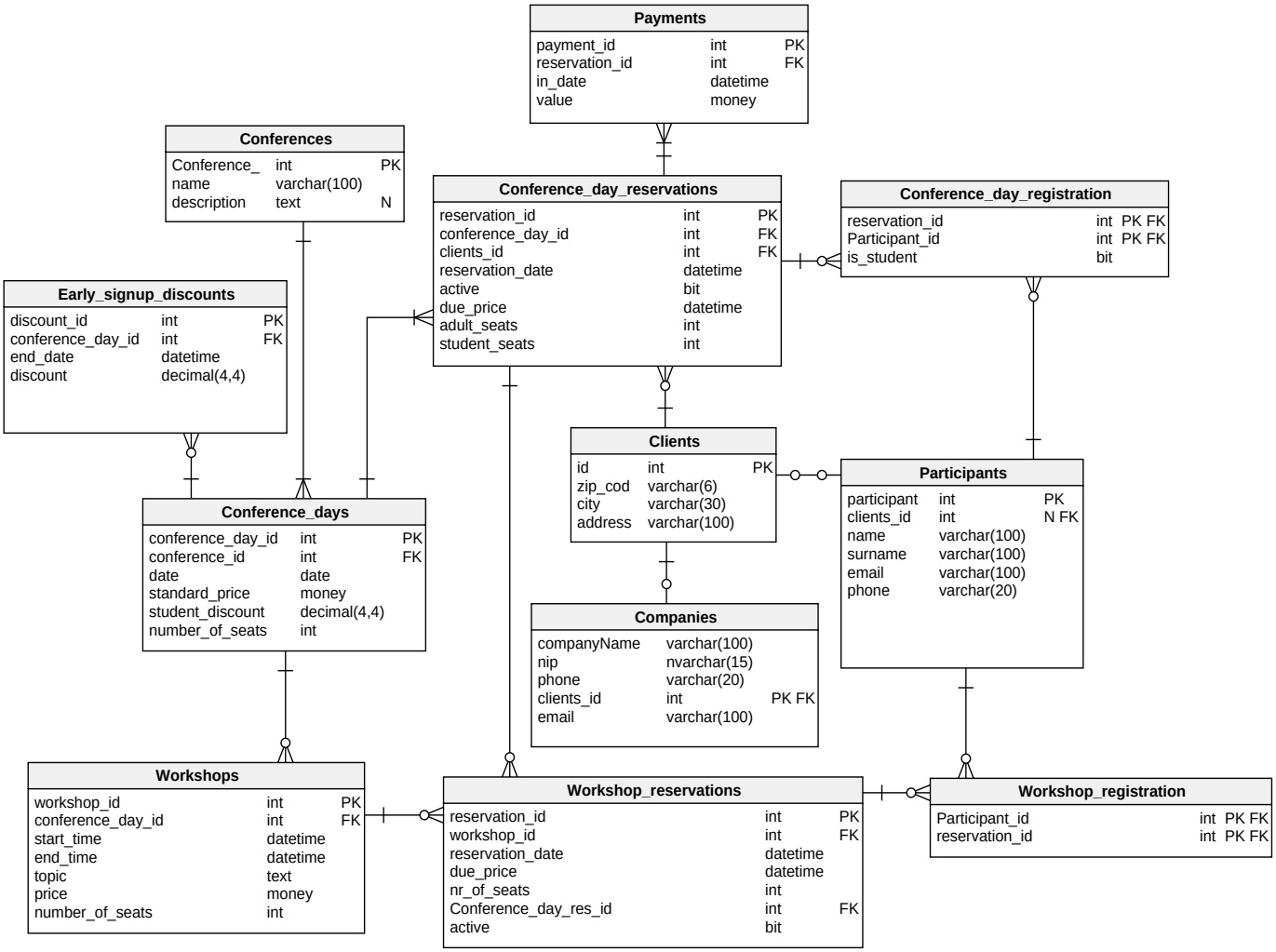
Podstawy baz danych
Projekt konferencje

Agnieszka Dutka, Maciek Trątnowiecki

AGH, Styczeń 2020

Objaśnienie schematu bazy

- Clients - Reprezentuje klientów chcących opłacić miejsca na konferencjach i warsztatach. Klientem może być zarówno firma, jak i osoba prywatna. W zależności od tego dane klienta reprezentowane są przez odpowiednią relację w bazie.
- Companies - Jeśli klient jest firmą, przechowuje jego dane.
- Participants - Jeśli klient jest osobą prywatną, przechowuje jego dane.
- Conferences - Reprezentuje konferencję z którą powiązane są odpowiednie dni konferencyjne, oraz warsztaty.
- Conference_days - Reprezentuje pojedynczy dzień konferencji. Powiązana jest z nim ustalona opłata za uczestnictwo. Zniżki obowiązujące w zależności od daty rejestracji zwarte są w relacji Early_Signup_Discounts.
- Early_Signup_Discounts - Odpowiada za informację o tabeli zniżek na dany dzień konferencyjny. Pojedyncza zniżka przechowywana jest w krotce z atrybutami w postaci procentowej obniżki ceny standardowej, oraz ostatniego dnia w którym obowiązuje.
- Conference_day_reservations - Realizuje rezerwacje na poszczególny dzień konferencji. Każda rezerwacja powiązana jest z klientem, który ją opłaca. Za powiązanie rezerwacji z uczestnikiem odpowiada osobna relacja. Zawiera także pole due_price określające termin płatności. Atrybut active odpowiada za możliwość rezygnacji z podjętej rezerwacji (uznaliśmy, że usuwanie krotki z bazy może nie być optymalnym rozwiązaniem, jako że zawarte w niej dane mogą jeszcze być przydatne z punktu widzenia logiki biznesowej). Atrybuty adult_seats i student_seats służą do liczenia kosztu podjęcia rezerwacji przed powiązaniem jej z uczestnikami konferencji.
- Conference_day_registration - Wiąże rezerwację z uczestnikami konferencji. Atrybut is_student informuje, czy danemu uczestnikowi przysługuje zniżka studencka.
- Payments - Przechowuje informacje o wpływach pieniężnych powiązanych z daną rejestracją.
- Workshops - Reprezentuje warsztaty odbywające się w trakcie odpowiednich dni konferencyjnych.
- Workshops_reservations - Opisuje rezerwacje na warsztaty w sposób analogiczny do rezerwacji na konferencje.
- Workshops_registrations - Łączy rezerwację z uczestnikami w sposób analogiczny do dni konferencyjnych.


Payments
payment_id int PK
reservation_id int FK
in_date datetime
value money
Conferences
Conference_ int PK
name varchar(100)
description text N
Conference_day_reservations
reservation_id int PK
conference_day_id int FK
clients_id int FK
reservation_date datetime
active bit
due_price datetime
adult_seats int
student_seats int
Conference_day_registration
reservation_id int PK FK
Participant_id int PK FK
is_student bit
Early_signup_discounts
discount_id int PK
conference_day_id int FK
end_date datetime
discount decimal(4,4)
Clients
id int PK
zip_cod varchar(6)
city varchar(30)
address varchar(100)
Participants
participant int PK
clients_id int N FK
name varchar(100)
surname varchar(100)
email varchar(100)
phone varchar(20)
Conference_days
conference_day_id int PK
conference_id int FK
date date
standard_price money
student_discount decimal(4,4)
number_of_seats int
Companies
companyName varchar(100)
nip nvarchar(15)
phone varchar(20)
clients_id int PK FK
email varchar(100)
Workshops
workshop_id int PK
conference_day_id int FK
start_time datetime
end_time datetime
topic text
price money
number_of_seats int
Workshop_reservations
reservation_id int PK
workshop_id int FK
reservation_date datetime
due_price datetime
nr_of_seats int
Conference_day_res_id int FK
active bit
Workshop_registration
Participant_id int PK FK
reservation_id int PK FK

Implementacja

```
-- tables
-- Table: Clients
CREATE TABLE Clients (
    id int  NOT NULL IDENTITY,
    zip_code varchar(6)  NOT NULL,
    city varchar(30)  NOT NULL,
    address varchar(100)  NOT NULL,
    CONSTRAINT Clients_pk PRIMARY KEY  (id)
);

-- Table: Companies
CREATE TABLE Companies (
    companyName varchar(100)  NOT NULL,
    nip nvarchar(15)  NOT NULL CHECK ((nip not like '%[^0-9]%') and (LEN(nip) = 10) and (nip
    not like '0%' or nip like '1%')),
    phone varchar(20)  NOT NULL,
    clients_id int  NOT NULL,
    email varchar(100)  NOT NULL CHECK (email like '%_@__%.__%'),
    CONSTRAINT unique_nip UNIQUE (nip),
    CONSTRAINT checkNip CHECK (dbo.IsValidNip(nip) = 1),
    CONSTRAINT Companies_pk PRIMARY KEY  (clients_id)
);

-- Table: Conference_day_registration
CREATE TABLE Conference_day_registration (
    reservation_id int  NOT NULL,
    Participant_id int  NOT NULL,
    is_student bit  NOT NULL DEFAULT 0,
    CONSTRAINT Conference_day_registration_pk PRIMARY KEY  (reservation_id,Participant_id)
);

-- Table: Conference_day_reservations
CREATE TABLE Conference_day_reservations (
    reservation_id int  NOT NULL IDENTITY,
    conference_day_id int  NOT NULL,
    clients_id int  NOT NULL,
    reservation_date datetime  NOT NULL DEFAULT GETDATE(),
    active bit  NOT NULL DEFAULT 1,
    due_price datetime  NOT NULL DEFAULT DATEADD(week, 2, GETDATE()) CHECK (due_price >=
    GETDATE()),
    adult_seats int  NOT NULL DEFAULT 0 CHECK (adult_seats >= 0),
    student_seats int  NOT NULL DEFAULT 0 CHECK (student_seats >= 0),
    CONSTRAINT Conference_day_reservations_pk PRIMARY KEY  (reservation_id)
);

-- Table: Conference_days
CREATE TABLE Conference_days (
    conference_day_id int  NOT NULL IDENTITY,
    conference_id int  NOT NULL,
    date date  NOT NULL DEFAULT GETDATE(),
    standard_price money  NOT NULL DEFAULT 0 CHECK (standard_price >= 0),
    student_discount decimal(4,4)  NOT NULL DEFAULT 0 CHECK (student_discount >= 0),
    number_of_seats int  NOT NULL DEFAULT 0 CHECK (number_of_seats >= 0),
    CONSTRAINT Conference_days_pk PRIMARY KEY  (conference_day_id)
);

-- Table: Conferences
CREATE TABLE Conferences (
    Conference_id int  NOT NULL IDENTITY,
    name varchar(100)  NOT NULL,
    description text  NULL,
    CONSTRAINT Conferences_pk PRIMARY KEY  (Conference_id)
);

-- Table: Early_signup_discounts
CREATE TABLE Early_signup_discounts (
    discount_id int  NOT NULL IDENTITY,
    conference_day_id int  NOT NULL,
    end_date datetime  NOT NULL,
    discount decimal(4,4)  NOT NULL DEFAULT 0,
    CONSTRAINT Early_signup_discounts_pk PRIMARY KEY  (discount_id)
);

-- Table: Participants
```

```
CREATE TABLE Participants (
    participant_id int  NOT NULL IDENTITY,
    clients_id int  NULL DEFAULT Null,
    name varchar(100)  NOT NULL,
    surname varchar(100)  NOT NULL,
    email varchar(100)  NOT NULL CHECK (email like '%_@__%.__%'),
    phone varchar(20)  NOT NULL,
    CONSTRAINT Participants_pk PRIMARY KEY  (participant_id)
);

-- Table: Payments
CREATE TABLE Payments (
    payment_id int  NOT NULL IDENTITY,
    reservation_id int  NOT NULL,
    in_date datetime  NOT NULL,
    value money  NOT NULL,
    CONSTRAINT Payments_pk PRIMARY KEY  (payment_id)
);

-- Table: Workshop_registration
CREATE TABLE Workshop_registration (
    Participant_id int  NOT NULL,
    reservation_id int  NOT NULL,
    CONSTRAINT Workshop_registration_pk PRIMARY KEY  (Participant_id,reservation_id)
);

-- Table: Workshop_reservations
CREATE TABLE Workshop_reservations (
    reservation_id int  NOT NULL IDENTITY,
    workshop_id int  NOT NULL,
    reservation_date datetime  NOT NULL DEFAULT GETDATE(),
    due_price datetime  NOT NULL DEFAULT DATEADD(week, 2, GETDATE()) CHECK (due_price >=
    GETDATE()),
    nr_of_seats int  NOT NULL DEFAULT 0 CHECK (nr_of_seats >= 0),
    Conference_day_res_id int  NOT NULL,
    active bit  NOT NULL DEFAULT 1,
    CONSTRAINT Workshop_reservations_pk PRIMARY KEY  (reservation_id)
);

-- Table: Workshops
CREATE TABLE Workshops (
    workshop_id int  NOT NULL IDENTITY,
    conference_day_id int  NOT NULL,
    start_time datetime  NOT NULL,
    end_time datetime  NOT NULL CHECK (end_time >= GETDATE()),
    topic text  NOT NULL,
    price money  NOT NULL CHECK (price >= 0),
    number_of_seats int  NOT NULL DEFAULT 0 CHECK (number_of_seats >= 0),
    CONSTRAINT Workshops_pk PRIMARY KEY  (workshop_id)
);

-- foreign keys
-- Reference: Companies_Clients (table: Companies)
ALTER TABLE Companies ADD CONSTRAINT Companies_Clients
    FOREIGN KEY (clients_id)
    REFERENCES Clients (id);

-- Reference: Conference_day_registration_Conference_day_reservations (table:
    Conference_day_registration)
ALTER TABLE Conference_day_registration ADD CONSTRAINT
    Conference_day_registration_Conference_day_reservations
    FOREIGN KEY (reservation_id)
    REFERENCES Conference_day_reservations (reservation_id)
    ON DELETE  CASCADE;

-- Reference: Conference_day_registration_Participants (table: Conference_day_registration)
ALTER TABLE Conference_day_registration ADD CONSTRAINT
    Conference_day_registration_Participants
    FOREIGN KEY (Participant_id)
    REFERENCES Participants (participant_id)
    ON DELETE  CASCADE;

-- Reference: Conference_day_reservations_Clients (table: Conference_day_reservations)
ALTER TABLE Conference_day_reservations ADD CONSTRAINT Conference_day_reservations_Clients
    FOREIGN KEY (clients_id)
    REFERENCES Clients (id);
```

```
1144
    -- Reference: Conference_day_reservations_Conference_days (table: Conference_day_reservations)
1146 ALTER TABLE Conference_day_reservations ADD CONSTRAINT
         Conference_day_reservations_Conference_days
         FOREIGN KEY (conference_day_id)
1148     REFERENCES Conference_days (conference_day_id)
         ON DELETE  CASCADE;
1150
    -- Reference: Conference_days_Conferences (table: Conference_days)
1152 ALTER TABLE Conference_days ADD CONSTRAINT Conference_days_Conferences
         FOREIGN KEY (conference_id)
1154     REFERENCES Conferences (Conference_id)
         ON DELETE  CASCADE;
1156
    -- Reference: Discounts_Conference_days (table: Early_signup_discounts)
1158 ALTER TABLE Early_signup_discounts ADD CONSTRAINT Discounts_Conference_days
         FOREIGN KEY (conference_day_id)
1160     REFERENCES Conference_days (conference_day_id)
         ON DELETE  CASCADE;
1162
    -- Reference: Participants_Clients (table: Participants)
1164 ALTER TABLE Participants ADD CONSTRAINT Participants_Clients
         FOREIGN KEY (clients_id)
1166     REFERENCES Clients (id)
         ON DELETE  SET NULL;
1168
    -- Reference: Payments_Conference_day_reservations (table: Payments)
1170 ALTER TABLE Payments ADD CONSTRAINT Payments_Conference_day_reservations
         FOREIGN KEY (reservation_id)
1172     REFERENCES Conference_day_reservations (reservation_id);

1174 -- Reference: Workshop_registration_Participants (table: Workshop_registration)
    ALTER TABLE Workshop_registration ADD CONSTRAINT Workshop_registration_Participants
1176     FOREIGN KEY (Participant_id)
         REFERENCES Participants (participant_id)
1178     ON DELETE  CASCADE;

1180 -- Reference: Workshop_registration_Workshop_reservations (table: Workshop_registration)
    ALTER TABLE Workshop_registration ADD CONSTRAINT Workshop_registration_Workshop_reservations
1182     FOREIGN KEY (reservation_id)
         REFERENCES Workshop_reservations (reservation_id)
1184     ON DELETE  CASCADE;

1186 -- Reference: Workshop_reservations_Conference_day_reservations (table: Workshop_reservations)
    ALTER TABLE Workshop_reservations ADD CONSTRAINT
         Workshop_reservations_Conference_day_reservations
1188     FOREIGN KEY (Conference_day_res_id)
         REFERENCES Conference_day_reservations (reservation_id)
1190     ON DELETE  CASCADE;

1192 -- Reference: Workshop_reservations_Workshops (table: Workshop_reservations)
    ALTER TABLE Workshop_reservations ADD CONSTRAINT Workshop_reservations_Workshops
1194     FOREIGN KEY (workshop_id)
         REFERENCES Workshops (workshop_id)
1196     ON DELETE  CASCADE;

1198 -- Reference: Workshops_Conference_days (table: Workshops)
    ALTER TABLE Workshops ADD CONSTRAINT Workshops_Conference_days
1200     FOREIGN KEY (conference_day_id)
         REFERENCES Conference_days (conference_day_id);
1202
    -- End of file.
```

../Create.sql

```
CREATE FUNCTION IsValidNip(
    @nip nvarchar(15)
)
    RETURNS bit
AS
BEGIN
    IF ISNUMERIC(@nip) = 0
        BEGIN
            RETURN 0
        END

    IF @nip = '0000000000'
        BEGIN
            RETURN 0
        END
    IF @nip = '1234567891'
        BEGIN
            RETURN 0
        END
    IF @nip = '1111111111'
        BEGIN
            RETURN 0
        END
    IF @nip = '1111111112'
        BEGIN
            RETURN 0
        END
    IF @nip = '9999999999'
        BEGIN
            RETURN 0
        END
    IF @nip = '1111111112'
        BEGIN
            RETURN 0
        END

    DECLARE @sum INT;
    SET @sum = 6 * CONVERT(INT, SUBSTRING(@nip, 1, 1)) +
               5 * CONVERT(INT, SUBSTRING(@nip, 2, 1)) +
               7 * CONVERT(INT, SUBSTRING(@nip, 3, 1)) +
               2 * CONVERT(INT, SUBSTRING(@nip, 4, 1)) +
               3 * CONVERT(INT, SUBSTRING(@nip, 5, 1)) +
               4 * CONVERT(INT, SUBSTRING(@nip, 6, 1)) +
               5 * CONVERT(INT, SUBSTRING(@nip, 7, 1)) +
               6 * CONVERT(INT, SUBSTRING(@nip, 8, 1)) +
               7 * CONVERT(INT, SUBSTRING(@nip, 9, 1));

    IF CONVERT(TINYINT, SUBSTRING(@nip, 10, 1)) = (@sum % 11)
        BEGIN
            RETURN 1
        END
    RETURN 0
END
```

../Functions/isValidNip.sql

Generator danych

```
from ParticipantsGenerator import *
from ClientsGenerator import *
from ConfDayResGenerator import *
from ConfDaysGenerator import *
from EsdGenerator import *
from ConferenceGenerator import *
from WorkshopGenerator import *
from WorkshopResGen import *
from ConfRegistrationGen import *
from WorkshopRegistrationGen import *
from PaymentGen import *

from random import Random
from faker import Faker

faker = Faker(['pl_PL'])
rand = Random()

generators = []

part_gen = ParticipantsGenerator(rand)
part_gen.make(None, 4000)
# part_gen.make(None, 5)

clients_gen = ClientsGenerator(part_gen)
generators.append(clients_gen)
clients_gen.make(400)
# clients_gen.make(10)

conf_gen = ConferenceGenerator(rand, faker)
generators.append(conf_gen)
conf_gen.make(72)
# conf_gen.make(3)

day_gen = ConfDaysGenerator(rand, faker)
generators.append(day_gen)
day_gen.make(conf_gen.conferences)

esd_gen = EsdGenerator(rand, faker)
generators.append(esd_gen)
esd_gen.make(day_gen.days)

day_res_gen = ConfDayResGenerator(clients_gen, rand, faker)
generators.append(day_res_gen)
day_res_gen.make(day_gen.days)

wor_gen = WorkshopGenerator(part_gen, rand, faker)
generators.append(wor_gen)
wor_gen.make(day_gen.days)

wor_res_gen = WorkshopResGen(day_res_gen)
generators.append(wor_res_gen)
wor_res_gen.make(wor_gen.workshops)

conf_reg_gen = ConfRegistrationGen(rand, part_gen)
generators.append(conf_reg_gen)
conf_reg_gen.make(day_res_gen.reservations)

wor_reg_gen = WorkshopRegistrationGen(rand, part_gen)
generators.append(wor_reg_gen)
wor_reg_gen.make(wor_res_gen.reservations)

pay_gen = PaymentGen(rand, faker)
generators.append(pay_gen)
pay_gen.make(day_res_gen.reservations)

for g in generators:
    print(g.to_sql())
```

../Generator/main.py

```
from datetime import datetime, timedelta, time
```

```
class AbstractClass:
    def random_time(self, date):
        return datetime.combine(date, time(self.rand.randint(0, 23), self.rand.randint(0, 59))
    )
```

../Generator/AbstractClass.py

```
def table_to_sql(table, n_row=True):
    res = '\n' if n_row else ''
    res += table[0].to_sql()
    lid = 0
    for v in range(1, len(table)):
        if v - lid < 999:
            res += ','
            res += table[v].to_sql(False)
        else:
            res += '\n'
            res += table[v].to_sql()
            lid = v
    return res
```

../Generator/AbstractGenerator.py

```
class Client:
    def __init__(self, cl_id, faker):
        self.cl_id = cl_id
        self.faker = faker

        add = self.faker.address().split('\n')
        self.address = add[0]
        self.zip_code = add[-1].split(' ')[0]
        self.city = ' '.join(add[-1].split(' ')[1:])

    def to_sql(self, start=True):
        values = "(" + str(self.cl_id) + ",\'" + self.zip_code + "\',\'" + self.city + "\',\'"
     + self.address + "\')"
        return "INSERT INTO CLIENTS (id,zip_code, city, address) " \
               "VALUES "+ values if start else values
```

../Generator/Client.py

```
from faker import Faker
import random
from ParticipantsGenerator import *
from Client import *
from Company import *


def table_to_sql(table):
    res = '\n'
    res += table[0].to_sql()
    lid = 0
    for v in range(1, len(table)):
        if v - lid < 999:
            res += ','
            res += table[v].to_sql(False)
        else:
            lid = v
            res += '\n'
            res += table[v].to_sql()
    return res


class ClientsGenerator:
    def __init__(self, participants_gen, next_client_id=1):
        self.faker = Faker(['pl_PL'])
        self.rand = random.Random()

        self.next_client_id = next_client_id
        self.participants_gen = participants_gen
        self.clients = []
        self.companies = []

    def choice(self):
```

```
        return self.rand.choice(self.clients)

    def clients_count(self):
        return len(self.clients)

    def make(self, n=1):
        for _ in range(n):
            cl = Client(self.next_client_id, self.faker)
            if self.rand.randint(0, 1) == 0:
                cm = Company(self.next_client_id, self.faker, self.rand)
                self.companies.append(cm)
            else:
                self.participants_gen.make(self.next_client_id)
            self.next_client_id += 1
            self.clients.append(cl)

    def to_sql(self):
        res = 'SET IDENTITY_INSERT Clients ON'
        res += table_to_sql(self.clients)

        res += '\nSET IDENTITY_INSERT Clients OFF'
        res += table_to_sql(self.companies)
        res += '\n'
        res += self.participants_gen.to_sql()

        self.clients = []
        self.companies = []
        return res
```

../Generator/ClientsGenerator.py

```
class Company:
    def __init__(self, clients_id ,faker, rand):
        self.faker = faker
        self.rand = rand
        self.clients_id = clients_id

        self.name = self.faker.company()
        self.phone = self.faker.phone_number()
        self.email = self.faker.email()
        self.nip = self.random_nip()

    def random_nip(self):
        res = ''
        sum = 0
        weights = [6, 5, 7, 2, 3, 4, 5, 6, 7]
        for i in range(8):
            k = self.rand.randint(1 if i < 3 else 0, 9)
            sum += weights[i] * k
            res += str(k)
        k = self.rand.randint(0, 9)
        if (sum + (k * weights[8])) % 11 == 10:
            k += (1 if k + 1 < 10 else -1)
        res += str(k)
        sum += k * weights[8]
        res += str(sum % 11)
        return res

    def to_sql(self, start=True):
        values = "(\'" + self.name + "\',\'" + self.nip + "\',\'" + self.phone + "\'," + str(
            self.clients_id) + ",\'" + self.email + "\')"
        return "INSERT INTO COMPANIES (companyName, nip, phone, clients_id, email) VALUES "+
    values if start else values
```

../Generator/Company.py

```
from AbstractClass import *


class ConfDayReservation(AbstractClass):
    def __init__(self, res_id, clients_id, day, part_count, faker, rand):
        self.faker = faker
        self.rand = rand

        self.res_id = res_id
        self.day_id = day.day_id
```

```
        self.clients_id = clients_id

        self.date = self.random_time(self.faker.date_between(start_date=datetime.today(), end_date=day.date))
        self.active = self.rand.randint(0, 1)
        self.due_price = self.date + timedelta(weeks=self.rand.randint(1, 4))
        self.adult_seats = self.rand.randint(1, min(day.free_seats, part_count))
        self.student_seats = self.rand.randint(0, max(0, min(day.free_seats, part_count) - self.adult_seats))

        self.workshops_price = 0
        self.day_price = day.price * self.adult_seats
        self.day_price += day.price * (1 - day.stud_disc) * self.student_seats
        esds = list(filter(lambda x: x.date > self.date.date(), day.esds))
        esds = sorted(esds, key=lambda x: x.date)
        esd = 0 if len(esds) == 0 else esds[-1].discount
        self.day_price *= (1 - esd)

    def to_sql(self, start=True):
        values = "(" + str(
            self.res_id) + "," + str(self.day_id) + "," + str(self.clients_id) + ",\'" + str(self.date) + "\',\'" + str(
            self.active) + "\',\'" + str(self.due_price) + "\'," + str(self.adult_seats) + "," + str(
            self.student_seats) + ")"
        return "INSERT INTO Conference_day_reservations (reservation_id, conference_day_id, clients_id, reservation_date, active, due_price, adult_seats, student_seats) VALUES " + values if start else values
```

../Generator/ConfDayReservation.py

```
from random import Random
from faker import Faker
from ConfDayReservation import *


class ConfDayResGenerator:
    def __init__(self, clients_gen, rand=Random(), faker=Faker(['pl_PL']), next_res_id=1):
        self.faker = faker
        self.rand = rand

        self.clients_gen = clients_gen
        self.next_res_id = next_res_id
        self.reservations = []

    def choice(self):
        return self.rand.choice(self.reservations)

    def res_count(self):
        return len(self.reservations)

    def to_sql(self):
        res = 'SET IDENTITY_INSERT Conference_day_reservations ON'
        res += '\n'
        res += self.reservations[0].to_sql()
        for v in range(1, len(self.reservations)):
            res += ','
            res += self.reservations[v].to_sql(False)
        res += '\nSET IDENTITY_INSERT Conference_day_reservations OFF'
        self.reservations = []
        return res

    def make(self, days):
        for day in days:
            n_res = self.rand.randint(2, self.clients_gen.clients_count() / 5)
            while day.free_seats > 0 and n_res > 0:
                n_res -= 1
                self.reservations.append(
                    ConfDayReservation(self.next_res_id, self.clients_gen.choice().cl_id, day,
                                       len(self.clients_gen.participants_gen.participants),
                self.faker, self.rand))
                self.next_res_id += 1
                day.free_seats -= self.reservations[-1].adult_seats
                day.free_seats -= self.reservations[-1].student_seats
```

../Generator/ConfDayResGenerator.py

```python
from faker import Faker
from random import Random
from ConferenceDay import *
from datetime import datetime, timedelta, time
from AbstractGenerator import table_to_sql


class ConfDaysGenerator:
    def __init__(self, rand=Random(), faker=Faker(['pl_PL']), next_day_id=1):
        self.faker = faker
        self.rand = rand

        self.next_day_id = next_day_id
        self.days = []

    def make(self, conferences):
        for c in conferences:
            self.days.append(ConferenceDay(self.next_day_id, c.conf_id, self.faker, self.rand)
    )
            self.next_day_id += 1
            date = self.days[-1].date
            n = self.rand.randint(2, 4)
            for _ in range(n):
                date += timedelta(days=1)
                self.days.append(ConferenceDay(self.next_day_id, c.conf_id, self.faker, self.
    rand, date))
                self.next_day_id += 1

    def to_sql(self):
        res = 'SET IDENTITY_INSERT Conference_days ON'
        res += table_to_sql(self.days)
        res += '\nSET IDENTITY_INSERT Conference_days OFF'
        self.days = []
        return res
```

../Generator/ConfDaysGenerator.py

```python
class ConferenceDay:
    def __init__(self, day_id, conf_id, faker, rand, day=None):
        self.faker = faker
        self.rand = rand

        self.day_id = day_id
        self.conf_id = conf_id
        self.price = round(self.rand.uniform(10.0, 1000.0), 2)
        self.stud_disc = round(self.rand.uniform(0.0, 0.5), 2)
        self.date = (self.faker.date_between(start_date='today', end_date='+5y')) if day is
    None else day
        self.numb_of_seats = self.rand.randint(150, 250)
        self.free_seats = self.numb_of_seats
        self.esds = []

    def to_sql(self, start=True):
        values = "(" + str(self.day_id) + "," + str(self.conf_id) + ",\'" + str(self.date) + "
    \'," + str(
            self.price) + "," + str(self.stud_disc) + ',' + str(self.numb_of_seats) + ")"
        return "INSERT INTO Conference_days (conference_day_id, conference_id, date,
    standard_price, student_discount, number_of_seats) VALUES " + values if start else values
```

../Generator/ConferenceDay.py

```python
from faker import Faker
from random import Random
from Conference import *
from AbstractGenerator import table_to_sql


class ConferenceGenerator:
    def __init__(self, rand=Random(), faker=Faker(['pl_PL']), next_conference_id=1):
        self.faker = faker
        self.rand = rand

        self.next_conference_id = next_conference_id
        self.conferences = []
```

```
    def to_sql(self):
        res = 'SET IDENTITY_INSERT Conferences ON'
        res += table_to_sql(self.conferences)
        res += '\nSET IDENTITY_INSERT Conferences OFF'
        self.conferences = []
        return res

    def make(self, n=1):
        for _ in range(n):
            self.conferences.append(Conference(self.next_conference_id, self.faker))
            self.next_conference_id += 1
```

../Generator/ConferenceGenerator.py

```
class Conference:
    def __init__(self, conf_id, faker):
        self.faker = faker

        self.conf_id = conf_id
        self.name = self.faker.bs()
        self.description = self.faker.text()

    def to_sql(self, start=True):
        values = "(" + str(self.conf_id) + ",\'" + self.name + '\',\'' + self.description + "\')"
        return "INSERT INTO Conferences (Conference_id, name, description) VALUES " + values if start else values
```

../Generator/Conference.py

```
from ConfRegistration import *
from AbstractGenerator import *


class ConfRegistrationGen:
    def __init__(self, rand, part_gen):
        self.rand = rand

        self.part_gen = part_gen
        self.registrations = []

    def to_sql(self):
        res = table_to_sql(self.registrations, False)
        self.registrations = []
        return res

    def make(self, reservations):
        for res in reservations:
            parts = set([p.part_id for p in self.part_gen.participants])

            for _ in range(res.adult_seats):
                p = self.rand.sample(parts, 1)
                self.registrations.append(ConfRegistration(res.res_id, p[0], 1, self.rand))
                parts.remove(p[0])

            for _ in range(res.student_seats):
                p = self.rand.sample(parts, 1)
                self.registrations.append(ConfRegistration(res.res_id, p[0], 0, self.rand))
                parts.remove(p[0])
```

../Generator/ConfRegistrationGen.py

```
class ConfRegistration:
    def __init__(self, res_id, participant_id, student, rand):
        self.rand = rand

        self.res_id = res_id
        self.student = student
        self.participant_id = participant_id

    def to_sql(self, start=True):
        values = "(" + str(self.res_id) + "," + str(self.participant_id) + "," + str(self.student) + ")"
        return "INSERT INTO Conference_day_registration (reservation_id, Participant_id, is_student) VALUES " + values if start else values
```

../Generator/ConfRegistration.py

```
from random import Random
from faker import Faker
from Esd import *
from AbstractGenerator import *


class EsdGenerator:
    def __init__(self, rand=Random(), faker=Faker(['pl_PL'])):
        self.faker = faker
        self.rand = rand

        self.esds = []

    def to_sql(self):
        res = table_to_sql(self.esds, False)
        self.esds = []
        return res

    def make(self, days):
        for day in days:
            for _ in range(self.rand.randint(1, 5)):
                e = Esd(self.faker, self.rand, day)
                self.esds.append(e)
                day.esds.append(e)
```

../Generator/EsdGenerator.py

```
class Esd:
    def __init__(self, faker, rand, day):
        self.faker = faker
        self.rand = rand

        self.day_id = day.day_id
        self.discount = round(self.rand.uniform(0.0, 0.5), 2)
        self.date = self.faker.date_between(start_date='-2y', end_date=day.date)

    def to_sql(self, start=True):
        values = "(" + str(self.day_id) + ",\'" + str(self.date) + "\'," + str(self.discount)
    + ")"
        return "INSERT INTO Early_signup_discounts (conference_day_id, end_date, discount)
    VALUES " + values if start else values
```

../Generator/Esd.py

```
class Participant:
    def __init__(self, part_id, faker, client_id=None):
        self.part_id = part_id
        self.client_id = client_id
        self.faker = faker

        n = self.faker.name().split(' ')
        self.name = n[0]
        self.surname = ' '.join(n[1:])
        self.phone = self.faker.phone_number()
        self.email = self.faker.email()

    def to_sql(self, start=True):
        values = "(" + str(self.part_id) + "," + (str(
            self.client_id) if self.client_id is not None else 'null') + ",\'" + self.name + "
    \',\'" + self.surname + "\',\'" + self.email + "\',\'" + self.phone + "\')"
        return "INSERT INTO Participants (participant_id,clients_id, name, surname, email,
    phone) VALUES " + values if start else values
```

../Generator/Participant.py

```
from faker import Faker
from Participant import *
from AbstractGenerator import table_to_sql


class ParticipantsGenerator:
```

```
    def __init__(self, rand, faker=Faker(['pl_PL']), next_participant_id=1):
        self.faker = faker
        self.rand = rand

        self.next_participant_id = next_participant_id
        self.participants = []

    def choice(self):
        return self.rand.choice(self.participants)

    def part_count(self):
        return len(self.participants)

    def make(self, clients_id=None, n=1):
        for _ in range(n):
            res = Participant(self.next_participant_id, self.faker, clients_id)
            self.next_participant_id += 1
            self.participants.append(res)

    def to_sql(self):
        res = 'SET IDENTITY_INSERT Participants ON'
        res += table_to_sql(self.participants)
        res += '\nSET IDENTITY_INSERT Participants OFF'
        self.participants = []
        return res
```

../Generator/ParticipantsGenerator.py

```
from Payment import *
from AbstractGenerator import *


class PaymentGen:
    def __init__(self, rand, faker):
        self.rand = rand
        self.faker = faker
        self.payments = []

    def make(self, reservations):
        for res in reservations:
            price = res.workshops_price + res.day_price
            payed = int(price)
            if self.rand.randint(1, 1000) % 5 == 0:  # nie oplacone
                payed = 0 if self.rand.randint(0, 1000) % 2 == 0 else self.rand.randint(0, int(price))
            if payed == 0:
                continue
            n_payments = self.rand.randint(1, 4)
            values = [payed // n_payments for _ in range(n_payments)]
            i = 0
            while sum(values) < payed:
                values[i % n_payments] += 1
                i += 1
            i = 0
            while i < n_payments:
                p = self.rand.randint(0, values[i])
                values[i] -= p
                values[(i + 1) % n_payments] += p
                i += 1
            values[-1] += round(price - int(price), 2)
            for value in values:
                self.payments.append(Payment(value, res, self.faker, self.rand))

    def to_sql(self):
        res = table_to_sql(self.payments, False)
        self.payments = []
        return res
```

../Generator/PaymentGen.py

```
from AbstractClass import *


class Payment(AbstractClass):
    def __init__(self, value, day_res, faker, rand):
        self.faker = faker
```

```
        self.rand = rand

        self.res_id = day_res.res_id
        self.date = self.random_time(self.faker.date_between(start_date=day_res.date, end_date=day_res.due_price))
        self.value = value

    def to_sql(self, start=True):
        values = "(" + str(self.res_id) + ",\'" + str(self.date) + "\'," + str(self.value) + ")"
        return "INSERT INTO Payments (reservation_id, in_date, value) VALUES " + values if start else values
```

../Generator/Payment.py

```
from random import Random
from faker import Faker
from Workshop import *
from AbstractGenerator import table_to_sql


class WorkshopGenerator:
    def __init__(self, part_gen, rand=Random(), faker=Faker(['pl_PL']), next_workshop_id=1):
        self.faker = faker
        self.rand = rand

        self.part_gen = part_gen
        self.workshops = []
        self.next_workshop_id = next_workshop_id

    def to_sql(self):
        res = 'SET IDENTITY_INSERT Workshops ON'
        res += table_to_sql(self.workshops)
        res += '\nSET IDENTITY_INSERT Workshops OFF'
        self.workshops = []
        return res

    def make(self, days):
        for day in days:
            for _ in range(self.rand.randint(2, 6)):
                self.workshops.append(
                    Workshop(self.next_workshop_id, day, self.part_gen.part_count(), self.faker, self.rand))
                self.next_workshop_id += 1
```

../Generator/WorkshopGenerator.py

```
from datetime import datetime, timedelta, time


class Workshop:
    def __init__(self, workshop_id, day, part_count, faker, rand):
        self.faker = faker
        self.rand = rand

        self.workshop_id = workshop_id
        self.day_id = day.day_id
        self.price = round(self.rand.uniform(10.0, 1000.0), 2)
        self.start = self.random_time(day.date)
        self.end = self.random_time(day.date)
        if self.start > self.end:
            self.start, self.end = self.end, self.start
        self.topic = self.faker.bs()
        self.numb_seats = self.rand.randint(1, min(day.numb_of_seats, part_count))
        self.free_seats = self.numb_seats

    def random_time(self, date):
        return datetime.combine(date, time(self.rand.randint(0, 23), self.rand.randint(0, 59)))

    def to_sql(self, start=True):
        values = "(" + str(self.workshop_id) + "," + str(self.day_id) + ",\'" + str(self.start) + "\',\'" + str(
            self.end) + "\',\'" + self.topic + "\'," + str(self.price) + "," + str(self.numb_seats) + ")"
```

```
        return "INSERT INTO Workshops (workshop_id, conference_day_id, start_time, end_time,
    topic, price, number_of_seats) VALUES " + values if start else values
```

../Generator/Workshop.py

```
from WorkshopRegistration import *
from AbstractGenerator import *


class WorkshopRegistrationGen:
    def __init__(self, rand, part_gen):
        self.rand = rand

        self.part_gen = part_gen
        self.registrations = []

    def to_sql(self):
        res = table_to_sql(self.registrations, False)
        self.registrations = []
        return res

    def make(self, reservations):
        for res in reservations:
            parts = set([p.part_id for p in self.part_gen.participants])
            for _ in range(res.nr_seats):
                p = self.rand.sample(parts, 1)
                self.registrations.append(WorkshopRegistration(res.res_id, p[0]))
                parts.remove(p[0])
```

../Generator/WorkshopRegistrationGen.py

```
class WorkshopRegistration:
    def __init__(self, res_id, participant_id):
        self.res_id = res_id
        self.participant_id = participant_id

    def to_sql(self, start=True):
        values = "(" + str(self.res_id) + "," + str(self.participant_id) + ")"
        return "INSERT INTO Workshop_registration (reservation_id, Participant_id) VALUES " +
    values if start else values
```

../Generator/WorkshopRegistration.py

```
from random import Random
from faker import Faker
from WorkshopRes import *
from AbstractGenerator import table_to_sql


class WorkshopResGen:
    def __init__(self, conf_day_res_gen, rand=Random(), faker=Faker(['pl_PL']), next_res_id=1)
    :
        self.faker = faker
        self.rand = rand

        self.conf_day_res_gen = conf_day_res_gen
        self.next_res_id = next_res_id
        self.reservations = []

    def to_sql(self):
        res = 'SET IDENTITY_INSERT Workshop_reservations ON'
        res += table_to_sql(self.reservations)
        res += '\nSET IDENTITY_INSERT Workshop_reservations OFF'
        self.reservations = []
        return res

    def make(self, workshops):
        for work in workshops:
            n_res = self.rand.randint(1, self.conf_day_res_gen.res_count() // 4)
            while work.free_seats > 0 and n_res > 0:
                n_res -= 1
                self.reservations.append(
                    WorkshopRes(self.next_res_id, work, self.conf_day_res_gen.choice(), self.
    faker, self.rand))
                self.next_res_id += 1
```

```
1030                work.free_seats -= self.reservations[-1].nr_seats
```

../Generator/WorkshopResGen.py

```
1000 from AbstractClass import *
1002
     class WorkshopRes(AbstractClass):
1004     def __init__(self, res_id, workshop, conf_res, faker, rand):
             self.faker = faker
1006         self.rand = rand

1008         self.res_id = res_id
             self.work_id = workshop.workshop_id
1010         self.date = self.random_time(
                 self.faker.date_between(start_date=datetime.today(), end_date=workshop.start.date
     ()))
1012         self.due_price = self.date + timedelta(weeks=self.rand.randint(1, 4))
             self.nr_seats = self.rand.randint(1, workshop.free_seats)
1014         self.conf_res_id = conf_res.res_id
             self.active = self.rand.randint(0, 1)
1016
             conf_res.workshops_price += self.nr_seats * workshop.price
1018
         def to_sql(self, start=True):
1020         values = "(" + str(
                 self.res_id) + "," + str(self.work_id) + ",\'" + str(self.date) + "\',\'" + str(
1022             self.due_price) + "\'," + str(
                 self.nr_seats) + ',' + str(self.conf_res_id) + ',' + str(self.active) + ")"
1024         return "INSERT INTO Workshop_reservations (reservation_id, workshop_id,
     reservation_date, due_price, nr_of_seats, Conference_day_res_id, active) VALUES " + values
      if start else values
```

../Generator/WorkshopRes.py