

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №2**

з дисципліни **Бази даних і засоби управління**

*на тему: “****Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL****”*

Виконав:

студент ІII курсу

групи КВ-01

Гнатенко Я.М.

Перевірив:

Павловський В. І.

Київ – 2022

*Метою роботи* є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

1. *Завдання* роботи полягає у наступному:
2. Реалізувати функції перегляду, внесення, редагування, та вилучення даних у таблицях бази даних, створеній у лабараторній роботі №1, засобами консольного інтерфейсу.
3. Передбачити автоматичне пакетне генерування “рандомізованих” даних у базі.
4. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
5. Програмний Код виконати згідно шаблону MVC

**Посилання на GIT: https://github.com/HYaroslawf/DB\_lab2**

**Логічна модель предметної галузі “Компанія”**

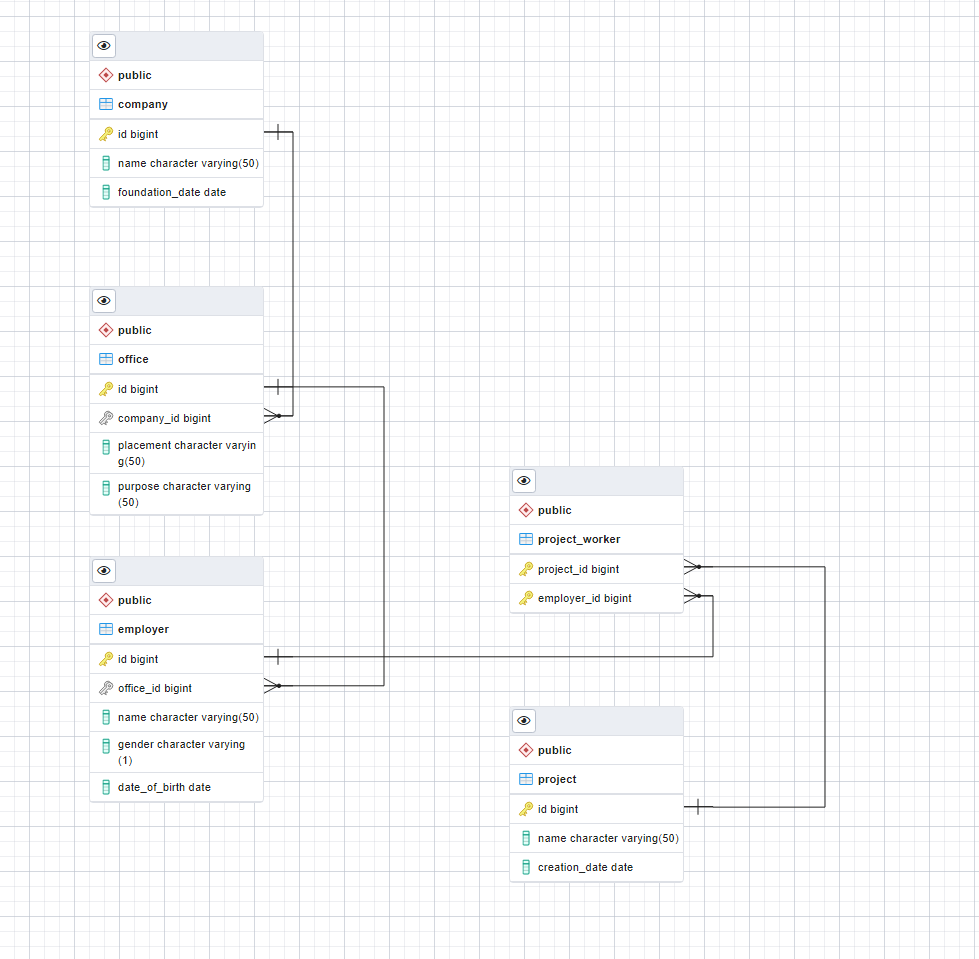


Рисунок 1. Логічна модель предметної галузі “Компанія”.

**Середовище та компоненти розробки:**

Для розробки використовувалась мова програмування Python, срередовище розробки Visual Studio Code, а також бібліотека, що надає API для доступу до PostgreSQL – psycopg2

**Шаблон проектування структура програми та її опис**

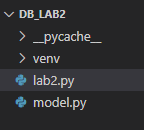
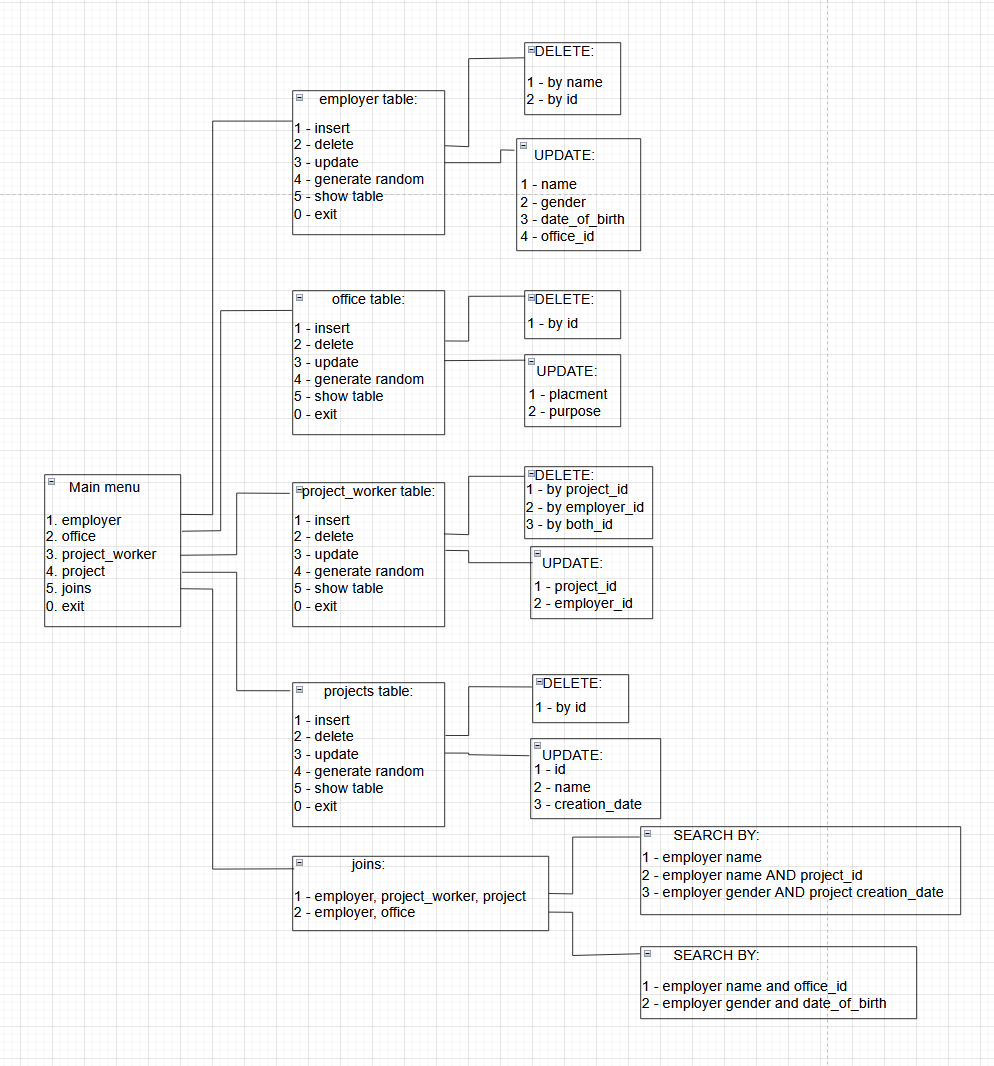


Рисунок 3. Структура програми

При проектуванні програми був використаний шаблон MVC. Сама програма складається з 2 файлів: **lab1.py** – реалізація view, **model.py** – реалізація model. **Controller** – реалізується як об’єднання **model** та **view**

**Меню користувача**

Рисунок 3 - Схема меню користувача

**Опис меню користувача:**

* в головному меню показано елементи з якими можна інтерактувати: 4 таблиці, пошук за допомогою “JOIN” і вихід “exit”
* кожна таблиця має власне підменю з параметрами: insert, delete, update, generate random, show table, exit. Підменю всіх таблиць ідентичне, різниця лише в параметрах, які потрібно заповнювати при виконанні операцій: insert, delete, update
* Підменю “join” надає 2 заготовлених варіантів JOIN-ів для пошуку. Після вибору заготовленого JOIN потрібно вибрати і заповнити відповідні параметри пошуку.

**Структура меню програми:**

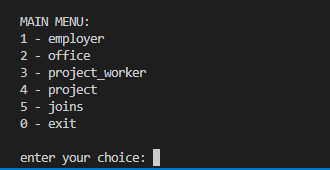


Рисунок 3. Головне меню

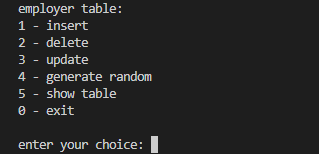


Рисунок 4. Підменю таблиці (підменю кожної таблиці ідентичне)

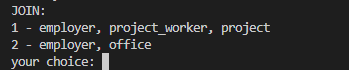
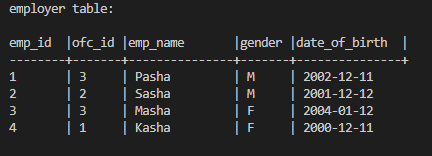


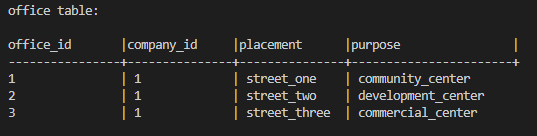
Рисунок 5. Підменю “JOIN”

**Внесення, редагування, вилучення та генерування рандомізованих даних таблиць(на прикладі таблиці employer та project\_worker)**

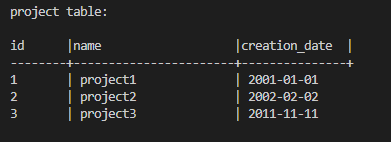
Результат операції Show(employer):



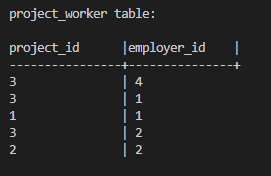
Результат операції Show(office):



Результат операції Show(project):

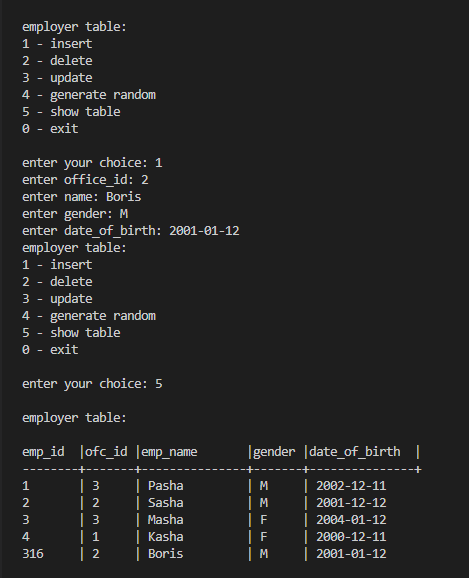


Результат операції Show(project\_worker):



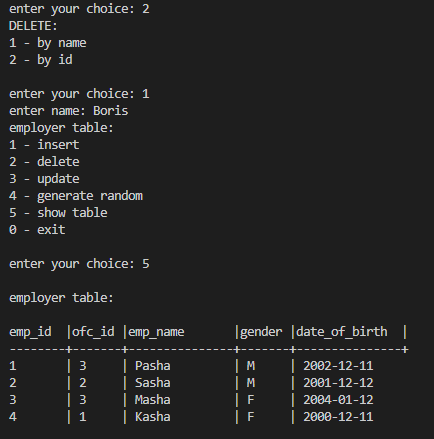
Виконання операції Insert(employer):

останній рядок



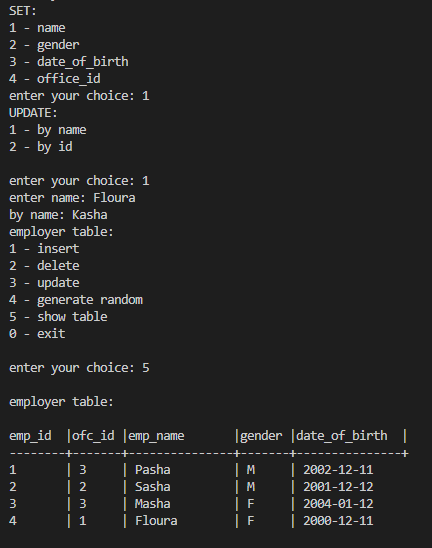
Виконання операції delete(employer):

останній рядок



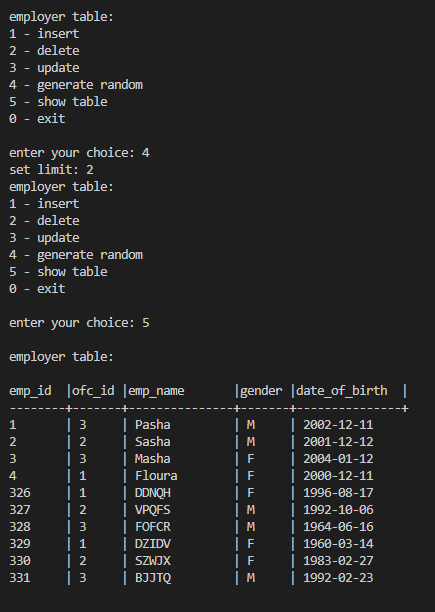
Виконання операції update(employer):

останній рядок(зміна імені з Kasha на Floura)

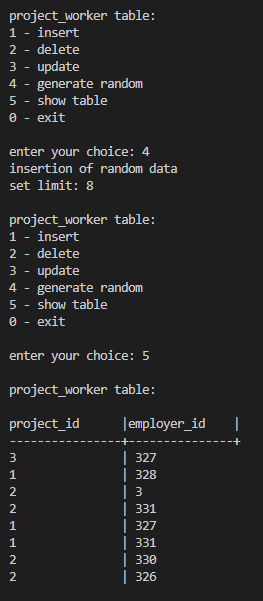


Виконання операції generate random(employer):

останні рядки

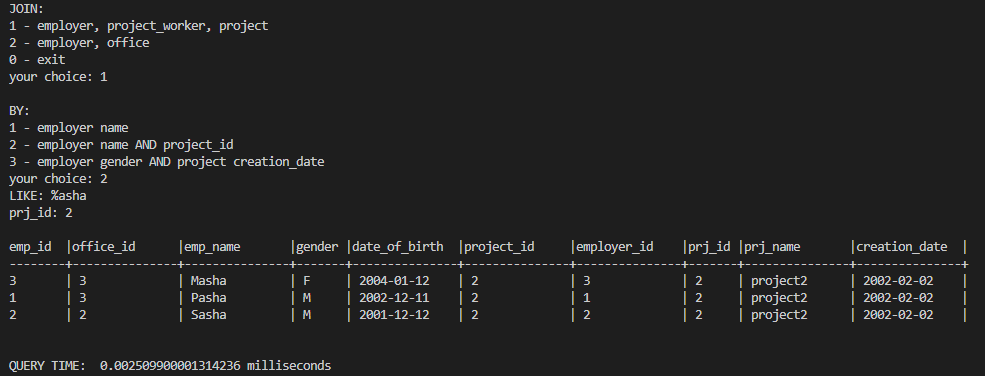


Виконання операції generate random(project\_worker):

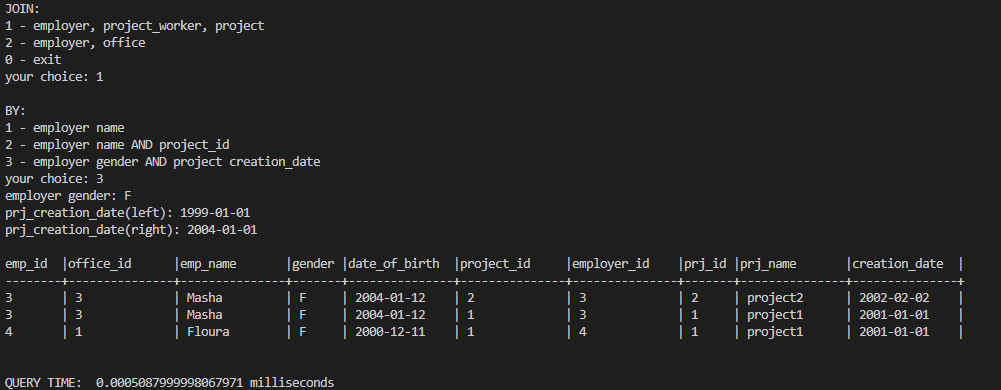


**Пошук з декількох сутностей**

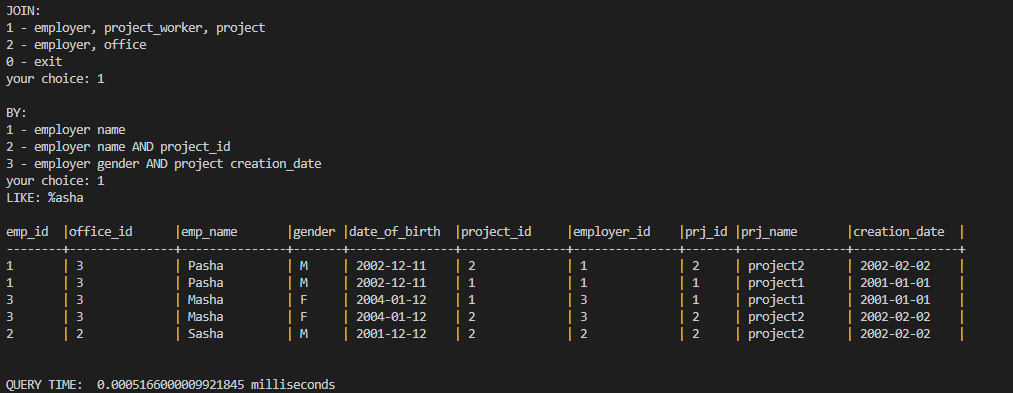
Пошук за параметрами employer name AND project\_id з сутностей employer, project\_worker, projec



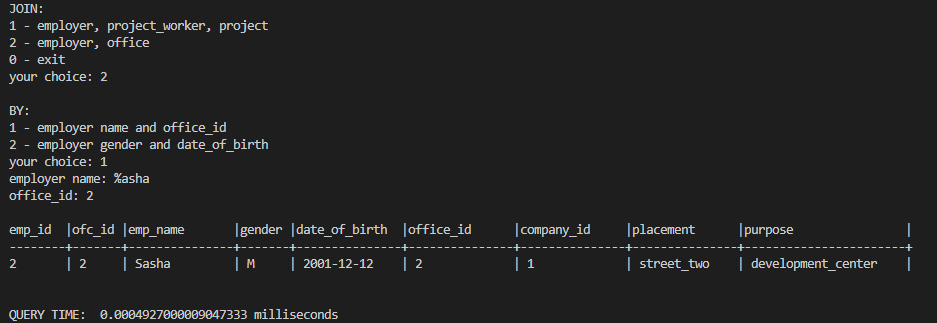
Пошук за параметрами employer gender AND project creation\_date з сутностей employer, project\_worker, project



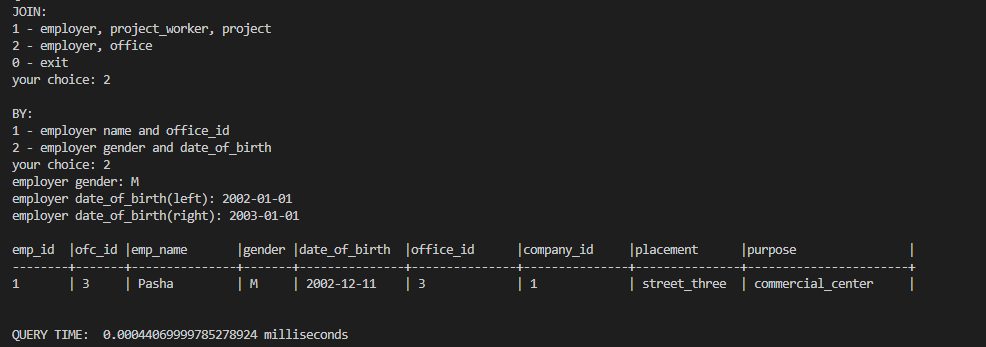
Пошук за параметрами employer name з сутностей employer, project\_worker, project



Пошук за параметрами employer name and office\_id з сутностей employer, office



Пошук за параметрами employer gender and date\_of\_birth з сутностей employer, office



**Код програми:**

employer в модулі view:

def employer():

    f = -1

    while f != 0:

        print('employer table:\n1 - insert\n2 - delete\n3 - update\n4 - generate random\n5 - show table\n0 - exit\n')

        f = int(input('enter your choice: '))

        if f==1:

            employer\_m(1, 0, 0)

        if f==2:

            print('DELETE:\n1 - by name\n2 - by id\n')

            upf = int(input('enter your choice: '))

            if upf==1:

                employer\_m(2, 0, 1)

            if upf==2:

                employer\_m(2, 0, 2)

        if f==3:

            print('SET:\n1 - name\n2 - gender\n3 - date\_of\_birth\n4 - office\_id')

            sf = int(input('enter your choice: '))

            if sf==1:

                print('UPDATE:\n1 - by name\n2 - by id\n')

                upf = int(input('enter your choice: '))

                if upf==1:

                    employer\_m(3, 1, 1)

                if upf==2:

                    employer\_m(3, 1, 2)

            if sf==2:

                print('UPDATE:\n1 - by name\n2 - by id\n')

                upf = int(input('enter your choice: '))

                if upf==1:

                    employer\_m(3, 2, 1)

                if upf==2:

                    employer\_m(3, 2, 2)

            if sf==3:

                print('UPDATE:\n1 - by name\n2 - by id\n')

                upf = int(input('enter your choice: '))

                if upf==1:

                    employer\_m(3, 3, 1)

                if upf==2:

                    employer\_m(3, 3, 2)

            if sf==4:

                print('UPDATE:\n1 - by name\n2 - by id\n')

                upf = int(input('enter your choice: '))

                if upf==1:

                    employer\_m(3, 4, 1)

                if upf==2:

                    employer\_m(3, 4, 2)

        if f==4:

            employer\_m(4, 0, 0)

        if f==5:

            employer\_m(5, 0, 0)

employer в модулі model:

def employer\_m(f, sf, upf):

    ins\_scr = 'INSERT INTO employer (office\_id, name, gender, date\_of\_birth) VALUES (%s, %s, %s, %s)'

    if f==1:

        ins\_data = (int(input('enter office\_id: ')), input('enter name: '), input('enter gender: '), input('enter date\_of\_birth: '))

        try:

            cur.execute(ins\_scr, ins\_data)

        except Exception as error:

            print(error)

        conn.commit()

    if f==2:

        if upf==1:

            del\_scr = "DELETE FROM employer where name=%s;"

            del\_scr1 = "DELETE FROM project\_worker where employer\_id=(SELECT id FROM employer where name=%s);"

            del\_data = (input('enter name: '),)

            try:

                cur.execute(del\_scr1, del\_data)

                cur.execute(del\_scr, del\_data)

            except Exception as error:

                print(error)

            conn.commit()

        if upf==2:

            del\_scr = "DELETE FROM employer where id=%s;"

            del\_scr1 = "DELETE FROM project\_worker where employer\_id=%s;"

            del\_data = (input('enter id: '),)

            try:

                cur.execute(del\_scr1, del\_data)

                cur.execute(del\_scr, del\_data)

            except Exception as error:

                print(error)

            conn.commit()

    if f==3:

        if sf==1:

            if upf==1:

                set\_data = (input('enter name: '), input('by name: '))

                up\_scr = "UPDATE employer SET name=%s where name=%s;"

            if upf==2:

                set\_data = (input('enter name: '), input('by id: '))

                up\_scr = "UPDATE employer SET name=%s where id=%s;"

        if sf==2:

            if upf==1:

                set\_data = (input('enter gender: '), input('by name: '))

                up\_scr = "UPDATE employer SET gender=%s where name=%s;"

            if upf==2:

                set\_data = (input('enter gender: '), input('by id: '))

                up\_scr = "UPDATE employer SET gender=%s where id=%s;"

        if sf==3:

            if upf==1:

                set\_data = (input('enter date\_of\_birth: '), input('by name: '))

                up\_scr = "UPDATE employer SET date\_of\_birth=%s where name=%s;"

            if upf==2:

                set\_data = (input('enter date: '), input('by id: '))

                up\_scr = "UPDATE employer SET date\_of\_birth=%s where id=%s;"

        if sf==4:

            if upf==1:

                set\_data = (input('enter office\_id: '), input('by name: '))

                up\_scr = "UPDATE employer SET office\_id=%s where name=%s;"

            if upf==2:

                set\_data = (input('enter office\_id: '), input('by id: '))

                up\_scr = "UPDATE employer SET office\_id=%s where id=%s;"

        try:

            cur.execute(up\_scr, set\_data)

        except Exception as error:

            print(error)

        conn.commit()

    if f==4:

        rand\_scr = ("INSERT INTO employer (office\_id, name, gender, date\_of\_birth) select office.id, chr((random()\*25+65)::int)||chr((random()\*25+65)::int)||chr((random()\*25+65)::int)||chr((random()\*25+65)::int)||chr((random()\*25+65)::int), chr((random()\*0+((random()\*1+10)::int)\*7)::int), date(timestamp '1940-01-01' + random() \* (timestamp '2004-12-20' - timestamp '1940-01-01')) from office, generate\_series(1,%s);")

        rand\_data = (input('set limit: '),)

        try:

            cur.execute(rand\_scr, rand\_data)

        except Exception as error:

            print(error)

        conn.commit()

    if f==5:

        print('\nemployer table:')

        try:

            cur.execute("SELECT \* from employer order by id")

            print('\nemp\_id  |ofc\_id |emp\_name       |gender |date\_of\_birth  |')

            print('--------+-------+---------------+-------+---------------+')

            for flow in cur.fetchall():

                print(flow[0], '\t|', flow[1], '\t|', flow[2], '\t|', flow[3], '\t|', flow[4],)

            print('\n')

        except Exception as error:

            print(error)

office в модулі view:

def office():

    f = -1

    while f!=0:

        print('office table:\n1 - insert\n2 - delete\n3 - update\n4 - generate random\n5 - show table\n0 - exit\n')

        f = int(input('enter your choice: '))

        if f==1:

            office\_m(1, 0, 0)

        if f==2:

            print('DELETE:\n1 - by id\n0 - exit')

            upf = int(input('enter your choice: '))

            if upf==1:

                office\_m(2, 0, 1)

        if f==3:

            print('SET:\n1 - placment\n2 - purpose\n')

            sf = int(input('enter your choice: '))

            if sf==1:

                print('UPDATE:\n1 - by placment\n2 - by id\n3 - purpose')

                upf = int(input('enter your choice: '))

                if upf==1:

                    office\_m(3, 1, 1)

                if upf==2:

                    office\_m(3, 1, 2)

                if upf==3:

                    office\_m(3, 1, 3)

            if sf==2:

                print('UPDATE:\n1 - by placment\n2 - by id\n3 - purpose')

                upf = int(input('enter your choice: '))

                if upf==1:

                    office\_m(3, 2, 1)

                if upf==2:

                    office\_m(3, 2, 2)

                if upf==3:

                    office\_m(3, 2, 3)

        if f==4:

            office\_m(4, 0, 0)

        if f==5:

            office\_m(5, 0, 0)

office в модулі model:

def office\_m(f, sf, upf):

    ins\_scr = 'INSERT INTO office (company\_id, placement, purpose) VALUES (%s, %s, %s)'

    if f==1:

        ins\_data = (int(input('enter company\_id: ')), input('enter placement: '), input('enter purpose: '))

        try:

            cur.execute(ins\_scr, ins\_data)

        except Exception as error:

            print(error)

        conn.commit()

    if f==2:

        if upf==1:

            del\_scr = "DELETE FROM office where id=%s;"

            del\_scr1 = "DELETE FROM employer where office\_id=%s;"

            del\_data = (input('enter id: '),)

            try:

                cur.execute(del\_scr1, del\_data)

                cur.execute(del\_scr, del\_data)

            except Exception as error:

                print(error)

            conn.commit()

    if f==3:

        if sf==1:

            if upf==1:

                set\_data = (input('enter placment '), input('by placment: '))

                up\_scr = "UPDATE office SET placement=%s where placment=%s;"

            if upf==2:

                set\_data = (input('enter placment: '), input('by id: '))

                up\_scr = "UPDATE office SET placement=%s where id=%s;"

            if upf==3:

                set\_data = (input('enter placment: '), input('by purpose: '))

                up\_scr = "UPDATE office SET placement=%s where purpose=%s;"

        if sf==2:

            if upf==1:

                set\_data = (input('enter purpose '), input('by placment: '))

                up\_scr = "UPDATE office SET purpose=%s where placment=%s;"

            if upf==2:

                set\_data = (input('enter purpose: '), input('by id: '))

                up\_scr = "UPDATE office SET purpose=%s where id=%s;"

            if upf==3:

                set\_data = (input('enter purpose: '), input('by purpose: '))

                up\_scr = "UPDATE office SET purpose=%s where purpose=%s;"

        try:

            cur.execute(up\_scr, set\_data)

        except Exception as error:

            print(error)

        conn.commit()

    if f==4:

        print('insertion of random data')

        rand\_data = (int(input('set limit: ')),)

        rand\_scr = "INSERT INTO office (company\_id, placement, purpose) select 1, chr((random()\*25+65)::int)||chr((random()\*25+65)::int)||chr((random()\*25+65)::int), chr((random()\*25+65)::int)||chr((random()\*25+65)::int)||chr((random()\*25+65)::int) from generate\_series(1,%s);"

        try:

            cur.execute(rand\_scr, rand\_data)

        except Exception as error:

            print(error)

        conn.commit()

    if f==5:

        print('\noffice table:')

        try:

            cur.execute("SELECT \* from office;")

            print('\noffice\_id       |company\_id     |placement      |purpose                |')

            print('----------------+---------------+---------------+-----------------------+')

            for flow in cur.fetchall():

                print(flow[0], '\t\t|', flow[1], '\t\t|', flow[2], '\t|', flow[3],)

            print('\n')

        except Exception as error:

            print(error)

project\_worker в модулі view:

def project\_worker():

    f = -1

    while f!=0:

        print('\nproject\_worker table: \n1 - insert\n2 - delete\n3 - update\n4 - generate random\n5 - show table\n0 - exit\n')

        f = int(input('enter your choice: '))

        if f==1:

            project\_worker\_m(1, 0)

        if f==2:

            print('DELETE:\n1 - by project\_id\n2 - by employer\_id\n3 - by both\_id\n0 - exit')

            upf = int(input('enter your choice: '))

            if upf==3:

                project\_worker\_m(2, 1)

            if upf==2:

                project\_worker\_m(2, 2)

            if upf==3:

                project\_worker\_m(2, 3)

            conn.commit()

        if f==3:

            print('SET:\n1 - project\_id\n2 - employer\_id\n')

            upf = int(input('enter your choice: '))

            if upf==1:

                project\_worker\_m(3, 1)

            if upf==2:

                project\_worker\_m(3, 2)

        if f==4:

            project\_worker\_m(4, 0)

        if f==5:

            project\_worker\_m(5, 0)

project\_worker в модулі model:

def project\_worker\_m(f, upf):

    if f==1:

        ins\_scr = 'INSERT INTO project\_worker (project\_id, employer\_id) VALUES (%s, %s)'

        ins\_data = (int(input('enter project\_id: ')), int(input('enter employer\_id: ')))

        try:

            cur.execute(ins\_scr, ins\_data)

        except Exception as error:

            print(error)

        conn.commit()

    if f==2:

        if upf==1:

            del\_scr = "DELETE FROM project\_worker where project\_id=%s;"

            del\_data = (input('enter project\_id: '),)

            try:

                cur.execute(del\_scr, del\_data)

            except Exception as error:

                print(error)

        if upf==2:

            del\_scr = "DELETE FROM project\_worker where employer\_id = %s;"

            del\_data = (input('enter employer\_id: '),)

            try:

                cur.execute(del\_scr, del\_data)

            except Exception as error:

                print(error)

        if upf==3:

            del\_scr = "DELETE FROM project\_worker where project\_id=%s and employer\_id = %s;"

            del\_data = (input('enter project\_id: '), input('enter employer\_id: '))

            try:

                cur.execute(del\_scr, del\_data)

            except Exception as error:

                print(error)

        conn.commit()

    if f==3:

        if upf==1:

            up\_scr = "UPDATE project\_worker SET project\_id=%s where project\_id=%s and employer\_id=%s;"

            set\_data = (input('SET project\_id '), input('WHERE project\_id '), input('AND employer\_id '))

        if upf==2:

            up\_scr = "UPDATE project\_worker SET employer\_id=%s where project\_id=%s and employer\_id=%s;;"

            set\_data = (input('SET employer\_id '), input('WHERE project\_id '), input('AND employer\_id '))

        try:

            cur.execute(up\_scr, set\_data)

        except Exception as error:

            print(error)

    if f==4:

        print('insertion of random data')

        hlp\_scr1 = "delete from help\_table;"

        hlp\_scr2 = "insert into help\_table (employer\_id, project\_id) select employer.id, project.id from employer, project;"

        ran\_scr1 = "delete from project\_worker;"

        ran\_scr2 = "insert into project\_worker (project\_id, employer\_id) select help\_table.project\_id, help\_table.employer\_id from help\_table order by random() limit %s;"

        ran\_data = (int(input('set limit: ')),)

        try:

            cur.execute(hlp\_scr1)

            cur.execute(hlp\_scr2)

            cur.execute(ran\_scr1)

            cur.execute(ran\_scr2, ran\_data)

        except Exception as error:

            print(error)

        conn.commit()

    if f==5:

        print('\nproject\_worker table:')

        try:

            cur.execute("SELECT \* from project\_worker;")

            print('\nproject\_id      |employer\_id    |')

            print('----------------+---------------+')

            for flow in cur.fetchall():

                print(flow[0], '\t\t|', flow[1],)

            print('\n')

        except Exception as error:

            print(error)

project в модулі view:

def project():

    f = -1

    while f!=0:

        print('project table:\n1 - insert\n2 - delete\n3 - update\n4 - generate random\n5 - show table\n0 - exit\n')

        f = int(input('enter your choice: '))

        if f==1:

            project\_m(1, 0)

        if f==2:

            project\_m(2, 0)

        if f==3:

            print('UPDATE:\n1 - id\n2 - name\n3 - creation\_date\n0 - exit\n')

            upf = int(input('enter your choice: '))

            if upf==1:

                project\_m(3, 1)

            if upf==2:

                project\_m(3, 2)

            if upf==3:

                project\_m(3, 3)

        if f==4:

            project\_m(4, 0)

        if f==5:

            project\_m(5, 0)

project в модулі model:

def project\_m(f, upf):

    if f==1:

        ins\_scr = 'INSERT INTO project (name, creation\_date) VALUES (%s, %s)'

        ins\_data = (input('enter name: '), input('enter creation\_date: '))

        try:

            cur.execute(ins\_scr, ins\_data)

        except Exception as error:

            print(error)

        conn.commit()

    if f==2:

        del\_scr = "DELETE FROM project where id = %s;"

        del\_scr1 = "DELETE FROM project\_worker where project\_id = %s;"

        del\_data = (input('enter id: '),)

        try:

            cur.execute(del\_scr1, del\_data)

            cur.execute(del\_scr, del\_data)

        except Exception as error:

            print(error)

        conn.commit()

    if f==3:

        if upf==1:

            up\_scr = "UPDATE project SET id=%s where id=%s;"

            set\_data = (input('SET id '), input('WHERE id '), )

        if upf==2:

            up\_scr = "UPDATE project SET name=%s where id=%s;"

            set\_data = (input('SET name '), input('WHERE id '), )

        if upf==3:

            up\_scr = "UPDATE project SET creation\_date=%s where id=%s;"

            set\_data = (input('SET creation\_date '), input('WHERE id '), )

        try:

            cur.execute(up\_scr, set\_data)

        except Exception as error:

            print(error)

        conn.commit()

    if f==4:

        print('isertion of random data\n')

        rand\_data = (int(input('set limit: ')), )

        rand\_scr = "INSERT INTO project (name, creation\_date) SELECT chr((random()\*25+65)::int)||chr((random()\*25+65)::int)||chr((random()\*25+65)::int), date(timestamp '2001-01-01' + random() \* (timestamp '2004-12-20' - timestamp '2001-01-01')) from generate\_series(1,%s);"

        try:

            cur.execute(rand\_scr, rand\_data)

        except Exception as error:

            print(error)

        conn.commit()

    if f==5:

        print('\nproject table:')

        try:

            cur.execute("SELECT \* from project;")

            print('\nid      |name                   |creation\_date  |')

            print('--------+-----------------------+---------------+')

            for flow in cur.fetchall():

                print(flow[0], '\t|', flow[1], '\t\t|', flow[2],)

            print('\n')

        except Exception as error:

            print(error)

joins в модулі view:

def joins():

    f=-1

    while f!=0:

        print('JOIN:\n1 - employer, project\_worker, project\n2 - employer, office\n0 - exit')

        f = int(input('your choice: '))

        if f==1:

            print('\nBY:\n1 - employer name\n2 - employer name AND project\_id\n3 - employer gender AND project creation\_date')

            by\_f = int(input('your choice: '))

            if by\_f==1:

                joins\_m(1,1)

            if by\_f==2:

                joins\_m(1,2)

            if by\_f==3:

                joins\_m(1,3)

        if f==2:

            print('\nBY:\n1 - employer name and office\_id\n2 - employer gender and date\_of\_birth')

            by\_f = int(input('your choice: '))

            if by\_f==1:

                joins\_m(2,1)

            if by\_f==2:

                joins\_m(2,2)

joins в модулі model:

def joins\_m(f, by\_f):

    if f==1:

        if by\_f==1:

            join\_data = (input('LIKE: '), )

            try:

                begin\_t = time.perf\_counter()

                cur.execute("SELECT \* from employer as em INNER JOIN project\_worker as pw on em.id=pw.employer\_id INNER JOIN project as prj on pw.project\_id=prj.id where em.name LIKE %s;", join\_data)

                end\_t = time.perf\_counter()

                print('\nemp\_id  |office\_id      |emp\_name       |gender |date\_of\_birth  |project\_id     |employer\_id    |prj\_id |prj\_name       |creation\_date  |')

                print('--------+---------------+---------------+-------+---------------+---------------+---------------+-------+---------------+---------------+')

                for flow in cur.fetchall():

                    print(flow[0], '\t|', flow[1], '\t\t|', flow[2], '\t|', flow[3], '\t|', flow[4], '\t|', flow[5], '\t\t|', flow[6], '\t\t|', flow[7], '\t|', flow[8], '\t|', flow[9], '\t|',)

                print('\n')

                print('QUERY TIME: ',end\_t-begin\_t, "milliseconds")

            except Exception as error:

                print(error)

        if by\_f==2:

            join\_data = (input('LIKE: '), input('prj\_id: '))

            try:

                begin\_t = time.perf\_counter()

                cur.execute("SELECT \* from employer as em INNER JOIN project\_worker as pw on em.id=pw.employer\_id INNER JOIN project as prj on pw.project\_id=prj.id where em.name LIKE %s and prj.id=%s;", join\_data)

                end\_t = time.perf\_counter()

                print('\nemp\_id  |office\_id      |emp\_name       |gender |date\_of\_birth  |project\_id     |employer\_id    |prj\_id |prj\_name       |creation\_date  |')

                print('--------+---------------+---------------+-------+---------------+---------------+---------------+-------+---------------+---------------+')

                for flow in cur.fetchall():

                    print(flow[0], '\t|', flow[1], '\t\t|', flow[2], '\t|', flow[3], '\t|', flow[4], '\t|', flow[5], '\t\t|', flow[6], '\t\t|', flow[7], '\t|', flow[8], '\t|', flow[9], '\t|',)

                print('\n')

                print('QUERY TIME: ',end\_t-begin\_t, "milliseconds")

            except Exception as error:

                print(error)

        if by\_f==3:

            join\_data = (input('employer gender: '), input('prj\_creation\_date(left): '), input('prj\_creation\_date(right): '))

            try:

                begin\_t = time.perf\_counter()

                cur.execute("SELECT \* from employer as em INNER JOIN project\_worker as pw on em.id=pw.employer\_id INNER JOIN project as prj on pw.project\_id=prj.id where em.gender=%s and prj.creation\_date between %s and %s;", join\_data)

                end\_t = time.perf\_counter()

                print('\nemp\_id  |office\_id      |emp\_name       |gender |date\_of\_birth  |project\_id     |employer\_id    |prj\_id |prj\_name       |creation\_date  |')

                print('--------+---------------+---------------+-------+---------------+---------------+---------------+-------+---------------+---------------+')

                for flow in cur.fetchall():

                    print(flow[0], '\t|', flow[1], '\t\t|', flow[2], '\t|', flow[3], '\t|', flow[4], '\t|', flow[5], '\t\t|', flow[6], '\t\t|', flow[7], '\t|', flow[8], '\t|', flow[9], '\t|',)

                print('\n')

                print('QUERY TIME: ',end\_t-begin\_t, "milliseconds")

            except Exception as error:

                print(error)

    if f==2:

        if by\_f==1:

            join\_data = (input('employer name: '), int(input('office\_id: ')))

            try:

                begin\_t = time.perf\_counter()

                cur.execute("SELECT \* from employer as em INNER JOIN office as ofc on ofc.id=em.office\_id where em.name LIKE %s and office\_id=%s;", join\_data)

                end\_t = time.perf\_counter()

                print('\nemp\_id  |ofc\_id |emp\_name       |gender |date\_of\_birth  |office\_id      |company\_id     |placement      |purpose                |')

                print('--------+-------+---------------+-------+---------------+---------------+---------------+---------------+-----------------------+')

                for flow in cur.fetchall():

                    print(flow[0], '\t|', flow[1], '\t|', flow[2], '\t|', flow[3], '\t|', flow[4], '\t|', flow[5], '\t\t|', flow[6], '\t\t|', flow[7], '\t|', flow[8], '\t|',)

                print('\n')

                print('QUERY TIME: ',end\_t-begin\_t, "milliseconds")

            except Exception as error:

                print(error)

        if by\_f==2:

            join\_data = (input('employer gender: '), input('employer date\_of\_birth(left): '), input('employer date\_of\_birth(right): '))

            try:

                begin\_t = time.perf\_counter()

                cur.execute("SELECT \* from employer as em INNER JOIN office as ofc on ofc.id=em.office\_id where em.gender=%s and em.date\_of\_birth between %s and %s;", join\_data)

                end\_t = time.perf\_counter()

                print('\nemp\_id  |ofc\_id |emp\_name       |gender |date\_of\_birth  |office\_id      |company\_id     |placement      |purpose                |')

                print('--------+-------+---------------+-------+---------------+---------------+---------------+---------------+-----------------------+')

                for flow in cur.fetchall():

                    print(flow[0], '\t|', flow[1], '\t|', flow[2], '\t|', flow[3], '\t|', flow[4], '\t|', flow[5], '\t\t|', flow[6], '\t\t|', flow[7], '\t|', flow[8], '\t|',)

                print('\n')

                print('QUERY TIME: ',end\_t-begin\_t, "milliseconds")

            except Exception as error:

                print(error)

Частина коду, що відповідає за підключення до БД:

try:

    conn = psycopg2.connect(

        host = hostname,

        dbname = database,

        user = username,

        password = pwd,

        port = port\_id

        )

    cur = conn.cursor()

except Exception as error:

    print(error)

def close\_connection():

    if cur is not None:

        print('closing cursor')

        cur.close()

    if conn is not None:

        print('closing connection')

        conn.close()