**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ**

**ФГБОУ ВО "Пензенский государственный университет"**

Кафедра "Математическое обеспечение и применение ЭВМ"

Отчет

по лабораторной работе №2

Выполнили:

Хамзин Х.

Суворов М.

Хусаинов И.

ст. гр. 17ВИ1

Проверила:

Попова Н. А.

2020

**Цель работы.** Разработка функций и хранимых процедур для автоматического заполнения оперативной базы данных тестовыми данными

**Код заполнения на языке Python**

import psycopg2

import random

from Utils.DataRandomMaker import random\_date

class Client(object):

def \_\_init\_\_(self, connection, constantRepo):

"""Constructor"""

self.connection = connection

self.cursor = connection.cursor()

self.repository = constantRepo

pass

def fillClients(self, countRows):

for i in range(countRows):

addres = f'{random.choice(self.repository.city)}, {random.choice(self.repository.street)} {random.randint(1, 100)}, кв.{random.randint(1, 500)}'

query = f'INSERT INTO client(adress, Phone\_number,Surname, name,lastname, Birth\_date, Passport, itn, id\_ur\_person)' \

f' VALUES(' \

f'\'{addres}\'' \

f',\'{str(random.randint(1000000000, 9999999999))}\'' \

f',\'{random.choice(self.repository.second\_name\_man)}\'' \

f',\'{random.choice(self.repository.first\_name\_man)}\'' \

f',\'{random.choice(self.repository.last\_name\_man)}\'' \

f',\'{random\_date("1.1.1950", "1.1.2005", random.random())}\'' \

f',\'{str(random.randint(1111111111, 9999999999))}\'' \

f',\'{str(random.randint(1000000000, 9999999999))}\'' \

f',\'{"1"}\')'

self.cursor.execute(query)

self.connection.commit()

def getCountClients(self):

self.cursor.execute("SELECT COUNT(\*) FROM client;")

rows = self.cursor.fetchall()

self.connection.commit()

return rows[0][0]

def updateClientAccessory(self, id\_ur, id\_client):

query = f'UPDATE client set id\_ur\_person = \'{id\_ur}\'' \

f' WHERE id\_client =\'{id\_client}\''

self.cursor.execute(query)

self.connection.commit()

import psycopg2

import random

from Utils.DataRandomMaker import random\_date

class Room(object):

def \_\_init\_\_(self, connection, constantRepo):

"""Constructor"""

self.connection = connection

self.cursor = connection.cursor()

self.repository = constantRepo

pass

def fillRoomsAndCategory(self, countRooms):

self.fillCategoryRoom()

self.fillRoom(countRooms)

def fillRoom(self, countRows):

countBuilding = self.getCountBuilding()

countCategoryRoom = self.getCountCategoryRoom()

counter = 1

for i in range(0, countRows):

self.cursor.execute('INSERT INTO Room(id\_category\_room, floor\_lvl,room\_number, room\_cost, id\_Build ) values (%s,%s,%s,%s,%s)',

(

random.randint(1, countCategoryRoom),

random.randint(1, 4),

counter,

random.randint(1600, 10000),

random.randint(1, countBuilding),

))

counter += 1

self.connection.commit()

def fillCategoryRoom(self):

for i in range(0, len(self.repository.category\_room\_name)):

self.cursor.execute('INSERT INTO Category\_room(name,beds\_count) values (%s,%s)', (random.choice(self.repository.category\_room\_name), random.randint(1,4)))

self.connection.commit()

def getCountRoom(self):

self.cursor.execute("SELECT COUNT(\*) FROM Room;")

rows = self.cursor.fetchall()

self.connection.commit()

return rows[0][0]

def getCountCategoryRoom(self):

self.cursor.execute("SELECT COUNT(\*) FROM Category\_room;")

rows = self.cursor.fetchall()

self.connection.commit()

return rows[0][0]

#эта функйия должан быть не тут

def getCountBuilding(self):

self.cursor.execute("SELECT COUNT(\*) FROM Build;")

rows = self.cursor.fetchall()

self.connection.commit()

return rows[0][0]

import psycopg2

import random

from Utils.DataRandomMaker import random\_date

class Room(object):

def \_\_init\_\_(self, connection, constantRepo):

"""Constructor"""

self.connection = connection

self.cursor = connection.cursor()

self.repository = constantRepo

pass

def fillRoomsAndCategory(self, countRooms):

self.fillCategoryRoom()

self.fillRoom(countRooms)

def fillRoom(self, countRows):

countBuilding = self.getCountBuilding()

countCategoryRoom = self.getCountCategoryRoom()

counter = 1

for i in range(0, countRows):

self.cursor.execute('INSERT INTO Room(id\_category\_room, floor\_lvl,room\_number, room\_cost, id\_Build ) values (%s,%s,%s,%s,%s)',

(

random.randint(1, countCategoryRoom),

random.randint(1, 4),

counter,

random.randint(1600, 10000),

random.randint(1, countBuilding),

))

counter += 1

self.connection.commit()

def fillCategoryRoom(self):

for i in range(0, len(self.repository.category\_room\_name)):

self.cursor.execute('INSERT INTO Category\_room(name,beds\_count) values (%s,%s)', (random.choice(self.repository.category\_room\_name), random.randint(1,4)))

self.connection.commit()

def getCountRoom(self):

self.cursor.execute("SELECT COUNT(\*) FROM Room;")

rows = self.cursor.fetchall()

self.connection.commit()

return rows[0][0]

def getCountCategoryRoom(self):

self.cursor.execute("SELECT COUNT(\*) FROM Category\_room;")

rows = self.cursor.fetchall()

self.connection.commit()

return rows[0][0]

#эта функйия должан быть не тут

def getCountBuilding(self):

self.cursor.execute("SELECT COUNT(\*) FROM Build;")

rows = self.cursor.fetchall()

self.connection.commit()

return rows[0][0]

import psycopg2

import random

from Utils.DataRandomMaker import random\_date

class Worker(object):

def \_\_init\_\_(self, connection, constantRepo):

"""Constructor"""

self.connection = connection

self.cursor = connection.cursor()

self.repository = constantRepo

pass

def fillWorker(self, countWorkers):

employment\_number = 1000

# заполение сотрудника

for i in range(countWorkers):

addres = f'{random.choice(self.repository.city)}, {random.choice(self.repository.street)} {random.randint(1, 100)}, кв.{random.randint(1, 500)}'

query = f'INSERT INTO Worker(employment\_number, surname, name, lastname, birth\_date, passport, address, itn)'\

f' VALUES({employment\_number},\'' \

f'{random.choice(self.repository.second\_name\_man)}\'' \

f',\'{random.choice(self.repository.first\_name\_man)}\',' \

f'\'{random.choice(self.repository.last\_name\_man)}\',' \

f'\'{random\_date("1.1.1950", "1.1.2005", random.random())}\'' \

f',\'{str(random.randint(1000000000, 9999999999))}\'' \

f',\'{addres}\'' \

f',\'{str(random.randint(100000000000, 999999999999))}\')'

employment\_number += 1

self.cursor.execute(query)

self.connection .commit()

def getCountWorker(self):

self.cursor.execute("SELECT COUNT(\*) FROM worker;")

rows = self.cursor.fetchall()

self.connection.commit()

return rows[0][0]

import random

from datetime import datetime, timedelta

from Model.Client import Client

from Model.Room import Room

from Model.Worker import Worker

class PurchaseService(object):

def \_\_init\_\_(self, connection, constantRepo):

"""Constructor"""

self.connection = connection

self.cursor = connection.cursor()

self.repository = constantRepo

self.client = Client(connection, constantRepo)

self.room = Room(connection, constantRepo)

self.woker = Worker(connection, constantRepo)

self.fillStatusContract()

pass

def fillStatusContract(self):

for name in self.repository.status\_contract:

self.cursor.execute(f'INSERT INTO Status\_contract(name) values (\'{name}\')')

self.connection.commit()

def fillBooking(self):

countClient = self.client.getCountClients()

countRooms= self.room.getCountRoom()

countService = self.getServiceCount()

notBooked = (list(range(1, countRooms)))

notUsedClient = (list(range(1, countClient)))

#TODO: Тут можно сделать более адаптивно и гибко

for i in range(1, countRooms - 1):

clintId = random.choice(notUsedClient)

roomId = random.choice(notBooked)

notBooked.remove(roomId)

diffBooking = random.randint(15, 35)

diffSeatl = random.randint(-14, 20)

diffDeparture = random.randint(21, 60)

bookingTime = datetime.now() - timedelta(days= diffBooking)

settlementTime = bookingTime + timedelta(days= diffSeatl)

departeTime = settlementTime + timedelta(days= diffDeparture)

self.cursor.execute(

'INSERT INTO Booking(settlement\_time, departure\_time, booking\_time, id\_room) values (%s,%s,%s,%s)', (settlementTime, departeTime, bookingTime, roomId))

self.connection.commit()

self.fillStatusBooking(i, settlementTime)

self.fillOccupiedClient(i, clintId)

self.fillContract(clintId, roomId, settlementTime, departeTime)

self.fillBiltPosition( roomId, i)

self.updateBookingContract(i)

self.updateBiltPosition(i, roomId)

def getCountBooking(self):

self.cursor.execute("SELECT COUNT(\*) FROM booking;")

rows = self.cursor.fetchall()

self.connection.commit()

return rows[0][0]

def fillStatusBooking(self, bookingId, settlementTime):

self.cursor.execute('INSERT INTO Status\_booking(id\_type\_booking, time\_operation) values (%s,%s)', (1, settlementTime))

self.connection.commit()

self.cursor.execute(

'UPDATE booking set id\_status = %s where id\_booking = %s ', (bookingId, bookingId))

self.connection.commit()

def fillOccupiedClient(self, idBooking, clintId):

self.cursor.execute('INSERT INTO Occupied\_client(id\_client, id\_booking) values (%s,%s)', (clintId , idBooking))

self.connection.commit()

pass

def fillContract(self, clintId, roomId, settlementTime, departeTime):

contractnumber = random.randint(11111111, 99999999)

roomCost = self.getRoomCostById(roomId)

countworker = self.woker.getCountWorker()

self.cursor.execute('INSERT INTO Contract(contract\_number, start\_date, end\_date, amount, id\_status\_contract, id\_client, id\_worker) '

'values (%s,%s,%s,%s,%s,%s,%s)',

(contractnumber, settlementTime, departeTime,roomCost + random.randint(150, 15564), 3, clintId, random.randint(1, countworker)))

self.connection.commit()

pass

def getRoomCostById(self, roomId):

self.cursor.execute("SELECT room\_cost FROM Room where id\_room = " + str(roomId))

cost = self.cursor.fetchall()

self.connection.commit()

return cost[0][0]

def fillBiltPosition(self, roomId, contractId):

serviceCount = self.getServiceCount()

self.cursor.execute('INSERT INTO Bilt\_position(id\_contract, id\_disscount, id\_room, id\_service, with\_discount) '

'values (%s,%s,%s,%s,%s)',

(contractId, 1, roomId, random.randint(1, serviceCount), self.getRoomCostById(roomId)))

pass

def getServiceCount(self):

self.cursor.execute("SELECT COUNT(\*) FROM Service;")

rows = self.cursor.fetchall()

self.connection.commit()

return rows[0][0]

def updateBookingContract(self, idBooking):

self.cursor.execute(

'UPDATE booking set id\_contract = %s where id\_booking = %s ', (idBooking, idBooking))

self.connection.commit()

pass

def updateBiltPosition(self, contractId, roomId):

serviceCount = self.getServiceCount()

self.cursor.execute('INSERT INTO Bilt\_position(id\_contract, id\_disscount, id\_service, with\_discount) '

'values (%s,%s,%s ,%s)',

(contractId, 1, random.randint(1, serviceCount), self.getRoomCostById(roomId)))

self.connection.commit()

pass

class ConstantRepository(object):

first\_name\_man = (

'Халим', 'Максим', 'Ильяс', 'Андрей', 'Святослав', 'Никита', 'Петр', 'Георгий', 'Руслан', 'Денис', 'Михаил',

'Рустам',

'Дмитрий', 'Егор', 'Кирилл', 'Валерий', 'Александр', 'Сергей', 'Иван', 'Виктор', 'Юрий', 'Игорь')

second\_name\_man = (

'Лысенко', 'Гаврилов', 'Данилов', 'Красовский', 'Дюжев', 'Макаров', 'Сидоров', 'Сенченко', 'Попов', 'Земнухов',

'Иванов', 'Петров', 'Орлушин', 'Чистяков', 'Русин', 'Хамзин', 'Суворов', 'Хусаинов', 'Пантелеев', 'Данилов',

'Назаров',

'Назаркин', 'Мельников', 'Макаров', 'Якомазов', 'Луба', 'Кулишов', 'Беляков')

last\_name\_man = (

'Халимович', 'Максимович', 'Ильясович', 'Андреевич', 'Святославович', 'Никитич', 'Георгиевич', 'Русланович',

'Денисович', 'Михайлович', 'Рустамович', 'Дмитриевич', 'Егорович', 'Кириллович', 'Валерьевич', 'Александрович',

'Сергеевич', 'Иванович', 'Викторович', 'Юрьевич', 'Игоревич')

count\_id = 1

street = (

'ул.Московская ', 'ул.Дмитриевская', 'пр.Победы ', 'пр.Строителей', 'ул.Ладожская', 'ул.Циолковского',

'ул.Водонаева ',

'ул.Октябрьская', 'ул.Ленина', 'ул.Сталинская', 'площадь Рефолюции', 'ул.Дачная', 'ул.Радужная', 'ул.Сиреневая',

'ул.Хорошая', 'ул.Виражная', 'ул.Гагарина')

city = (

'г.Москва', 'г.Пенза', 'г.Воронеж', 'г.Санкт-Петербург', 'г.Витебск', 'г.Казань', 'г.Сочи', 'г.Иваново',

'г.Мурманск',

'г.Волгоград', 'г.Ростов', 'г.Кузнецк', 'г.Сургут', 'г.Анадырь')

ur\_name = ['Не юр.лицо', 'Тинькофф', 'Microsoft', 'Apple', 'YouTube', 'Mailru', 'Газпром', 'Лукойл', 'Роснефть',

'ИнтерЭйр', 'Tortuga', 'Codeinside', 'Vigrom', 'OpenSolutions', 'Bitgames', 'Лейхтрум', 'Молескинес',

'Пож-центр', 'Intel', 'AMD', 'Foxconn', 'Qualcomm', 'Finmax', 'PickPoint', 'Ponyexpress', 'Почта России',

'Aliexpress']

category\_room\_name = [

"Семейная",

"Холостятска",

"Все включено",

"Мечта"

]

status\_contract = ["Расторгнут", "Истек", "Активен"]

def \_\_init\_\_(self):

pass

import time

def random\_date(start, end, prop):

return str\_time\_prop(start, end, '%d.%m.%Y', prop)

def str\_time\_prop(start, end, format, prop):

stime = time.mktime(time.strptime(start, format))

etime = time.mktime(time.strptime(end, format))

ptime = stime + prop \* (etime - stime)

return time.strftime(format, time.localtime(ptime))

import datetime

import random

import psycopg2

from Model import Client

from Model import UrPerson

from Model import Worker

from Model.Room import Room

from Purchase.Service.PurchaseService import PurchaseService

from Repository.ConstantRepository import \*

#

## Можно было придерадиваться синтексиса такого, но Халим не вывез погуглить

# cursor.execute("INSERT INTO a\_table (c1, c2, c3) VALUES(%s, %s, %s)", (v1, v2, v3))

conn = psycopg2.connect(dbname='postgres', user='root',

password='root', host='localhost')

cursor = conn.cursor()

constantRepository = ConstantRepository()

client = Client.Client(conn, constantRepository)

urPerson = UrPerson.UrPerson(conn, constantRepository)

worker = Worker.Worker(conn, constantRepository)

room = Room(conn, constantRepository)

purchaseService = PurchaseService(conn, constantRepository)

cursor.execute(

"INSERT INTO Disscount (Disscount\_name,Diss\_person, Diss\_legal) VALUES ('Постоянный клиент',12,12),('Заказ более 5 номеров за раз',10,15),('Заказ более 10 номеров за раз',15,20),('Заказ более 20 номеров за раз',25,30),('Скидка к случаю дня рождения',40,0),('Вернувшийся клиент(2 посещение)',10,15),('Скидки нет',0,0);")

conn.commit()

cursor.execute("INSERT INTO type\_booking(Name\_type) VALUES ('Оплачено'),('Забронированно'),('Отмена заказа');")

conn.commit()

cursor.execute(

"INSERT INTO Build(Build\_adress,Prestige) VALUES ('ул. Полярная 4',2),('ул. Полярная 3',3),('ул. Полярная 2',4),('ул. Полярная 1',5);")

conn.commit()

urPerson.fillUrPerson()

client.fillClients(2000)

worker.fillWorker(100)

countUrPerson = urPerson.getCountUrPerson()

countWorker = worker.getCountWorker()

count\_id = 1

#Заполняем приналдеожность клиентов к юр лицу

countClintFromUrPerson = 100

for i in range(1, 100):

id\_urperson = random.randint(1, countUrPerson)

client.updateClientAccessory(id\_urperson, i)

# Заполнение отзывов NORM TODO:Вынести функцию

cursor.execute("INSERT INTO FeedBack(id\_client,Mark,Descroption) VALUES "

"(1,4,'Неплохое обслуживание Юр лиц.'),"

"(2,4,'Обязательно вернемся в следующем году!'),"

"(3,4,'Высокое качество обслуживаения.'),"

"(4,3,'Некоторые наши сотрудники остались недовольны завтраком.'),"

"(5,3,'Один из наших сотрудников отравился.'),"

"(7,4,'Приятный персонал, много опций для номеров.'),"

"(8,4,'Все отлично, но цены для такой гостиницы должны быть чуть ниже.'),"

"(11,5,'Весь отдых прошел на высшем уровне!'),"

"(12,5,'Лучшая гостиница, которую я когда-либо посещал.'),"

"(14,2,'Отвратительно организован процесс бронирования, не смог вернуть полную стоимость после отмены заказа.'),"

"(15,3,'В целом неплохо, завтрак оставляет желать лучшего.'),"

"(16,4,'Обслуживание прошло на должном уровне.'),"

"(22,5,'Прекрасный отель. Буду советовать своим знакомым и друзьям!'),"

"(26,5,'Сервис был на высшем уровне. Порадовало количество и качество дополнительных услуг'),"

"(27,4,'Понравилось обслуживание и рабочий персонал'),"

"(34,4,'Хорошее расположение отеля, все нужные места в шаговой доступности'),"

"(37,1,'Ужасное обслуживание, в номере не убираются, ТВ не работает, поражает непрофессионализм сотрудников!'),"

"(42,2,'Отдых прошел неплохо, но некоторые сотрудники по хамски себя вели. Советую пересмотреть ваши кадры.'),"

"(44,2,'Научитесь готовить нормальную еду! Ужас просто!'),"

"(54,2,'Не приносили завтрак в номер, ТВ работает через раз, арендованный ноутбук слишком слабый.'),"

"(59,2,'Не работает кондиционер в номере отеля. '),"

"(62,3,'Для этих денег должны быть более лучшие условия'),"

"(66,3,'Редко проводятся уборки в отеле'),"

"(67,3,'Из всего меню понравились только грудка по-французски с макаронами'),"

"(69,4,'Благодаря высокой скорости Интернета смог посмотреть свой любимый матч без единой задержки.'),"

"(70,4,'Очень красивые уборщицы доставляли зрительное наслаждение каждое утро'),"

"(77,5,'Провел отличные выходные в этом отеле. Воспоминания со мной останутся надолго'),"

"(79,5,'Прекрасная сауна с бассейном, в которую включено барное обслуживание. Особенно понравился коктейль \"Гимнаст\". '),"

"(80,5,'Всё общение с персоналом было приятным. Хорошие номера и большой их выбор. '),(83,4,'Администратор подобрал подходящий для меня номер очень быстро и в соответствии с моим кошельком. '),"

"(45,4,'Поиграл в свою любимую фифу, благодаря playstation в моем номере и в целом провел время с удовольствием'),"

"(46,4,'Иногда прерывалось Wi-Fi соединение, но то, что мне нужно я успел найти'),"

"(47,2,'В номере остался минибар с открытыми и недопитами бутылками.'),"

"(48,3,'Хорошие салаты в меню, но напитки так себе'),"

"(49,4,'Есть хорошая возможность воспользоваться услугами массажиста. Было очень приятно!'),"

"(50,4,'Улыбчивый персонал и хороший администратор, который ответил на все вопросы'),"

"(51,3,'Мини-бар в номере оставляет желать лучшего. Нет ни русской водки, ни портвейна. '),"

"(52,3,'Холодильник в номере начал протекать, его починили оперативно, но было неприятно'),"

"(53,4,'Богатый шведский стол, практически всегда всё вкусно, за исключением отдельных блюд'),"

"(28,4,'Удобные матрацы в номере обеспечили мне приятный сон на протяжении всей недели'),"

"(29,5,'Всё было на высшем уровне от персонала до меню ресторана'),"

"(30,5,'Прекрасно организован вечерний досуг с музыкой и танцами, некогда скучать. '),"

"(31,5,'Этот отель останется в моей памяти на долгие годы. Очень хорошие воспоминания!'),"

"(32,5,'Обязательно вернемся в следующий раз.'),"

"(33,5,'Прекрасный отель!');")

conn.commit()

room.fillRoomsAndCategory(500)

# Заполнение услуг

cursor.execute(

"INSERT INTO service (service\_name, service\_cost) values ('Завтрак в номер',600),('Обед в номер',1500),('Ежедневная уборка',600),('Ужин в номер',700),('Минибар',3000),('Wi-fi',200),('Стирка личных вещей',300),('Глажка одежды',200),('Тренажерный зал',900),('Сейф',400),('Аренда ноутбука',1000),('Игровая приставка(PS4)',600),('Цифровое ТВ',300),('Сауна',1200),('Хамам',900),('Русская баня',900),('Бассейн',500),('Инструктор по фитнесу',1000),('Инструктор по плаванию',900),('Кружок детского творчества',400),('Аквапарк',1300),('Казино',2200),('Аренда беседки и гриля',2000),('Сеанс массажа',700),('Аренда переговорной комнаты',1500);")

conn.commit()

)

conn.commit()

purchaseService.fillBooking()

conn.close()

**Вывод:** Разработали функций и хранимых процедур для автоматического заполнения оперативной базы данных тестовыми данными