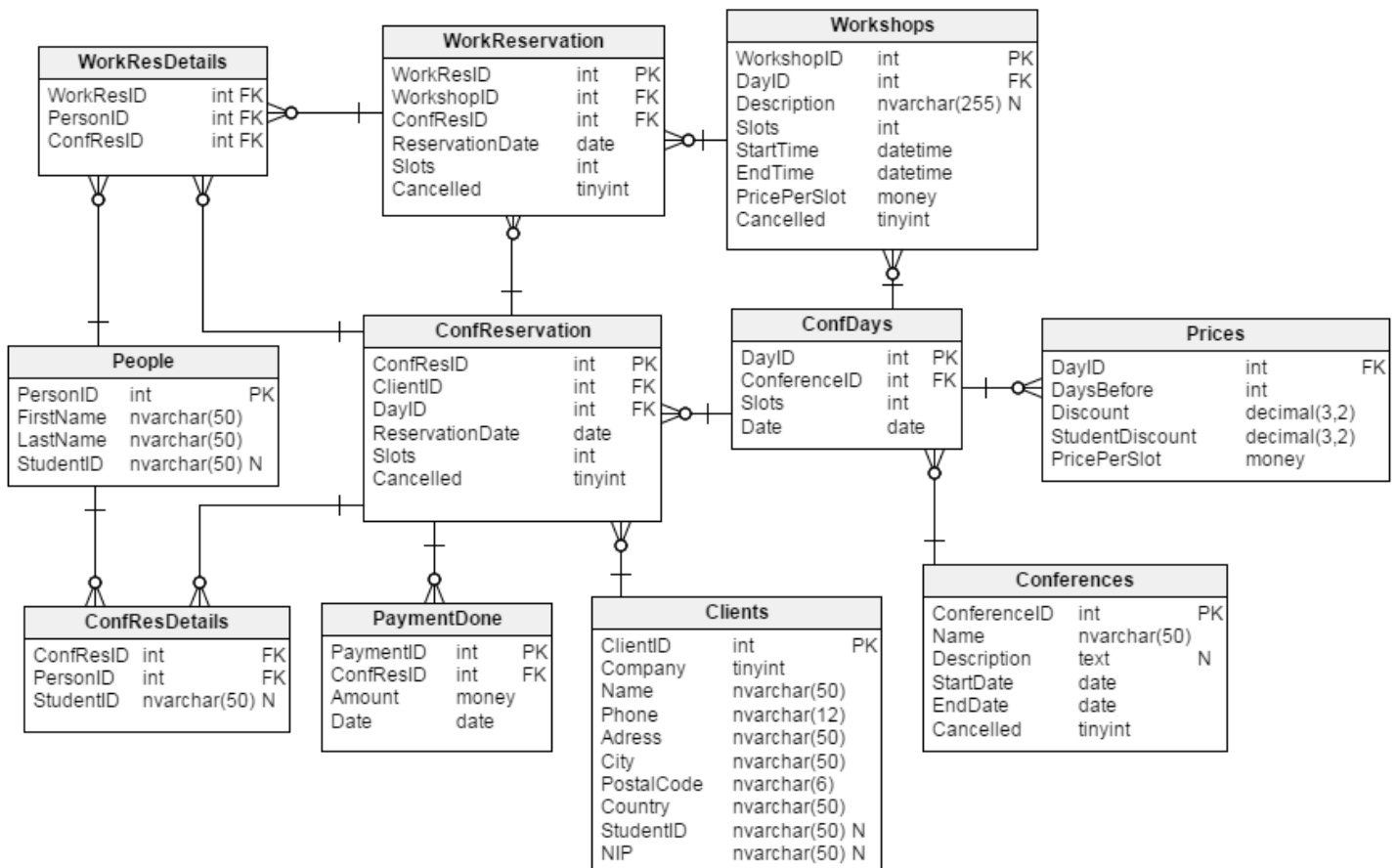
Artur Czopek

Mateusz Kasznia

1. Opis:

Firma organizuje konferencje, które mogą być jedno lub kilkudniowe. Klientami mogą być zarówno indywidualne osoby jak i firmy, natomiast uczestnikami konferencji są osoby (firma nie musi podawać od razu przy rejestracji listy uczestników może zarezerwować odpowiednią ilość miejsc na określone dni oraz na warsztaty, natomiast na 2 tygodnie przed rozpoczęciem musi te dane uzupełnić). Dla konferencji kilkudniowych, uczestnicy mogą rejestrować na dowolne z tych dni, dowolną liczbę osób. Klient może zmienić liczbę osób na rezerwacji, lub całkiem ją anulować (do 2 tygodni przed konferencją).

2. Schemat:



3. Opis tabel:

- **Clients**

Tabela reprezentująca klientów w bazie danych. Każdy klient posiada dane: (Company, Name, Phone, Address, City, PostalCode, Country, StudentID, NIP). Jeśli klient jest firmą to pole (Company) ustawione jest na 1, a w polu (Name) znajduje się nazwa firmy, dla klienta będącego osobą prywatną (Firma) ustawiona jest na 0, telefon może składać się tylko ze znaków numerycznych (cyfry i znaki typu +/), kod musi być typu *-*** (gdzie * to cyfra).

```
CREATE TABLE Clients (  
    ClientID int NOT NULL IDENTITY(1,1),  
    Company tinyint NOT NULL CHECK(Clients.Company LIKE '[0-1]'),  
    Name nvarchar(50) NOT NULL,  
    Phone nvarchar(50) NOT NULL CHECK(Clients.Phone LIKE  
    '+[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),  
    Address nvarchar(50) NOT NULL,  
    City nvarchar(50) NOT NULL,  
    PostalCode nvarchar(50) NOT NULL CHECK(Clients.PostalCode LIKE  
    '[0-9][0-9]-[0-9][0-9][0-9]'),  
    Country nvarchar(50) NOT NULL,  
    StudentID nvarchar(50) NULL,  
    NIP nvarchar(50) NULL,  
    CONSTRAINT Clients_pk PRIMARY KEY (ClientID)  
);
```

- **Conferences**

Tabela do której wpisywane są konferencje. Konferencja posiada dane: (Name, Description, StartDate, EndDate, Cancelled). Konferencja może zostać anulowana (pole 'Cancelled' ustawiane wtedy na 1).

```
CREATE TABLE Conferences (  
    ConferenceID int NOT NULL IDENTITY(1,1),  
    Name nvarchar(50) NOT NULL,  
    Description varchar(255) NULL,  
    StartDate date NOT NULL CHECK (StartDate > GETDATE()),  
    EndDate date NOT NULL CHECK (EndDate > GETDATE()),  
    Cancelled tinyint NOT NULL DEFAULT 0,  
    CONSTRAINT Conferences_pk PRIMARY KEY (ConferenceID),  
    CONSTRAINT StartDateEndDate_C CHECK (StartDate <= EndDate)  
);
```

- **ConfDays**

Tabela do której wpisywane są poszczególne dni konferencji. Dzień konferencji odwołuje się do konkretnej konferencji (ConferenceID), ma swoją datę (Date), oraz ilość miejsc (Slots). Warunki: ilość miejsc musi być większa niż 0.

```
CREATE TABLE ConfDays (  
    DayID int NOT NULL IDENTITY(1,1),  
    ConferenceID int NOT NULL,  
    Slots int CHECK (Slots > 0) NOT NULL,  
    Date date NOT NULL,  
    CONSTRAINT ConfDays_pk PRIMARY KEY (DayID)  
);
```

- **Prices**

Tabela reprezentująca różne progi cenowe na dni konferencji. Cena przypisana jest do dnia konferencji (przez DayID). Cena jednego biletu na dany dzień jest w polu (PricePerSlot). Tabela ta zawiera również zniżkę (Discount) jeśli dokonamy rezerwacji odpowiednio wcześniej (DaysBefore) oraz zniżkę dla studentów (StudentDiscount) Warunki: Cena, zniżka, oraz ilość dni przed nie mogą być ujemne.

```
CREATE TABLE Prices (  
    DayID int NOT NULL,  
    DaysBefore int NOT NULL CHECK (DaysBefore >= 0),  
    Discount decimal(3,2) CHECK (Discount >= 0) NOT NULL,  
    StudentDiscount decimal(3,2) CHECK (StudentDiscount >= 0) NOT NULL,  
    PricePerSlot decimal(15,2) CHECK (PricePerSlot >= 0) NOT NULL  
);
```

- **People**

Tabela reprezentująca osoby. Posiada pola: (FirstName, LAsTName, StudentID). Jeśli osoba nie jest studentem, pole StudentID ustawiamy na NULL.

```
CREATE TABLE People (  
    PersonID int NOT NULL IDENTITY(1,1),  
    FirstName nvarchar(50) NOT NULL,  
    LAsTName nvarchar(50) NOT NULL,  
    StudentID nvarchar(50) NULL,  
    CONSTRAINT People_pk PRIMARY KEY (PersonID)  
);
```

- **ConfReservation**

Tabela reprezentująca rezerwacje klientów na dany dzień konferencji. Podpięta jest pod dzień konferencji (DayID), dotyczy klienta (ClientID), i rezerwuje liczbę miejsc (Slots). Posiada datę rezerwacji, oraz pole (Cancelled) ustawione odpowiednio na 0 lub 1. Warunki: pole Slots nie może być ujemne.

```
CREATE TABLE ConfReservation (  
    ConfResID int NOT NULL IDENTITY(1,1),  
    ClientID int NOT NULL,  
    DayID int NOT NULL,  
    ReservationDate DATETIME NOT NULL,  
    Slots int CHECK (Slots > 0) NOT NULL,  
    Cancelled tinyint NOT NULL DEFAULT 0,  
    CONSTRAINT ConfReservation_pk PRIMARY KEY (ConfResID)  
);
```

- **Workshops**

Tabela reprezentująca warsztat. Warsztat posiada opis (Description), odbywa się w przedziale czasowym (StartTime, EndTime), w dniu konferencji (DayID), maksymalnie może w nim uczestniczyć ilość osób, określona przez (Slots). Koszt uczestnictwa w warsztacie to (PricePerSlot). Warsztat może zostać anulowany (Cancelled).

Warunki: Ilość miejsc nie może być ujemna, data rozpoczęcia i zakończenia późniejsza od obecnej daty, cena nie może być ujemna, data zakończenia późniejsza niż data rozpoczęcia.

```
CREATE TABLE Workshops (  
    WorkshopID int NOT NULL IDENTITY(1,1),  
    DayID int NOT NULL,  
    Description varchar(255) NULL,  
    Slots int NOT NULL CHECK (Slots > 0),  
    StartTime datetime NOT NULL CHECK (StartTime > GETDATE()),  
    EndTime datetime NOT NULL CHECK (EndTime > GETDATE()),  
    PricePerSlot decimal(15,2) CHECK (PricePerSlot >= 0) NOT NULL,  
    Cancelled tinyint NOT NULL DEFAULT 0,  
    CONSTRAINT Workshops_pk PRIMARY KEY (WorkshopID),  
    CONSTRAINT StartTimeEndTime_W CHECK (StartTime < EndTime)  
);
```

- **WorkReservation**

Tabela reprezentująca rezerwacje klientów na dany warsztat. Podpięta jest pod rezerwację konferencji (ConfResID). Dotyczy warsztatu (WorkshopID), i rezerwuje liczbę miejsc (Slots). Rezerwacja warsztatu może zostać anulowana (Cancelled).

Warunki: Liczba miejsc nie może być ujemna.

```
CREATE TABLE WorkReservation (  
    WorkResID int NOT NULL IDENTITY(1,1),  
    WorkshopID int NOT NULL,  
    ConfResID int NOT NULL,  
    ReservationDate DATETIME NOT NULL,  
    Slots int NOT NULL CHECK (Slots > 0),  
    Cancelled tinyint NOT NULL DEFAULT 0,  
    CONSTRAINT WorkReservation_pk PRIMARY KEY (WorkResID)  
);
```

- **PaymentDone**

Tabela zawierająca informacje o wpłatach. Zawiera (ConfResID) której dotyczy, oraz wartość wpłaty (Amount) i datę (Date). Warunki: Wartość wpłaty nie może być ujemna.

```
CREATE TABLE PaymentDone (  
    PaymentID int NOT NULL IDENTITY(1,1),  
    ConfResID int NOT NULL,  
    Amount decimal(15,2) CHECK (PaymentDone.Amount > 0) NOT NULL,  
    Date date NOT NULL,  
    CONSTRAINT PaymentDone_pk PRIMARY KEY (PaymentID)  
);
```

- **ConfResDetails**

Tabela łącząca konkretne osoby (PersonID) z poszczególną rezerwacją (ConfResID).

```
CREATE TABLE ConfResDetails (  
    ConfResID int NOT NULL,  
    PersonID int NOT NULL,  
    StudentID nvarchar(50) NULL  
);
```

- **WorkResDetails**

Tabela łącząca konkretne osoby (PersonID) z poszczególnej rezerwacji (ConfResID) z rezerwacją warsztatu (WorkResID).

```
CREATE TABLE WorkResDetails (  
    WorkResID int NOT NULL,  
    PersonID int NOT NULL,  
    ConfResID int NOT NULL  
);
```

4. Widoki:

- **Niepełne zgłoszenia**

Pokazuje zgłoszenia które jeszcze nie są wypełnione

```
CREATE VIEW Niepelne_zgloszenia  
AS  
    SELECT cr.ConfResID, c.Name AS [Nazwa klienta], c.Phone,  
        conf.Name AS [Nazwa Konferencji], cd.Date AS [Data Konferencji],  
        cr.Slots AS [Ile powinno być],  
        cr.Slots - (SELECT COUNT(*) FROM ConfResDetails AS crd WHERE  
            crd.ConfResID = cr.ConfResID) AS [Ilu brakuje]  
  
    FROM ConfReservation AS cr  
    JOIN Clients AS c  
    ON cr.ClientID = c.ClientID  
    JOIN ConfDays AS cd  
    ON cr.DayID = cd.DayID  
    JOIN Conferences AS conf  
    ON cd.ConferenceID = conf.ConferenceID  
  
    WHERE cr.Slots - (SELECT COUNT(*) FROM ConfResDetails AS crd  
        WHERE crd.ConfResID = cr.ConfResID) >0  
    AND cr.Cancelled = 0  
  
GO
```

- **Najpopularniejsze konferencje (top 20)**

```
CREATE VIEW Najpopularniejsze_konferencje  
AS  
    SELECT TOP 20 conf.ConferenceID, conf.Name, COUNT(*) AS [Ilosc chetnych]  
    FROM Conferences AS conf  
    LEFT JOIN ConfDays AS cd  
    ON conf.ConferenceID = cd.ConferenceID  
    LEFT JOIN ConfReservation AS cr  
    ON cr.DayID = cd.DayID  
    JOIN ConfResDetails AS crd  
    ON crd.ConfResID = cr.ConfResID  
    GROUP BY conf.ConferenceID, conf.Name  
    ORDER BY COUNT (*) DESC  
  
GO
```

- **Najpopularniejsze warsztaty (top 20)**

```
CREATE VIEW Najpopularniejsze_warsztaty
AS
    SELECT TOP 20 work.WorkshopID, work.Description, work.StartTime, work.EndTime,
    COUNT(*) AS [Ilosc chetnych]
    FROM Workshops AS work
    LEFT JOIN WorkReservation AS wr
    ON work.WorkshopID = wr.WorkshopID
    JOIN WorkResDetails AS wrd
    ON wr.WorkResID = wrd.WorkResID
    GROUP BY work.WorkshopID, work.Description, work.StartTime, work.EndTime
    ORDER BY COUNT (*) DESC
GO
```

- **Najczęściej korzystający z usług**

```
CREATE VIEW Najczesciej_korzystajacy_z_uslug
AS
    SELECT Name AS [Client],
    (SELECT COUNT(*)
    FROM ConfReservation
    WHERE (Clients.ClientID = ClientID) AND ConfReservation.Cancelled = 0) AS [How
    many times:]
    FROM Clients
    GROUP BY ClientID, Name
    ORDER BY [Ile:] DESC
GO
```

- **Anulowane konferencje**

```
CREATE VIEW Anulowane_konferencje
AS
    SELECT *
    FROM Conferences AS c
    WHERE c.Cancelled=1
GO
```

- **Anulowane rezerwacje na konferencje**

```
CREATE VIEW Anulowane_konf_rezerwacje
AS
    SELECT *
    FROM ConfReservation AS cr
    WHERE cr.Cancelled=1
GO
```

- **Anulowane warsztaty**

```
CREATE VIEW Anulowane_warsztaty
AS
    SELECT *
    FROM Workshops AS w
    WHERE w.Cancelled=1
GO
```

- **Anulowane rezerwacje na warsztaty**

```
CREATE VIEW Anulowane_wars_rezerwacje
AS
    SELECT *
    FROM WorkReservation AS wr
    WHERE wr.Cancelled=1

GO
```

- **Klienci prywatni**

```
CREATE VIEW Klienci_prywatni
AS
    SELECT *
    FROM Clients AS c
    WHERE c.Company=0

GO
```

- **Firmy**

```
CREATE VIEW Klienci_firmowi
AS
    SELECT *
    FROM Clients AS c
    WHERE c.Company=1

GO
```

5. Funkcje, Procedury i Triggery:

- **Czy rezerwacja konferencji jest ok**

Sprawdza czy ilość osób nie została przekroczona

```
CREATE TRIGGER Czy_konf_rez_ok
ON ConfReservation
AFTER INSERT,UPDATE AS
BEGIN
    IF EXISTS (SELECT [TAK]
                FROM inserted
                JOIN ConfDays AS cd
                ON inserted.DayId =cd.DayId
                JOIN ConfReservation AS cr ON cd.DayID = cr.DayID
                WHERE inserted.DayID = cr.DayID
                GROUP BY cr.DayID, cd.Slots
                HAVING SUM(cr.Slots) > cd.Slots
            )
    BEGIN
        RAISERROR ('Nie mozna wpisac wiecej osob na ta konferencje. Brak miejsc', 16,
        1)
        ROLLBACK TRANSACTION
    END
END
```

- **Czy rezerwacja warsztatu jest ok**

Sprawdza czy ilość osób nie została przekroczona

```
CREATE TRIGGER Czy_wars_rez_ok
ON WorkReservation
AFTER INSERT,UPDATE AS
BEGIN
    IF EXISTS (SELECT [TAK]
                FROM inserted
                JOIN Workshops AS w
                ON inserted.WorkshopID = w.WorkshopID
                JOIN WorkReservation AS wr ON wr.WorkshopID = w.WorkshopID
                WHERE inserted.WorkshopID = wr.WorkshopID
                GROUP BY wr.WorkshopID, w.Slots
                HAVING SUM(wr.Slots) > w.Slots
               )
    BEGIN
        RAISERROR ('Nie mozna wpisac wiecej osob na ten warsztat. Brak miejsc', 16, 1)
        ROLLBACK TRANSACTION
    END
END
```

- **Czy osoba jest już na konferencji**

Sprawdza czy dodawana osoba nie jest już zapisana na konferencje

```
CREATE TRIGGER Czy_osoba_jest_konf
ON ConfResDetails
AFTER INSERT,UPDATE AS
BEGIN
    IF (SELECT COUNT(*)
        FROM inserted
        JOIN ConfReservation AS cr
        ON inserted.ConfResID = cr.ConfResID
        JOIN ConfResDetails AS crd
        ON crd.ConfResID = cr.ConfResID
        WHERE inserted.PersonID = crd.PersonID
       )>1
    BEGIN
        RAISERROR ('Osoba juz jest na konferencji', 16, 1)
        ROLLBACK TRANSACTION
    END
END
```


- **Czy osoba zapisująca się na warsztat jest na konferencji**
Sprawdza czy dodawana do warsztatu osoba jest zapisana na odpowiednią konferencję

```
CREATE TRIGGER czy_osoba_wars_jest_konf
ON WorkResDetails
AFTER INSERT,UPDATE AS
BEGIN
    IF (SELECT COUNT(*)
        FROM inserted AS wrdi
        JOIN WorkReservation AS ws
        ON wrdi.WorkResID = ws.WorkResID
        JOIN
        Workshops AS w ON w.WorkshopID = ws.WorkshopID
        JOIN
        ConfReservation AS cr ON cr.ConfResID=ws.ConfResID
        JOIN
        ConfResDetails AS crd ON cr.ConfResID = crd.ConfResID
        WHERE wrdi.PersonID=crd.PersonID
        )=0
    BEGIN
        RAISERROR ('Osoba nie jest zapisana na konferencje na ktorej jest ten warsztat.', 16,
        1)
        ROLLBACK TRANSACTION
    END
END
GO
```

- **Czy osoba chcąc zapisać się na warsztat nie bierze w tym czasie udziału w innym**

```
CREATE TRIGGER Czy_nie_koliduja_warsztaty
ON WorkResDetails
AFTER INSERT,UPDATE AS
BEGIN
    IF( SELECT COUNT(*)
        FROM inserted AS iwrđ
        JOIN WorkReservation AS wr
        ON wr.WorkResID = iwrđ.WorkResID
        JOIN Workshops AS w
        ON w.WorkshopID = wr.WorkshopID
        JOIN WorkResDetails AS wrđ
        ON iwrđ.PersonID = wrđ.PersonID
        JOIN WorkReservation AS wr2
        ON wrđ.WorkResID = wr2.WorkResID
        JOIN Workshops AS w2
        ON w2.WorkshopID = wr2.WorkshopID
        WHERE (w.StartTime < w2.EndTime AND w.EndTime > w2.StartTime)
        )>1
    BEGIN
        RAISERROR ('Ta osoba bierze już udział w innym warsztacie w tych godzinach.',
        16, 1)
        ROLLBACK TRANSACTION
    END
END
```

- **Osoby zapisane na dany dzień konferencji**

```
CREATE PROCEDURE Osoby_dzien_konf
(
    @dayid int
)
AS
    SELECT DISTINCT p.PersonID, p.FirstName, p.LAStName, p.StudentID
    FROM People AS p
    JOIN ConfResDetails AS crd
    ON p.PersonID = crd.PersonID
    JOIN ConfReservation AS cr
    ON cr.ConfResID = crd.ConfResID
    WHERE cr.DayID = @dayid AND cr.Cancelled = 0
GO
```

- **Osoby zapisane na dany warsztat**

```
CREATE PROCEDURE Osoby_warsztat
(
    @workshopid int
)
AS
    SELECT DISTINCT p.PersonID, p.FirstName, p.LastName, p.StudentID
    FROM WorkReservation AS wr
    JOIN WorkResDetails AS wrd
    ON wrd.WorkResID = wr.WorkResID
    JOIN ConfResDetails AS crd
    ON wrd.ConfResID = crd.ConfResid
    JOIN People AS p
    ON crd.PersonID = p.PersonID
    WHERE wr.WorkshopID = @workshopid AND wr.Cancelled = 0
    ORDER BY p.PersonID
GO
```

- **Generuj ID**

Generuje ID dla osób na dany dzień konferencji

```
CREATE PROCEDURE Generuj_id
(
    @dayid int
)
AS
    SELECT p.FirstName, p.LAStName,
    CAST(cr.DayID AS nvarchar(5)) + '-' + CAST(cr.ConfResID AS nvarchar(5)) + '-'
    + CAST(p.PersonID AS nvarchar(5)) + SUBSTRING(p.FirstName, 0,5) +
    SUBSTRING(p.LAStName,0,5) + '-' + SUBSTRING(c.Name, 0,4) AS [Identyfikator]
    FROM People AS p
    JOIN ConfResDetails AS crd
    ON p.PersonID = crd.PersonID
    JOIN ConfReservation AS cr
    ON crd.ConfResID = cr.ConfResID
    JOIN Clients AS c
    ON cr.ClientID = c.ClientID
    WHERE cr.DayID = @dayid
GO
```

- **Moje Konferencje**

Generuje dla danej osoby listę

```
CREATE PROCEDURE Moje_konferencje
(
    @personid int
)
AS
    SELECT p.FirstName AS [Imie], p.LAStName AS [Nazwisko], c.Name AS [Nazwa
konferencji]
    FROM People AS p
    JOIN ConfResDetails AS crd
    ON p.PersonId= crd.PersonID
    JOIN ConfReservation AS cr
    ON crd.ConfResID = cr.ConfResID
    JOIN ConfDays AS cd
    ON cr.DayID = cd.DayID
    JOIN Conferences AS c
    ON cd.ConferenceID = c.ConferenceID
    WHERE p.PersonID = @personid
GO
```

- **Moje Warsztaty**

```
CREATE PROCEDURE Moje_warsztaty
(
    @personid int
)
AS
    SELECT p.FirstName AS [Imie], p.LAStName AS [Nazwisko], w.Description AS
[Nazwa Warsztatu]
    FROM People AS p
    JOIN WorkResDetails AS wrd
    ON p.PersonID = wrd.PersonID
    JOIN WorkReservation AS wr
    ON wrd.WorkResId = wr.WorkResId
    JOIN Workshops AS w
    ON wr.WorkshopID = w.WorkshopID
    WHERE p.PersonID = @personid
GO
```

- **Płatności konferencja**

Wyświetla płatności dla danej konferencji

```
CREATE PROCEDURE Platnosci_konferencja
(
    @conferenceid int
)
AS
    SELECT c.Name AS [Nazwa klienta], conf.Name AS [Nazwa konferencji], cd.Date AS
[Dzień konferencji], SUM(pd.Amount) AS [Płatności Dokonane]
    FROM Conferences AS conf
    JOIN ConfDays AS cd
    ON conf.ConferenceId = cd.ConferenceID
    JOIN ConfReservation AS cr
    ON cd.DayID = cr.DayID
    JOIN Clients AS c
    ON cr.ClientID = c.ClientID
    JOIN PaymentDone AS pd
    ON cr.ConfResID = pd.ConfResID
    WHERE conf.ConferenceID = @ conferenceid AND cr.Cancelled = 0
    GROUP BY c.Name, conf.Name, cd.Date
GO
```

- **Płatności rezerwacja**

Wyświetla płatności dla danej rezerwacji

```
CREATE PROCEDURE Platnosci_rezerwacja
(
    @reservationid int
)
AS
    SELECT c.Name AS [Nazwa klienta], conf.Name AS [Nazwa konferencji], cd.Date AS
    [Dzień konferencji], SUM(pd.Amount) AS [Płatności Dokonane]
    FROM Conferences AS conf
    JOIN ConfDays AS cd
    ON conf.ConferenceID = cd.ConferenceID
    JOIN ConfReservation AS cr
    ON cd.DayID = cr.DayID
    JOIN Clients AS c
    ON cr.ClientID = c.ClientID
    JOIN PaymentDone AS pd
    ON cr.ConfResID = pd.ConfResID
    WHERE cr.ConfResID = @reservationid AND cr.Cancelled = 0
    GROUP BY c.Name, conf.Name, cd.Date
GO
```

- **Progi cenowe**

Wyświetla progi cenowe dla danej konferencji

```
CREATE PROCEDURE Progi_cenowe
(
    @confid int
)
AS
    SELECT pr.DayID, pr.DaysBefore, FORMAT(pr.PricePerSlot*(1-pr.Discount), 'N',
    'en-us') AS [Cena normalna],
    FORMAT(pr.PricePerSlot*(1-pr.StudentDiscount), 'N', 'en-us') AS [Cena
    studencka] FROM Prices AS pr
    JOIN ConfDays AS cd
    ON pr.DayID = cd.DayID
    WHERE cd.ConferenceID = @confid
GO
```

- **Osoby z danej firmy/osoby prywatnej**

Wyświetla osoby z firmy o podanym ID

```
CREATE PROCEDURE Osoby_firma
(
    @clientid int
)
AS
    SELECT p.PersonID, p.FirstName, p.LAStName, p.StudentID
    FROM People AS p
    JOIN ConfResDetails AS crd
    ON p.PersonID = crd.PersonID
    JOIN ConfReservation AS cr
    ON crd.ConfResID = cr.ConfResID
    JOIN Clients AS c
    ON cr.ClientID = c.ClientID
    WHERE c.ClientID = @clientid
GO
```

- **Dodaj prywatnego klienta**

Imię, tel, adres, miasto, kod, kraj, studentid

```
CREATE PROCEDURE Dodaj_klient_prywatny
(
    @name nvarchar(50),
    @phone nvarchar(12),
    @adres nvarchar(50),
    @city nvarchar(50),
    @postcode nvarchar(6),
    @country nvarchar(50),
    @studentid nvarchar(50)
)
AS
    INSERT INTO Clients VALUES(0, @name, @phone, @adres, @city, @postcode,
    @country, @studentid, NULL)
GO
```

- **Dodaj firmę**

Nazwa, tel, adres, miasto, kod, kraj, nip

```
CREATE PROCEDURE Dodaj_klient_firma
(
    @name nvarchar(50),
    @phone nvarchar(12),
    @adres nvarchar(50),
    @city nvarchar(50),
    @postcode nvarchar(6),
    @country nvarchar(50),
    @nip nvarchar(50)
)
AS
    INSERT INTO Clients VALUES(1, @name, @phone, @adres, @city, @postcode,
    @country, NULL, @nip)
GO
```

- **Zmień dane klienta prywatnego**

```
CREATE PROCEDURE Zmien_dane_prywatny
(
    @id int,
    @name nvarchar(50)=NULL,
    @phone nvarchar(12)=NULL,
    @adres nvarchar(50)=NULL,
    @city nvarchar(50)=NULL,
    @postcode nvarchar(6)=NULL,
    @country nvarchar(50)=NULL,
    @studentid nvarchar(50)=NULL
)
AS
    IF @name IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.Name = @name
        WHERE Clients.ClientID = @id AND Clients.Company = 0
    END

    IF @phone IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.Phone = @phone
        WHERE Clients.ClientID = @id AND Clients.Company = 0
    END

    IF @adres IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.Address = @adres
        WHERE Clients.ClientID = @id AND Clients.Company = 0
    END

    IF @city IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.City = @city
        WHERE Clients.ClientID = @id AND Clients.Company = 0
    END

    IF @postcode IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.PostalCode = @postcode
        WHERE Clients.ClientID = @id AND Clients.Company = 0
    END

    IF @country IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.Country = @country
        WHERE Clients.ClientID = @id AND Clients.Company = 0
    END

    IF @studentid IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.StudentID = @studentid
        WHERE Clients.ClientID = @id AND Clients.Company = 0
    END

GO
```

- **Zmień dane firmy**

```
CREATE PROCEDURE Zmien_dane_firma
(
    @id int,
    @name nvarchar(50)=NULL,
    @phone nvarchar(12)=NULL,
    @adres nvarchar(50)=NULL,
    @city nvarchar(50)=NULL,
    @postcode nvarchar(6)=NULL,
    @country nvarchar(50)=NULL,
    @nip nvarchar(50)=NULL
)
AS
    IF @name IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.Name = @name
        WHERE Clients.ClientID = @id AND Clients.Company = 1
    END

    IF @phone IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.Phone = @phone
        WHERE Clients.ClientID = @id AND Clients.Company = 1
    END

    IF @adres IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.Address = @adres
        WHERE Clients.ClientID = @id AND Clients.Company = 1
    END

    IF @city IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.City = @city
        WHERE Clients.ClientID = @id AND Clients.Company = 1
    END

    IF @postcode IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.PostalCode = @postcode
        WHERE Clients.ClientID = @id AND Clients.Company = 1
    END

    IF @country IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.Country = @country
        WHERE Clients.ClientID = @id AND Clients.Company = 1
    END

    IF @nip IS NOT NULL
    BEGIN
        UPDATE Clients
        SET Clients.NIP = @nip
        WHERE Clients.ClientID = @id AND Clients.Company = 1
    END

GO
```

- **Dodaj osobę**

Imię, nazwisko, studentid

```
CREATE PROCEDURE Dodaj_osoba
(
    @firstname nvarchar(50),
    @sname nvarchar(50),
    @studentid nvarchar(50)
)
AS
    INSERT INTO People VALUES(@firstname, @sname, @studentid)
GO
```

- **Zmień studentID**

IDosoby, IDstudenta

```
CREATE PROCEDURE Zmien_student
(
    @id int,
    @studentid nvarchar(50)
)
AS
    UPDATE People
    SET People.StudentID = @studentid
    WHERE People.PersonID = @id
GO
```

- **Dodaj konferencję**

Nazwa, opis, start, koniec, liczba miejsc

```
CREATE PROCEDURE Dodaj_konferencje
(
    @name nvarchar(50),
    @desc text,
    @start date,
    @end date,
    @slots int
)
AS
    INSERT INTO Conferences VALUES (@name, @desc, @start, @end, 0);
    DECLARE @days int = DATEDIFF(DAY, @start, @end) + 1;
    DECLARE @cnt int = 0;
    DECLARE @currentDate date = @start;
    DECLARE @confId int = (SELECT ConferenceID FROM Conferences WHERE Name = @name
    AND StartDate = @start AND EndDate = @end);
    WHILE @cnt < @days
    BEGIN
        INSERT INTO ConfDays VALUES(@confId, @slots, @currentDate);
        SET @currentDate = DATEADD(DAY, 1, @currentDate);
        SET @cnt = @cnt + 1;
    END;
GO
```

- **Zmień ilość miejsc na konferencji**

```
CREATE PROCEDURE zmien_ilosc_miejsc_konf
(
    @dayid int,
    @slots int
)
AS
    UPDATE ConfDays
    SET ConfDays.Slots = @slots
    WHERE ConfDays.DayID = @dayid
GO
```


- **Dodaj rezerwacje na konferencję**

KlientID, DayID, liczba miejsc

```
CREATE PROCEDURE Dodaj_rezerwacja_konf
(
    @clientid int,
    @dayid int,
    @slots int
)
AS
    INSERT INTO ConfReservation VALUES (@clientid, @dayid, GETDATE(), @slots, 0)
GO
```

- **Dodaj osobę do rezerwacji na konferencję**

IDKonferencji, IDOsoby

```
CREATE PROCEDURE Dodaj_osoba_do_rez_konf
(
    @confresid int,
    @personid int
)
AS
    INSERT INTO ConfResDetails VALUES(@confresid, @personid, (SELECT StudentID
    FROM People WHERE PersonID = @personid))
GO
```

- **Dodaj próg cenowy**

IDDnia, ile dni przed, zniżka studencka, cena normalna

```
CREATE PROCEDURE Dodaj_prog_cenowy
(
    @dayid int,
    @daysbefore int,
    @disc decimal(3,2),
    @studentdisc decimal(3,2),
    @price money
)
AS
    INSERT INTO Prices VALUES(@dayid, @daysbefore, @disc, @studentdisc, @price)
GO
```

- **Dodaj warsztat**

IDDnia, opis, liczba miejsc, start, koniec, cena

```
CREATE PROCEDURE Dodaj_warsztat
(
    @dayid int,
    @desc text,
    @slots int,
    @starttime datetime,
    @endtime datetime,
    @price money
)
AS
    INSERT INTO Workshops VALUES(@dayid, @desc, @slots, @starttime, @endtime,
    @price, 0)
GO
```

- **Dodaj rezerwacje na warsztat**

IDwarsztatu, IDRezerwacji na konferencję , liczba miejsc

```
CREATE PROCEDURE Dodaj_rezerwacja_wars
(
    @workid int,
    @confresid int,
    @slots int
)
AS
    INSERT INTO WorkReservation VALUES (@workid, @confresid, GETDATE(), @slots, 0)
GO
```

- **Zmień ilość miejsc na warsztacie**

IDWarsztatu, liczba miejsc

```
CREATE PROCEDURE Zmien_miejsca_wars
(
    @workid int,
    @slots int
)
AS
    UPDATE Workshops
    SET     Workshops.Slots = @slots
    WHERE WorkshopID = @workid
GO
```

- **Dodaj osobę do rezerwacji na warsztat**

IDRezerwacji, IDOsoby

```
CREATE PROCEDURE Dodaj_osoba_do_rez_wars
(
    @workresid int,
    @personid int
)
AS
    INSERT INTO WorkResDetails VALUES(@workresid, @personid, (SELECT ConfResID
    FROM WorkReservation WHERE WorkResId = @workresid))
GO
```

- **Anuluj konferencję**

```
CREATE PROCEDURE Anuluj_konferencja
(
    @confid int
)
AS
    --konferencja
    UPDATE Conferences
    SET Conferences.Cancelled = 1
    WHERE ConferenceID = @confid

    --rezerwacje konferencja
    UPDATE cr
    SET cr.Cancelled = 1
    FROM ConfReservation AS cr
        JOIN ConfDays AS cd
        ON cr.DayID = cd.DayID
        JOIN Conferences AS c
        ON cd.ConferenceID = c.ConferenceID
    WHERE c.ConferenceID = @confid

    --warsztaty
    UPDATE w
    SET w.Cancelled = 1
    FROM WorkShops AS w
        JOIN ConfDays AS cd
        ON w.DayID = cd.DayID
        JOIN Conferences AS c
        ON cd.ConferenceID = c.ConferenceID
    WHERE c.ConferenceID = @confid

    --rezerwacje warsztaty
    UPDATE wr
    SET wr.Cancelled = 1
    FROM WorkReservation AS wr
        JOIN ConfReservation AS cr
        ON wr.ConfResID = cr.ConfResID
        JOIN ConfDays AS cd
        ON cr.DayID = cd.DayID
        JOIN Conferences AS c
        ON cd.ConferenceID = c.ConferenceID
    WHERE c.ConferenceID = @confid

GO
```

- **Anuluj rezerwację na konferencję**

```
CREATE PROCEDURE Anuluj_rezerwacja_konf
(
    @confresid int
)
AS
    --konferencja rezerwacja
    UPDATE ConfReservation
    SET ConfReservation.Cancelled = 1
    WHERE ConfResID = @confresid

    --warsztat rezerwacja
    UPDATE WorkReservation
    SET WorkReservation.Cancelled = 1
    WHERE ConfResID = @confresid

GO
```

- **Anuluj warsztat**

```
CREATE PROCEDURE Anuluj_warsztat
(
    @workid int
)
AS
    --warsztat
    UPDATE Workshops
    SET Workshops.Cancelled = 1
    WHERE WorkshopID = @workid

    --workreservation
    UPDATE WorkReservation
    SET WorkReservation.Cancelled = 1
    WHERE WorkshopID = @workid

GO
```

- **Anuluj rezerwację na warsztat**

```
CREATE PROCEDURE Anuluj_rezerwacja_warsztat
(
    @workresid int
)
AS
    UPDATE WorkReservation
    SET WorkReservation.Cancelled = 1
    WHERE WorkResID = @workresid

GO
```

- **Lista osób z danej rezerwacji na konferencję**

```
CREATE PROCEDURE Pokaz_rezerwacja_konf
(
    @confresid int
)
AS
    SELECT p.FirstName, p.LAStName, p.StudentID, cr.Cancelled
    FROM People AS p
    JOIN ConfResDetails AS crd
    ON p.PersonID = crd.PersonID
    JOIN ConfReservation AS cr
    ON crd.ConfResID = cr.ConfResID
    WHERE cr.ConfResID = @confresid

GO
```

- **Lista osób z danej rezerwacji na warsztat**

```
CREATE PROCEDURE Pokaz_rezerwacja_wars
(
    @workresid int
)
AS
    SELECT p.FirstName, p.LAStName, p.StudentID, wr.Cancelled
    FROM People AS p
    JOIN WorkResDetails AS wrd
    ON p.PersonID = wrd.PersonID
    JOIN WorkReservation AS wr
    ON wrd.WorkResID = wr.WorkResID
    WHERE wr.WorkResID = @workresid

GO
```

- **Policz ile jest studentów w rezerwacji**

```
CREATE PROCEDURE Policz_ile_studentow
(
    @confresid int
)
AS
    RETURN (SELECT COUNT(*)
            FROM ConfReservation AS cr
            JOIN ConfResDetails AS crd
            ON cr.ConfResID = crd.ConfResID
            JOIN People AS p
            ON crd.PersonID = p.PersonID
            WHERE cr.ConfResID = @confresid AND p.StudentID IS NOT NULL)

GO
```

- **Policz ile jest nie studentów**

```
CREATE PROCEDURE Policz_ile_niestudentow
(
    @confresid int
)
AS
    RETURN (SELECT COUNT(*)
            FROM ConfReservation AS cr
            JOIN ConfResDetails AS crd
            ON cr.ConfResID = crd.ConfResID
            JOIN People AS p
            ON crd.PersonID = p.PersonID
            WHERE cr.ConfResID = @confresid AND p.StudentID IS NULL)

GO
```

- **Ile dni przed konferencją złożono rezerwację**

```
CREATE PROCEDURE Ile_dni_wczesniej
(
    @confresid int
)
AS
    IF(SELECT cr.ReservationDate
    FROM ConfReservation AS cr
    WHERE cr.ConfResID = @confresid
    )<(DATEADD(DAY, -30, (SELECT TOP 1 cd.date
    FROM ConfDays as cd
    JOIN ConfReservation as cr
    ON cr.DayID = cd.DayID
    WHERE cr.ConfResID = @confresid
    ORDER BY Date)))
    RETURN 30
    ELSE
    IF(SELECT cr.ReservationDate
    FROM ConfReservation AS cr
    WHERE cr.ConfResID = @confresid
    )<(DATEADD(DAY, -20, (SELECT TOP 1 cd.date
    FROM ConfDays as cd
    JOIN ConfReservation as cr
    ON cr.DayID = cd.DayID
    WHERE cr.ConfResID = @confresid
    ORDER BY Date)))
    RETURN 20
    ELSE
    IF(SELECT cr.ReservationDate
    FROM ConfReservation AS cr
    WHERE cr.ConfResID = @confresid
    )<(DATEADD(DAY, -10, (SELECT TOP 1 cd.date
    FROM ConfDays as cd
    JOIN ConfReservation as cr
    ON cr.DayID = cd.DayID
    WHERE cr.ConfResID = @confresid
    ORDER BY Date)))
    RETURN 10
    ELSE
    RETURN 0
```

- **Policz cenę rezerwacji warsztatu**

```
CREATE PROCEDURE Policz_cene_rezerwacji_wars
(
    @workresid int
)
AS
    SELECT SUM(wr.Slots*w.PricePerSlot)
    FROM WorkReservation AS wr
    JOIN Workshops AS w
    ON wr.WorkshopID = w.WorkshopID
    WHERE wr.WorkResID = @workresid

GO
```

- **Policz cenę rezerwacji konferencji**

```
CREATE PROCEDURE Policz_cene_rezerwacji_konf
(
    @confresid int
)
AS
    DECLARE @howManyStudents INT
    DECLARE @howManyNotStudents INT
    DECLARE @howManyDaysBefore INT
    EXEC @howManyStudents = Policz_ile_studentow @confresid
    EXEC @howManyNotStudents = Policz_ile_niestudentow @confresid
    EXEC @howManyDaysBefore = Ile_dni_wczesniej @confresid
    IF(@howManyDaysBefore != 0)
    (
        SELECT SUM(@howManyNotStudents*p.PricePerSlot*(1-p.Discount) +
            @howManyStudents*p.PricePerSlot*(1-p.StudentDiscount)*(1-p.Discount))
        FROM ConfReservation AS cr
        JOIN ConfDays AS cd
        ON cr.DayID = cd.DayID
        JOIN Prices AS p
        ON cd.DayID = p.DayID
        WHERE cr.ConfResID = @confresid AND p.DaysBefore = @howManyDaysBefore
    )
    ELSE
    (
        SELECT SUM(@howManyNotStudents*p.PricePerSlot +
            @howManyStudents*p.PricePerSlot*(1-p.StudentDiscount))
        FROM ConfReservation AS cr
        JOIN ConfDays AS cd
        ON cr.DayID = cd.DayID
        JOIN Prices AS p
        ON cd.DayID = p.DayID
        WHERE cr.ConfResID = @confresid AND p.DaysBefore = @howManyDaysBefore
    )
GO
```

6. Generator danych:

Mieliśmy przygotować zestaw danych dla naszego systemu bazodanowego, który będzie odpowiadał firmie działającej na przestrzeni trzech lat, gdzie firma organizuje średnio 2 konferencje w miesiącu trwające zwykle 2-3 dni i z średnią ilością warsztatów 4 na dzień. Ponadto na każdej konferencji powinno być zarejestrowanych średnio 200 osób.

Nasz generator został napisany w języku Java. Początkowo generuje bazę danych i firm, następnie generujemy konferencje, a w nich po kolei dni, warsztaty, poszczególne rezerwacje i odpowiednie płatności. Wszystkie dane generatora jakie chcemy ustalić wystarczy ustawić w pliku Data.java, reszta magii nie powinna interesować użytkownika :)

Aby spełnić wymagania ustawiliśmy generator tak, aby generował 100 klientów (AMOUNT_OF_CLIENTS), w tym zarówno firmy jak i ludzi. Ponadto generujemy 800 klientów normalnych (AMOUNT_OF_ADULTS), a także 400 studentów (AMOUNT_OF_STUDENTS). Po prostych rachunkach doszliśmy do wniosku, że ilość konferencji jakie należy wygenerować to 72 (AMOUNT_OF_CONFS). Reszta danych jest generowana bardziej losowo, ale generator daje nam możliwość ustalenia minimalnych i maksymalnych wartości jakie chcemy mieć dla odpowiednich danych. Na odpowiednich etapach korzystamy z list i tablic haszujących co zapewnia nam poprawność danych generowanych. Dane do przekazania do bazy są generowane do pliku .sql, w naszym przypadku Data.sql, nazwę także można zadeklarować w generatorze. Wygenerowane dane już są w pliku, nie będziemy zamieszczać ich tutaj, gdyż tych danych w naszym przypadku jest zazwyczaj ponad 70 tysięcy linijek tekstu. Pokażemy tutaj tylko fragmenty z wygenerowanego pliku.

--GENERATED CLIENTS:

```
INSERT INTO Clients VALUES (0,'Michał Kazmierczak', '+48797093067', 'Zamkowa 64', 'Bukowina
Tatrzańska', '98-220', 'Polska', '576406', NULL)
INSERT INTO Clients VALUES (1,'Activmed', '+48362595842', 'Kościuszki 36', 'Gdynia', '73-354',
'Polska', NULL, '0887768449')
INSERT INTO Clients VALUES (0,'Zbigniew Piątek', '+48103935037', 'Łąkowa 4', 'Bukowina Tatrzańska',
'65-171', 'Polska', '285900', NULL)
INSERT INTO Clients VALUES (0,'Marcin Włodarczyk', '+48193603699', 'Czarnowiejska 31', 'Bukowina
Tatrzańska', '11-746', 'Polska', '775957', NULL)
INSERT INTO Clients VALUES (1,'Activity', '+48550520014', 'Słowackiego 110', 'Warszawa', '43-414',
'Polska', NULL, '6286944331')
INSERT INTO Clients VALUES (1,'Vodafone', '+48618583454', 'Popiełuszki 73', 'Szczecin', '64-689',
'Polska', NULL, '4583035541')
INSERT INTO Clients VALUES (0,'Jacek Piątek', '+48999396495', 'Leśna 64', 'Jaworzna', '08-615',
'Polska', '216496', NULL)
INSERT INTO Clients VALUES (0,'Iwona Kopeć', '+48444470960', 'Słowackiego 36', 'Przemysł', '33-640',
'Polska', '086582', NULL)
INSERT INTO Clients VALUES (0,'Piotr Urban', '+48441263753', 'Szewska 47', 'Gdańsk', '45-836',
'Polska', '042503', NULL)
INSERT INTO Clients VALUES (0,'Tadeusz Mazur', '+48508698081', 'Łąkowa 64', 'Zamosc', '90-836',
'Polska', '506291', NULL)
INSERT INTO Clients VALUES (1,'Nokia', '+48654943606', 'Spółdzielców 93', 'Wieliczka', '92-303',
'Polska', NULL, '9338966087')
```

--GENERATED PEOPLE

```
INSERT INTO People VALUES ('Łukasz', 'Kolodziej', NULL)
INSERT INTO People VALUES ('Małgorzata', 'Tomczak', NULL)
INSERT INTO People VALUES ('Jolanta', 'Kazmierczak', NULL)
INSERT INTO People VALUES ('Stanisław', 'Kania', NULL)
INSERT INTO People VALUES ('Michał', 'Tomczyk', NULL)
INSERT INTO People VALUES ('Andrzej', 'Czaja', NULL)
INSERT INTO People VALUES ('Aleksandra', 'Cieslak', NULL)
INSERT INTO People VALUES ('Andrzej', 'Stepień', NULL)
INSERT INTO People VALUES ('Jan', 'Pająk', NULL)
```

--GENERATED STUDENTS:

```
INSERT INTO People VALUES ('Marek', 'Polak', '836240')
INSERT INTO People VALUES ('Mateusz', 'Stankiewicz', '609902')
INSERT INTO People VALUES ('Mateusz', 'Kopeć', '841208')
INSERT INTO People VALUES ('Jakub', 'Stasiak', '874061')
INSERT INTO People VALUES ('Marianna', 'Kurek', '926597')
INSERT INTO People VALUES ('Mariusz', 'Mazur', '059721')
INSERT INTO People VALUES ('Helena', 'Pająk', '540235')
INSERT INTO People VALUES ('Ryszard', 'Wrobel', '958485')
```

--CONFERENCE 32:

```
INSERT INTO Conferences VALUES ('Targi pracy', NULL, '2017-07-01', '2017-07-01', 0)
INSERT INTO ConfDays VALUES (32, 248, '2017-07-01')
INSERT INTO Prices VALUES (76, 30, 0.30, 0.50, 90)
INSERT INTO Prices VALUES (76, 20, 0.26, 0.46, 90)
INSERT INTO Prices VALUES (76, 10, 0.22, 0.42, 90)
INSERT INTO Workshops VALUES (76, 'Zrób to sam', 81, '2017-07-01 02:30', '2017-07-01 04:30', 48, 0)
INSERT INTO Workshops VALUES (76, 'Warsztat dyskusyjny', 54, '2017-07-01 04:30', '2017-07-01 06:00',
42, 0)
INSERT INTO Workshops VALUES (76, 'Warsztat dyskusyjny', 12, '2017-07-01 05:30', '2017-07-01 07:30',
56, 0)
INSERT INTO ConfReservation VALUES (39, 76, '2017-05-25', 4, 0)
INSERT INTO WorkReservation VALUES (264, 1493, '2017-05-25', 4, 0)
INSERT INTO ConfResDetails VALUES (1493, 255, NULL)
INSERT INTO WorkResDetails VALUES (844, 255, 1493)
INSERT INTO ConfResDetails VALUES (1493, 799, NULL)
INSERT INTO WorkResDetails VALUES (844, 799, 1493)
INSERT INTO ConfResDetails VALUES (1493, 639, NULL)
INSERT INTO WorkResDetails VALUES (844, 639, 1493)
INSERT INTO ConfResDetails VALUES (1493, 1168, (SELECT StudentID FROM People WHERE PersonID = 1168))
INSERT INTO WorkResDetails VALUES (844, 1168, 1493)
```



```

INSERT INTO PaymentDone VALUES (1493, 426.00, '2017-05-30')
INSERT INTO ConfReservation VALUES (94, 76, '2017-06-03', 6, 0)
INSERT INTO ConfResDetails VALUES (1494, 535, NULL)
INSERT INTO ConfResDetails VALUES (1494, 442, NULL)
INSERT INTO ConfResDetails VALUES (1494, 694, NULL)
INSERT INTO ConfResDetails VALUES (1494, 84, NULL)
INSERT INTO ConfResDetails VALUES (1494, 572, NULL)
INSERT INTO ConfResDetails VALUES (1494, 110, NULL)
INSERT INTO PaymentDone VALUES (1494, 199.80, '2017-06-06')
INSERT INTO PaymentDone VALUES (1494, 199.80, '2017-06-04')

```

--CONFERENCE 66:

```

INSERT INTO Conferences VALUES ('Targi gier', NULL, '2018-10-04', '2018-10-07', 0)
INSERT INTO ConfDays VALUES (66, 220, '2018-10-04')
INSERT INTO Prices VALUES (153, 30, 0.30, 0.50, 235)
INSERT INTO Prices VALUES (153, 20, 0.28, 0.48, 235)
INSERT INTO Prices VALUES (153, 10, 0.26, 0.46, 235)
INSERT INTO Workshops VALUES (153, 'Szkolenie niespodzianka', 32, '2018-10-04 01:30', '2018-10-04 03:30', 57, 0)
INSERT INTO Workshops VALUES (153, 'Dyskusje znanych osobistości', 69, '2018-10-04 02:30', '2018-10-04 04:30', 51, 0)
INSERT INTO Workshops VALUES (153, 'Prosty warsztat', 44, '2018-10-04 03:00', '2018-10-04 04:30', 23, 0)
INSERT INTO ConfReservation VALUES (16, 153, '2018-09-08', 8, 0)
INSERT INTO WorkReservation VALUES (535, 2956, '2018-09-08', 8, 0)
INSERT INTO ConfResDetails VALUES (2956, 334, NULL)
INSERT INTO WorkResDetails VALUES (1675, 334, 2956)
INSERT INTO ConfResDetails VALUES (2956, 165, NULL)
INSERT INTO WorkResDetails VALUES (1675, 165, 2956)
INSERT INTO ConfResDetails VALUES (2956, 452, NULL)
INSERT INTO WorkResDetails VALUES (1675, 452, 2956)
INSERT INTO ConfResDetails VALUES (2956, 41, NULL)
INSERT INTO WorkResDetails VALUES (1675, 41, 2956)
INSERT INTO ConfResDetails VALUES (2956, 8, NULL)
INSERT INTO WorkResDetails VALUES (1675, 8, 2956)
INSERT INTO ConfResDetails VALUES (2956, 303, NULL)
INSERT INTO WorkResDetails VALUES (1675, 303, 2956)
INSERT INTO ConfResDetails VALUES (2956, 152, NULL)
INSERT INTO WorkResDetails VALUES (1675, 152, 2956)
INSERT INTO ConfResDetails VALUES (2956, 1046, (SELECT StudentID FROM People WHERE PersonID = 1046))
INSERT INTO WorkResDetails VALUES (1675, 1046, 2956)
INSERT INTO PaymentDone VALUES (2956, 1714.60, '2018-09-08')
INSERT INTO ConfDays VALUES (66, 220, '2018-10-05')
INSERT INTO Prices VALUES (154, 30, 0.30, 0.50, 85)
INSERT INTO Prices VALUES (154, 20, 0.28, 0.48, 85)
INSERT INTO Prices VALUES (154, 10, 0.26, 0.46, 85)
INSERT INTO Workshops VALUES (154, NULL, 54, '2018-10-05 03:00', '2018-10-05 04:30', 48, 0)
INSERT INTO Workshops VALUES (154, NULL, 46, '2018-10-05 03:30', '2018-10-05 04:30', 48, 0)
INSERT INTO Workshops VALUES (154, 'Zrób to sam', 42, '2018-10-05 04:00', '2018-10-05 05:30', 42, 0)
INSERT INTO ConfReservation VALUES (19, 154, '2018-09-11', 4, 0)
INSERT INTO WorkReservation VALUES (538, 2976, '2018-09-11', 4, 0)
INSERT INTO ConfResDetails VALUES (2976, 608, NULL)
INSERT INTO WorkResDetails VALUES (1687, 608, 2976)
INSERT INTO ConfResDetails VALUES (2976, 162, NULL)
INSERT INTO WorkResDetails VALUES (1687, 162, 2976)
INSERT INTO ConfResDetails VALUES (2976, 633, NULL)
INSERT INTO WorkResDetails VALUES (1687, 633, 2976)
INSERT INTO ConfResDetails VALUES (2976, 237, NULL)
INSERT INTO WorkResDetails VALUES (1687, 237, 2976)
INSERT INTO PaymentDone VALUES (2976, 218.40, '2018-09-15')
INSERT INTO PaymentDone VALUES (2976, 218.40, '2018-09-16')

```

7. Role:

- **Administrator bazy** (obsługa bazy, rozwiązywanie ewentualnych problemów)
- **Pracownicy obsługi** (dostęp do widoków i procedur informacyjnych, dostęp do funkcji, obsługa ma udzielać pomocy właścicielowi)
- **Właściciel** (może tworzyć konferencje i warsztaty, usuwać/anulować własne wydarzenia, modyfikować ich dane, dostęp do wszystkich widoków i procedur związanych z utworzonymi konferencjami + wszystko to co klient)
- **Klient** (może robić/anulować rezerwacje na konferencje i warsztaty, zgłaszać dowolną ilość osób na nie, korzystać z funkcji mówiącej o jego płatnościach, użyć funkcji zwracającej informację ile jeszcze pozostało do zapłaty za daną konferencję + klient może być uczestnikiem)
- **Osoba** (może sprawdzać na co jest zapisana, może stać się klientem, korzystać z procedury do edycji własnych danych)