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»

Тема: «Проектування бази даних та ознайомлення з базовими операціями СУБД PostgreSQL»

Виконав: студент III курсу

ФПМ групи КВ-94

Чекмезов Г. В.

Перевірив: Петрашенко А.В.

Київ 2021

Загальне завдання роботи:

1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. 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.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

GitHub: https://github.com/glebbovski/DataBase/tree/main/Lab2

Мова програмування: Python.

Використані бібліотеки: psycopg2, time

“Сутність-зв’язок”

Сутності

Електронна пошта

Користувачі

Повідомлення

Папки

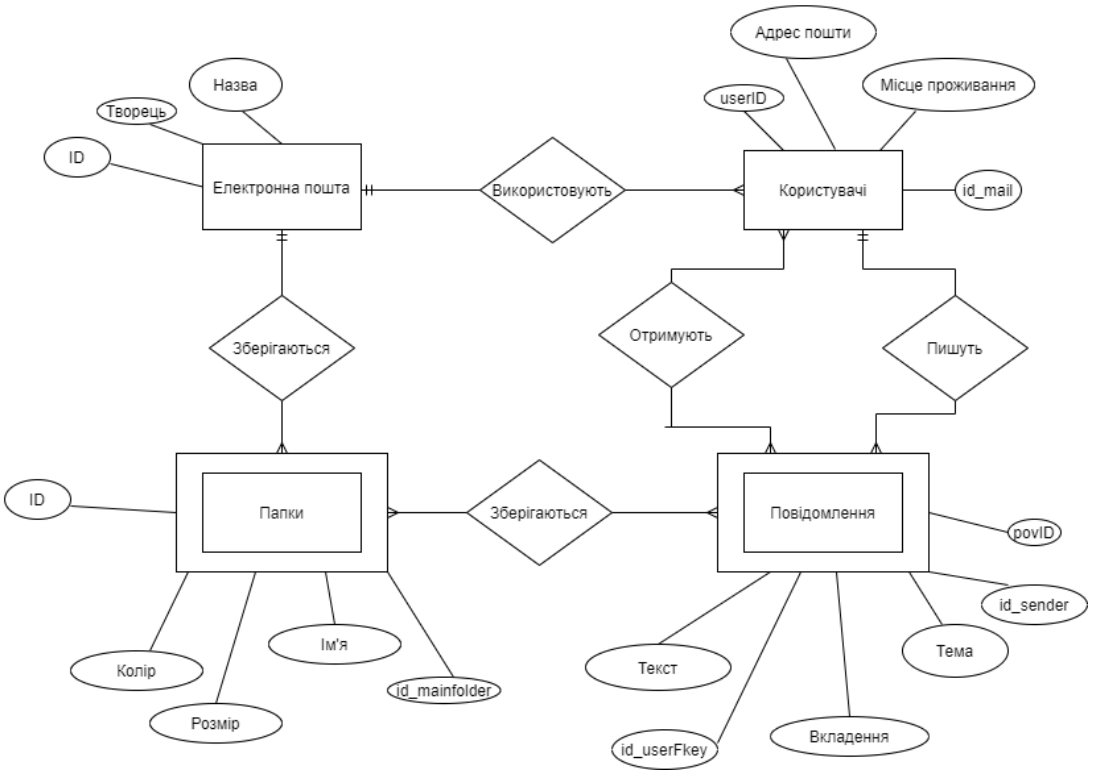
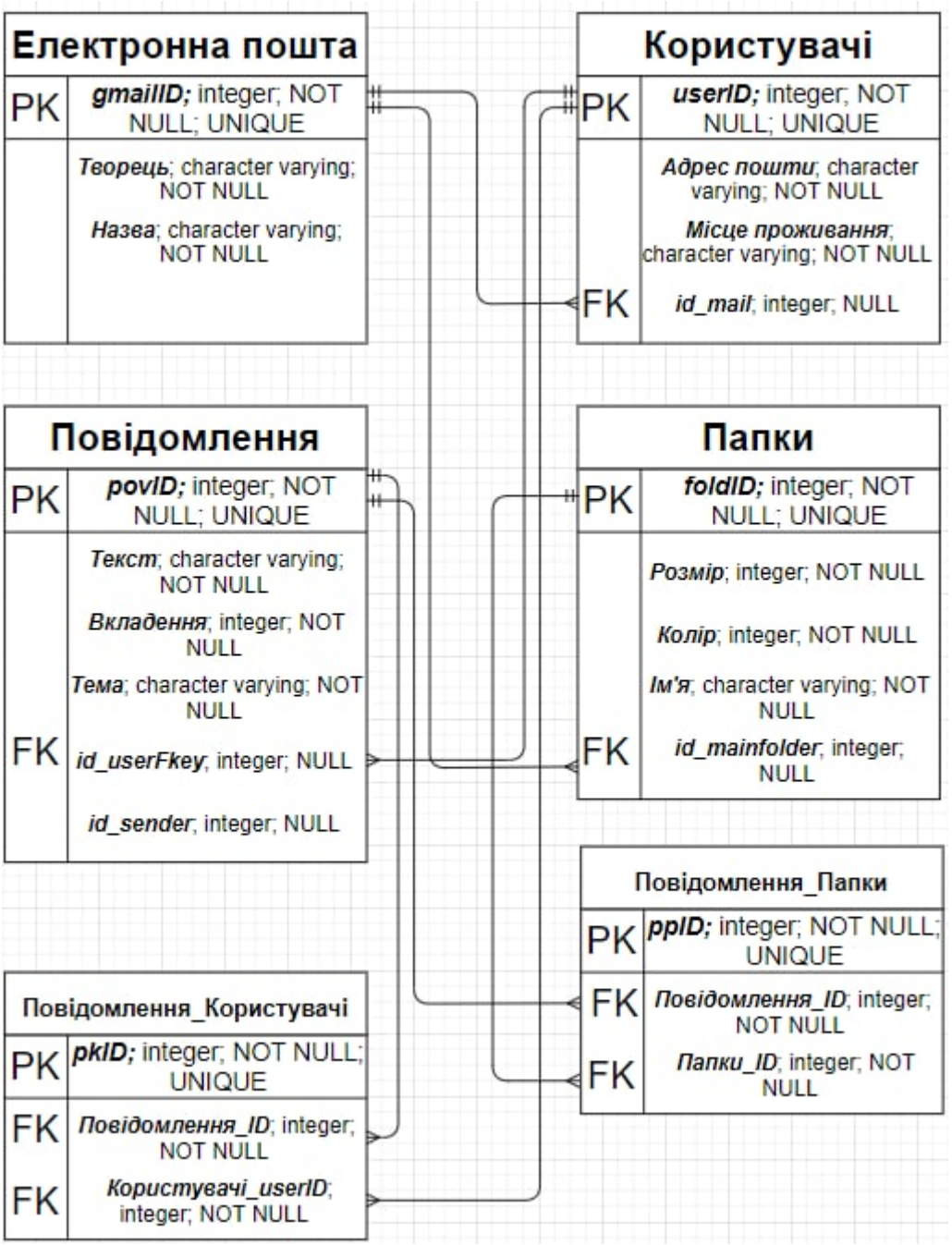


Схема бази даних у графічному вигляді:



Опис бази даних:

У даному випадку маємо 4 сутності: Електронна пошта, Користувачі, Повідомлення, Папки.

Перша сутність “Електронна пошта” потрібна для оброблення інформації, яку саме електронну пошту використовує користувач у даний момент часу (хто є автором даної пошти, її назва).

Друга сутність – “Користувачі”. Використовується для ведення обліку користувачів пошти шляхом ідентифікації. Також містить інформацію про унікальний поштовий адрес кожного користувача та про місце проживання.

Третя сутність називається “Повідомлення”. Використовується для ведення обліку усіх повідомлень, відправлених чи отриманих певним користувачем та визначення окремих особливостей пошти, таких як: ID, тему, вкладення та текст.

Четверта сутність – “Папки”. Необхідна для ведення обліку папок, які містять різні повідомлення, на певній електронній пошті. Має такі характерні риси як колір, ім’я, розмір та ID.

Опис меню програми:

Меню складається з 9 пунктів:

```
1 => One table
2 => All tables
3 => Insertion
4 => Delete some inf
5 => Updating
6 => Selection
7 => Searching
8 => Random inf
...
0 = > Exit
```

1) One table – вивід на екран однієї таблиці, яку обере користувач.
2) All tables – вивід на екран усіх таблиць.
3) Insertion – вставка у вибрану користувачем таблицю нового рядка.
4) Delete some inf – видалення одного або декількох рядків з обраної таблиці.

5) Updating – оновлення даних у будь-якому рядку, який обере користувач у конкретній таблиці.
6) Selection – формування запитів для фільтрації трьома способами.
7) Random inf – заповнення таблиць випадковими даними.
8) Exit – завершення роботи програми.

Завдання 1

Insert

На прикладі батьківської таблиці Email та дочірньої Folders

Запис у Email:

```
Choose your table: 1
gmailID = 5
creator(str) = tyu
name(str) = fgh
email
SQL query =  DO $$ BEGIN if (1=1) and not exists (s
['ЗАМЕЧАНИЕ:  added\n']
1 => Continue insertion, 2 => Stop insertion =>
```

```
**************************

gmailID      creator      name
8            user            gleb
3            indexenjoyer              revolution
1            rty            fgh
9            GZ            BQ
10            GI            KQ
11            FK            MR
5            tyu            fgh
```

Запис рядка с первинним ключем, який вже знаходиться у таблиці:

```
Choose your table: 1
gmailID = 5
creator(str) = dfg
name(str) = sdf
email
SQL query =  DO $$ BEGIN if (1=1) and not exists (select gmailID from
['ЗАМЕЧАНИЕ:  wrong way, the row with gmailID = 5 exists\n']
1 => Continue insertion, 2 => Stop insertion =>
```

Спроба запису у дочірню таблицю з вторинним ключем, який не відповідає первинному батьківської:

```
Choose your table: 2
foldID = 1
Size = 5
Colour = 6
nameof(str) = yui
id_mainfolder = 27
folders
SQL query =  DO $$ BEGIN IF EXISTS (select gmailID from email where gmailID = 27) and not e
['ЗАМЕЧАНИЕ:  gmailID = 27 is not present in table or row with foldID = 1 exists already\n']
1 => Continue insertion, 2 => Stop insertion =>
```

Запис з ключем, який відповідає первинному:

```
Choose your table: 2
foldID = 1
Size = 2
Colour = 3
nameof(str) = енг
id_mainfolder = 5
folders
SQL query =  DO $$ BEGIN IF EXISTS (select gmailID fror
['ЗАМЕЧАНИЕ:  added\n']
1 => Continue insertion, 2 => Stop insertion =>
```

```
foldID       Size      Colour      nameof      id_mainfolder
30            13           3           QR            1
31            69           199           KG            9
1            2            3           енг            5
****************************
```

Лістинг для Insert:

```
def insertbyusertoEmail(f,s,t,added,notice):
    connect = connection.connection()
    cursor = connect.cursor()
    insert = 'DO $$ BEGIN if (1=1) and not exists (select gmailID from email
where gmailID = {}) then INSERT INTO email(gmailID, creator, name) VALUES
({},{},{}); ' \
             'raise notice {}; else raise notice {}; ' \
             'end if; end $$;'.format(f, f, s, t, added, notice)
    cursor.execute(insert)
    connect.commit()
    print(connect.notices)
    cursor.close()
    connection.connectionlost(connect)

@staticmethod
def insertbyusertoFolders(f,s,t,fouth, fifth, added,notice):
    connect = connection.connection()
    cursor = connect.cursor()
    insert = 'DO $$    BEGIN IF EXISTS (select gmailID from email where gmailID
= {}) and not exists (select foldId from folders where foldId = {}) THEN ' \
             'INSERT INTO folders(foldID, Size, Colour, nameof, id_mainfolder)
values ({}, {}, {}, {}, {}); ' \
             'RAISE NOTICE {};' \
             ' ELSE RAISE NOTICE {};' \
             'END IF; ' \
             'END $$;'.format(fifth, f, f, s, t, fouth, fifth, added, notice)
    cursor.execute(insert)
    connect.commit()
    print(connect.notices)
    cursor.close()
    connection.connectionlost(connect)

@staticmethod
def insertbyusertoUsers(f,s,t,fouth,added,notice):
    connect = connection.connection()
    cursor = connect.cursor()
    insert = 'DO $$ BEGIN IF EXISTS (select gmailID from email where gmailID =
{}) and not exists (select userID from users where userID = {}) THEN ' \
             'INSERT INTO users(userID, adress, place, id_mail) values ({}, {},
{}, {}); ' \
             'RAISE NOTICE {};' \
             ' ELSE RAISE NOTICE {};' \
             'END IF; ' \
             'END $$;'.format(fouth, f, f, s, t, fouth, added, notice)
    cursor.execute(insert)
    connect.commit()
    print(connect.notices)
    cursor.close()
    connection.connectionlost(connect)

@staticmethod
def insertbyusertoNotifications(f,s,t,fouth,fifth,sixth,added,notice):
    connect = connection.connection()
    cursor = connect.cursor()
    insert = 'DO $$    BEGIN IF EXISTS (select userID from users where userID =
{}) and not exists (select povID from notifications where povID = {}) THEN '
\
```

```
            'INSERT INTO notifications(povID, text, addfiles, title,
id_userfkey, id_sender) values ({}, {}, {}, {}, {}, {}); ' \
            'RAISE NOTICE {};' \
            ' ELSE RAISE NOTICE {};' \
            'END IF; ' \
            'END $$;'.format(fifth, f, f, s, t, fouth, fifth, sixth, added,
notice)
   cursor.execute(insert)
   connect.commit()
   print(connect.notices)
   cursor.close()
   connection.connectionlost(connect)
```

Update

У нашому випадку редагування ключів є неможливим

```
foldID        Size       Colour       nameof       id_mainfolder
30             13             3             QR             1
31             69             199             KG             9
1             2             3             енг             5
**************************

Row to update where foldID = 1
Size = 5
Colour = 7
nameof(str) = puppy
folders
SQL query =  DO $$ BEGIN IF EXISTS (select foldID from folder
['ЗАМЕЧАНИЕ:  updated\n']
1 => Continue update, 2 => Stop update =>
```

```
foldID       Size      Colour      nameof      id_mainfolder
30            13           3           QR            1
31            69           199           KG            9
1             5            7           puppy          5
**************************
```

При спробі редагувати рядок, якого не існує:

```
Row to update where foldID = 5
Size = 1
Colour = 2
nameof(str) = фів
folders
SQL query =>  DO $$ BEGIN IF EXISTS (select
['ЗАМЕЧАНИЕ:  foldID = 5 is not present in t
1 => Continue update, 2 => Stop update =>
```

Лістинг для Update:

```
@staticmethod
def UpdateEmail(idk, set2, set3, updated, notice):
   connect = connection.connection()
   cursor = connect.cursor()
   update = 'DO $$ BEGIN IF EXISTS (select gmailID from email where gmailID =
{}) THEN ' \
            'update email set creator = {}, name = {} where gmailID = {}; ' \
            'RAISE NOTICE {};' \
            ' ELSE RAISE NOTICE {};' \
            'END IF; ' \
            'END $$;'.format(idk, set2, set3, idk, updated, notice)
   cursor.execute(update)
   connect.commit()
   print(connect.notices)
   cursor.close()
   connection.connectionlost(connect)


@staticmethod
def UpdateFolders(idk, set1, set2, set3, updated, notice):
   connect = connection.connection()
   cursor = connect.cursor()
   update = 'DO $$ BEGIN IF EXISTS (select foldID from folders where foldID =
{})' \
            ' THEN ' \
            'update folders set Size = {}, Colour = {}, nameof = {} where
foldID = {}; ' \
            'RAISE NOTICE {};' \
            ' ELSE RAISE NOTICE {};' \
            'END IF; ' \
            'END $$;'.format(idk, set1, set2, set3, idk, updated, notice)
```

```
    cursor.execute(update)
    connect.commit()
    print(connect.notices)
    cursor.close()
    connection.connectionlost(connect)

@staticmethod
def UpdateUsers(idk, adress, place, updated, notice):
    connect = connection.connection()
    cursor = connect.cursor()
    update = 'DO $$ BEGIN IF EXISTS (select userID from users where userID =
{}) ' \
             ' THEN ' \
             'update users set adress = {}, place = {} where userID = {}; ' \
             'RAISE NOTICE {};' \
             ' ELSE RAISE NOTICE {};' \
             'END IF; ' \
             'END $$;'.format(idk, adress, place, idk, updated, notice)
    cursor.execute(update)
    connect.commit()
    print(connect.notices)
    cursor.close()
    connection.connectionlost(connect)

@staticmethod
def UpdateNotifications(idk, text, addfiles, title, sender, updated, notice):
    connect = connection.connection()
    cursor = connect.cursor()
    update = 'DO $$ BEGIN IF EXISTS (select povID from notifications where
povID = {}) ' \
             ' THEN ' \
             'update notifications set text = {}, addfiles = {}, title = {},
id_sender = {} where povID = {}; ' \
             'RAISE NOTICE {};' \
             ' ELSE RAISE NOTICE {};' \
             'END IF; ' \
             'END $$;'.format(idk, text, addfiles, title, sender, idk,
updated, notice)
    cursor.execute(update)
    connect.commit()
    print(connect.notices)
    cursor.close()
    connection.connectionlost(connect)
```

Delete

На прикладі таблиці Notifications та її дочірніх таблиць notifications_users та folders_notifications

Таблиці до видалення інфомарції:

Notifications:

```
povID       text      addfiles      title      id_userfkey      id_sender
1            WT            100           EN            2            26
2            UB            138           MF            3            205
**************************
SQL query =>  select * from public.folders_notifications

****************************

ppID       notifications_ID     folders_ID
1             1             1
2             2             1
****************************
 ****************************

 pkID       notifications_ID     users_ID
 1             2             3
 2             1             2
 3             1             2
 4             1             3
 5             1             2
 ****************************
```

Видалення:

```
 povID       text      addfiles      title      id_userfkey      id_sender
 1            WT            100           EN            2            26
 **************************

Choose your table: 4
Attribute to delete povID = 2
notifications
SQL query =>  DO $$ BEGIN if exists (select povID from notifi
['ЗАМЕЧАНИЕ:  deleted\n']
1 => Continue delete, 2 => Stop delete => 2
Continue to work with db => 1, stop => 2. Your choice =>
```

```
**************************

povID       text      addfiles      title      id_userfkey      id_sender
1            WT            100           EN            2             26
**************************

 ppID       notifications_ID     folders_ID
 1            1             1
 ****************************

  ****************************

 pkID       notifications_ID     users_ID
 2            1             2
 3            1             2
 4            1             3
 5            1             2
 ****************************
```

При спробі видалення неіснуючого рядка:

```
Choose your table: 4
Attribute to delete povID = 2
notifications
SQL query =>  DO $$ BEGIN if exists (select povID
['ЗАМЕЧАНИЕ:  something went wrong\n']
1 => Continue delete, 2 => Stop delete =>
```

Лістинг Delete:

```
@staticmethod
def deleteEmail(idk, delete, notice):
   connect = connection.connection()
   cursor = connect.cursor()
   delete = 'DO $$ BEGIN IF EXISTS (select gmailID from email where gmailID =
{}) then ' \
            'delete from folders_notifications where folders_ID in (select
foldID from folders where id_mainfolder = {});' \
            'delete from folders_notifications where notifications_ID in
(select povID from notifications where id_userfkey in (select userID from
users where id_mail = {}));' \
```

```
            'delete from notifications_users where notifications_ID in (select
povID from notifications where id_userfkey in (select userID from users where
id_mail = {}));' \
            'delete from notifications_users where users_ID in (select userID
from users where id_mail = {});' \
            'delete from notifications where id_userfkey in (select userID
from users where id_mail = {});' \
            'delete from users where id_mail = {};' \
            'delete from folders where id_mainfolder = {};' \
            'delete from email where gmailID= {};' \
            'raise notice {};' \
            'else raise notice {};' \
            'end if;' \
            'end $$;'.format(idk, idk, idk, idk, idk, idk, idk, idk, idk,
delete, notice)
   cursor.execute(delete)
   connect.commit()
   print(connect.notices)
   cursor.close()
   connection.connectionlost(connect)

@staticmethod
def deleteFolders(idk, delete, notice):
   connect = connection.connection()
   cursor = connect.cursor()
   delete = 'DO $$ BEGIN if ' \
            'exists (select foldID from folders where foldID = {}) then ' \
            ' delete from folders_notifications where folders_ID in (select
foldID from folders where id_mainfolder = {});' \
            'delete from folders where foldID= {};' \
            'raise notice {};' \
            'else raise notice {};' \
            'end if;' \
            'end $$;'.format(idk, idk, idk, delete, notice)
   cursor.execute(delete)
   connect.commit()
   print(connect.notices)
   cursor.close()
   connection.connectionlost(connect)

@staticmethod
def deleteUsers(idk, delete, notice):
   connect = connection.connection()
   cursor = connect.cursor()
   delete = 'DO $$ BEGIN if ' \
            'exists (select userID from users where userID = {}) then ' \
            'delete from notifications_users where notifications_ID in (select
povID from notifications ' \
            'where id_userfkey in (select userID from users where userID =
{}));' \
            'delete from notifications_users where users_ID in (select povID
from notifications ' \
            'where id_userfkey = {});' \
            'delete from notifications where id_userfkey = {};' \
            'delete from users where userID = {};' \
            'raise notice {};' \
            'else raise notice {};' \
            'end if;' \
            'end $$;'.format(idk, idk, idk, idk, idk, delete, notice)
   cursor.execute(delete)
```

```
    connect.commit()
    print(connect.notices)
    cursor.close()
    connection.connectionlost(connect)

@staticmethod
def deleteNotifications(idk, delete, notice):
    connect = connection.connection()
    cursor = connect.cursor()
    delete = 'DO $$ BEGIN if exists (select povID from notifications where
povID = {}) then ' \
             'delete from notifications_users where notifications_ID = {};' \
             'delete from folders_notifications where notifications_ID = {};' \
             'delete from notifications where povID= {};' \
             'raise notice {};' \
             'else raise notice {};' \
             'end if;' \
             'end $$;'.format(idk, idk, idk, idk, delete, notice)
    cursor.execute(delete)
    connect.commit()
    print(connect.notices)
    cursor.close()
    connection.connectionlost(connect)
```

Завдання №2

Передбачити автоматичне пакетне генерування “рандомізованих” даних:

На прикладі таблиці Email:

```
**************************

gmailID      creator      name
8            user           gleb
3            indexenjoyer             revolution
1            rty            fgh
9            GZ            BQ
10            GI            KQ
11            FK            MR
5            tyu            fgh
```

```
Choose your table: 1
How much datas do you want to add => 2
email
SQL query =>  INSERT INTO email (Creator, Name) s
Inserted randomly
1 => Continue random, 2 => Stop random =>
```

gmailID	creator	name
8	user	gleb
3	indexenjoyer	revolution
1	rty	fgh
9	GZ	BQ
10	GI	KQ
11	FK	MR
5	tyu	fgh
12	AF	QJ
13	RV	XE

Лістинг:

```
@staticmethod
def randomik(table, kolvo):
        connect = connection.connection()
        cursor = connect.cursor()
        check = True
        while check:
            if table == 1:
                insert = "INSERT INTO email (Creator,
Name) select chr(trunc(65 +
random()*26)::int)||chr(trunc(65 + r" \
                         "andom()*26)::int), " \
```

```
                        "chr(trunc(65 +
random()*26)::int)||chr(trunc(65 + random()*26)::int) "
\
                        "from
generate_series(1,{})".format(kolvo)
                cursor.execute(insert)
                check = False
            elif table == 2:
                res = 0
                while (True):
                    insert = "INSERT INTO folders(Size,
Colour, Nameof, id_mainfolder) select random() * 256,"
\
                        "random() * 256," \
                        "chr(trunc(65 +
random()*26)::int)||chr(trunc(65 + random()*26)::int),"
\
                        "(select gmailID from email
order by random() limit 1)"\
                        "from generate_series(1,1)"
                    cursor.execute(insert)
                    res = res + 1
                    if(res == kolvo):
                        break
                check = False
            elif table == 3:
                res = 0
                while (res != kolvo):
```

```
                    insert = "INSERT INTO users (adress,
place, id_mail) select " \
                             "chr(trunc(65 +
random()*25)::int)||chr(trunc(65 + " \
                             "random()*25)::int), " \
                             "chr(trunc(65 +
random()*25)::int)||chr(trunc(65 + random()*25)::int)," 
\
                             "(select gmailID from email
order by random() limit 1) " \
                             "from generate_series(1,1)"
                    cursor.execute(insert)
                    res = res + 1


                check = False
            elif table == 4:
                res = 0
                while (res != kolvo):
                    insert = "INSERT INTO notifications
(text, addfiles, title, id_userfkey, id_sender) select
" \
                             "chr(trunc(65 +
random()*26)::int)||chr(trunc(65 + r" \
                             "andom()*26)::int), random() *
256," \
                             "chr(trunc(65 +
random()*26)::int)||chr(trunc(65 + random()*26)::int)," 
\
```

```
                                "(select userID from users
order by random() limit 1)," \
                                "random() * 256 " \
                                "from generate_series(1,1)"
                            cursor.execute(insert)
                            res = res + 1
                        check = False
                    check = False
                print(Tables[table])
                print("SQL query => ", insert)
                connect.commit()
                print('Inserted randomly')
                cursor.close()
                connection.connectionlost(connect)
```

Завдання №3

Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно.

```
Your choice is: 6
----------------------------------------
1 => Show size and colour of folders which created by *creator* where name length is greater than *value* or equal
----------------------------------------
2 => Show text and addfiles of user message, where count of addfiles less than *value* on the mail *adress*
----------------------------------------
3 => Show size, colour and nameof of folder, where the message with title *title* is stored
----------------------------------------
Your choice is 1
Enter the required length(int) = 2
Enter required creator(str) = indexenjoyer
SQL query =>  select size, colour, name, creator from (select c.size, c.colour, p.name,
                            p.creator from
                                folders c left join email p
                                 on p.gmailID = c.id_mainfolder where length(p.name) >= 2 and p.creator LIKE 'indexenjoyer'
                                  group by c.size,
                                 c.colour, p.name, p.creator) as foo
**************************

size       colour           name           creator
2            3           revolution            indexenjoyer
89            47           revolution            indexenjoyer
--------------------------
```

```
Your choice is 2
Enter required value(int) = 300
Enter required adress(str) = gmail
SQL query =>  select adress, addfiles, text from (select p.text, p.addfiles, c.adress from
                                        notifications p right join users c on p.id_userfkey = c.userID
                                        where p.addfiles < 300 and c.adress LIKE 'gmail' group by
                                        c.adress, p.addfiles, p.text) as foo

**************************

adress          addfiles          text
gmail           26            poi
**************************

Time of request 4 ms
Selected
1 => Continue selection, 2 => Stop selection =>

Your choice is 3
Enter required email(str) = revolution
SQL query =>  select name, nameof, size, colour, name from (select c.name, p.nameof, p.size, p.colour from
                                    folders p left join email c on c.gmailID=p.id_mainfolder
                                    where c.name LIKE 'revolution' group by c.name, p.nameof, p.size, p.colour) as foo

**************************

name          nameof          size          colour
revolution          Popsa           89           47
revolution          Type           2           3
**************************

Time of request 5 ms
Selected
1 => Continue selection, 2 => Stop selection =>
```

Лістинг:

```
@staticmethod
def selectionone(len, creator):
    connect = connection.connection()
    cursor = connect.cursor()
    select = """select size, colour, name, creator from (select
c.size, c.colour, p.name,
                                        p.creator from
                                            folders c left join email
p
                                             on p.gmailID =
c.id_mainfolder where length(p.name) >= {} and p.creator LIKE
'{}'
                                             group by c.size,
```

```
                                        c.colour, p.name,
p.creator) as foo""".format(len, creator)
    print("SQL query => ", select)
    beg = int(time.time() * 1000)
    cursor.execute(select)
    end = int(time.time() * 1000) - beg
    datas = cursor.fetchall()
    print('Time of request {} ms'.format(end))
    print('Selected')
    cursor.close()
    connection.connectionlost(connect)
    return datas

@staticmethod
def selectiontwo(value, adress):
    connect = connection.connection()
    cursor = connect.cursor()
    select = """select adress, addfiles, text from (select
p.text, p.addfiles, c.adress from
                                                notifications p
right join users c on p.id_userfkey = c.userID
                                                where
p.addfiles < {} and c.adress LIKE '{}' group by
                                                c.adress,
p.addfiles, p.text) as foo
                    """.format(value, adress)
    print("SQL query => ", select)
    beg = int(time.time() * 1000)
    cursor.execute(select)
    end = int(time.time() * 1000) - beg
    datas = cursor.fetchall()
    print('Time of request {} ms'.format(end))
    print('Selected')
```

```
    cursor.close()
    connection.connectionlost(connect)
    return datas

@staticmethod
def selectionthree(title):
    connect = connection.connection()
    cursor = connect.cursor()
    select = """select name, nameof, size, colour, name from
(select c.name, p.nameof, p.size, p.colour from
                                                folders p left
join email c on c.gmailID=p.id_mainfolder
                                                where c.name
LIKE '{}' group by c.name, p.nameof, p.size, p.colour) as foo
                    """.format(title)
    print("SQL query => ", select)
    beg = int(time.time() * 1000)
    cursor.execute(select)
    end = int(time.time() * 1000) - beg
    datas = cursor.fetchall()
    print('Time of request {} ms'.format(end))
    print('Selected')
    cursor.close()
    connection.connectionlost(connect)
    return datas
```

Код програмного модулю “model.py”:

```
import random
import connection
import time


Tables = {
    1: 'email',
    2: 'folders',
    3: 'users',
    4: 'notifications',
    5: 'folders_notifications',
    6: 'notifications_users'
}


class Model:
    @staticmethod
    def existingtable(table):
        if str(table).isdigit():
            table = int(table)
            cons = True
            while cons:
                if table == 1 or table == 2 or table ==
3 or table == 4 or table == 5 or table == 6:
                    return table
                else:
                    print('///Try again.')
                    return 0
        else:
```

```
            print('Try again')
            return 0


    @staticmethod
    def outputonetable(table):
        connect = connection.connection()
        cursor = connect.cursor()
        show = 'select * from
public.{}'.format(Tables[table])
        print("SQL query => ", show)
        print('')
        cursor.execute(show)
        datas = cursor.fetchall()
        cursor.close()
        connection.connectionlost(connect)
        return datas

    @staticmethod
    def insertbyusertoEmail(f,s,t,added,notice):
        connect = connection.connection()
        cursor = connect.cursor()
        insert = 'DO $$ BEGIN if (1=1) and not exists
(select gmailID from email where gmailID = {}) then
INSERT INTO email(gmailID, creator, name) VALUES
({},{},{}); ' \
                 'raise notice {}; else raise notice {};
' \
```

```
                'end if; end $$;'.format(f, f, s, t,
added, notice)
        cursor.execute(insert)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)

    @staticmethod
    def insertbyusertoFolders(f,s,t,fouth, fifth,
added,notice):
        connect = connection.connection()
        cursor = connect.cursor()
        insert = 'DO $$    BEGIN IF EXISTS (select
gmailID from email where gmailID = {}) and not exists
(select foldId from folders where foldId = {}) THEN ' \
                'INSERT INTO folders(foldID, Size,
Colour, nameof, id_mainfolder) values ({}, {}, {}, {},
{}); ' \
                'RAISE NOTICE {};' \
                ' ELSE RAISE NOTICE {};' \
                'END IF; ' \
                'END $$;'.format(fifth, f, f, s, t,
fouth, fifth, added, notice)
        cursor.execute(insert)
        connect.commit()
        print(connect.notices)
        cursor.close()
```

```
        connection.connectionlost(connect)

    @staticmethod
    def insertbyusertoUsers(f,s,t,fouth,added,notice):
        connect = connection.connection()
        cursor = connect.cursor()
        insert = 'DO $$ BEGIN IF EXISTS (select gmailID
from email where gmailID = {}) and not exists (select
userID from users where userID = {}) THEN ' \
                 'INSERT INTO users(userID, adress,
place, id_mail) values ({}, {}, {}, {}); ' \
                 'RAISE NOTICE {};' \
                 ' ELSE RAISE NOTICE {};' \
                 'END IF; ' \
                 'END $$;'.format(fouth, f, f, s, t,
fouth, added, notice)
        cursor.execute(insert)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)

    @staticmethod
    def
insertbyusertoNotifications(f,s,t,fouth,fifth,sixth,add
ed,notice):
        connect = connection.connection()
        cursor = connect.cursor()
```

```
        insert = 'DO $$    BEGIN IF EXISTS (select
userID from users where userID = {}) and not exists
(select povID from notifications where povID = {}) THEN
' \
                 'INSERT INTO notifications(povID, text,
addfiles, title, id_userfkey, id_sender) values ({},
{}, {}, {}, {}, {}); ' \
                 'RAISE NOTICE {};' \
                 ' ELSE RAISE NOTICE {};' \
                 'END IF; ' \
                 'END $$;'.format(fifth, f, f, s, t,
fouth, fifth, sixth, added, notice)
        cursor.execute(insert)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)

    @staticmethod
    def deleteEmail(idk, delete, notice):
        connect = connection.connection()
        cursor = connect.cursor()
        delete = 'DO $$ BEGIN IF EXISTS (select gmailID
from email where gmailID = {}) then ' \
                 'delete from folders_notifications
where folders_ID in (select foldID from folders where
id_mainfolder = {});' \
```

```
                'delete from folders_notifications
where notifications_ID in (select povID from
notifications where id_userfkey in (select userID from
users where id_mail = {}));' \
                'delete from notifications_users where
notifications_ID in (select povID from notifications
where id_userfkey in (select userID from users where
id_mail = {}));' \
                'delete from notifications_users where
users_ID in (select userID from users where id_mail =
{});' \
                'delete from notifications where
id_userfkey in (select userID from users where id_mail
= {});' \
                'delete from users where id_mail = {};'
\
                'delete from folders where
id_mainfolder = {};' \
                'delete from email where gmailID= {};'
\
                'raise notice {};' \
                'else raise notice {};' \
                'end if;' \
                'end $$;'.format(idk, idk, idk, idk,
idk, idk, idk, idk, idk, idk, delete, notice)
        cursor.execute(delete)
        connect.commit()
        print(connect.notices)
```

```
        cursor.close()
        connection.connectionlost(connect)

    @staticmethod
    def deleteFolders(idk, delete, notice):
        connect = connection.connection()
        cursor = connect.cursor()
        delete = 'DO $$ BEGIN if ' \
                 'exists (select foldID from folders
where foldID = {}) then ' \
                 ' delete from folders_notifications
where folders_ID in (select foldID from folders where
id_mainfolder = {});' \
                 'delete from folders where foldID= {};'
\
                 'raise notice {};' \
                 'else raise notice {};' \
                 'end if;' \
                 'end $$;'.format(idk, idk, idk, delete,
notice)
        cursor.execute(delete)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)

    @staticmethod
    def deleteUsers(idk, delete, notice):
```

```
        connect = connection.connection()
        cursor = connect.cursor()
        delete = 'DO $$ BEGIN if ' \
                 'exists (select userID from users where
userID = {}) then ' \
                 'delete from notifications_users where
notifications_ID in (select povID from notifications '
\
                 'where id_userfkey in (select userID
from users where userID = {}));' \
                 'delete from notifications_users where
users_ID in (select povID from notifications ' \
                 'where id_userfkey = {});' \
                 'delete from notifications where
id_userfkey = {};' \
                 'delete from users where userID = {};'
\
                 'raise notice {};' \
                 'else raise notice {};' \
                 'end if;' \
                 'end $$;'.format(idk, idk, idk, idk,
idk, delete, notice)
        cursor.execute(delete)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)
```

```
    @staticmethod
    def deleteNotifications(idk, delete, notice):
        connect = connection.connection()
        cursor = connect.cursor()
        delete = 'DO $$ BEGIN if exists (select povID
from notifications where povID = {}) then ' \
                 'delete from notifications_users where
notifications_ID = {};' \
                 'delete from folders_notifications
where notifications_ID = {};' \
                 'delete from notifications where povID=
{};' \
                 'raise notice {};' \
                 'else raise notice {};' \
                 'end if;' \
                 'end $$;'.format(idk, idk, idk, idk,
delete, notice)
        cursor.execute(delete)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)

    @staticmethod
    def UpdateEmail(idk, set2, set3, updated, notice):
        connect = connection.connection()
        cursor = connect.cursor()
```

```
        update = 'DO $$ BEGIN IF EXISTS (select gmailID
from email where gmailID = {}) THEN ' \
                 'update email set creator = {}, name =
{} where gmailID = {}; ' \
                 'RAISE NOTICE {};' \
                 ' ELSE RAISE NOTICE {};' \
                 'END IF; ' \
                 'END $$;'.format(idk, set2, set3, idk,
updated, notice)
        cursor.execute(update)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)


    @staticmethod
    def UpdateFolders(idk, set1, set2, set3, updated,
notice):
        connect = connection.connection()
        cursor = connect.cursor()
        update = 'DO $$ BEGIN IF EXISTS (select foldID
from folders where foldID = {})' \
                 ' THEN ' \
                 'update folders set Size = {}, Colour =
{}, nameof = {} where foldID = {}; ' \
                 'RAISE NOTICE {};' \
                 ' ELSE RAISE NOTICE {};' \
```

```
                'END IF; ' \
                'END $$;'.format(idk, set1, set2, set3,
idk, updated, notice)
        cursor.execute(update)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)

    @staticmethod
    def UpdateUsers(idk, adress, place, updated,
notice):
        connect = connection.connection()
        cursor = connect.cursor()
        update = 'DO $$ BEGIN IF EXISTS (select userID
from users where userID = {}) ' \
                ' THEN ' \
                'update users set adress = {}, place =
{} where userID = {}; ' \
                'RAISE NOTICE {};' \
                ' ELSE RAISE NOTICE {};' \
                'END IF; ' \
                'END $$;'.format(idk, adress, place,
idk, updated, notice)
        cursor.execute(update)
        connect.commit()
        print(connect.notices)
        cursor.close()
```

```
        connection.connectionlost(connect)


    @staticmethod
    def UpdateNotifications(idk, text, addfiles, title,
sender, updated, notice):
        connect = connection.connection()
        cursor = connect.cursor()
        update = 'DO $$ BEGIN IF EXISTS (select povID
from notifications where povID = {}) ' \
                 ' THEN ' \
                 'update notifications set text = {},
addfiles = {}, title = {}, id_sender = {} where povID =
{}; ' \
                 'RAISE NOTICE {};' \
                 ' ELSE RAISE NOTICE {};' \
                 'END IF; ' \
                 'END $$;'.format(idk, text, addfiles,
title, sender, idk, updated, notice)
        cursor.execute(update)
        connect.commit()
        print(connect.notices)
        cursor.close()
        connection.connectionlost(connect)


    @staticmethod
    def selectionone(len, creator):
        connect = connection.connection()
```

```
        cursor = connect.cursor()
        select = """select size, colour, name, creator
from (select c.size, c.colour, p.name,
                                        p.creator from
                                            folders c
left join email p
                                             on
p.gmailID = c.id_mainfolder where length(p.name) >= {}
and p.creator LIKE '{}'
                                              group by
c.size,
                                             c.colour,
p.name, p.creator) as foo""".format(len, creator)
        print("SQL query => ", select)
        beg = int(time.time() * 1000)
        cursor.execute(select)
        end = int(time.time() * 1000) - beg
        datas = cursor.fetchall()
        print('Time of request {} ms'.format(end))
        print('Selected')
        cursor.close()
        connection.connectionlost(connect)
        return datas

    @staticmethod
    def selectiontwo(value, adress):
        connect = connection.connection()
        cursor = connect.cursor()
```

```
        select = """select adress, addfiles, text from
(select p.text, p.addfiles, c.adress from

notifications p right join users c on p.id_userfkey =
c.userID

where p.addfiles < {} and c.adress LIKE '{}' group by

c.adress, p.addfiles, p.text) as foo
                            """.format(value, adress)
        print("SQL query => ", select)
        beg = int(time.time() * 1000)
        cursor.execute(select)
        end = int(time.time() * 1000) - beg
        datas = cursor.fetchall()
        print('Time of request {} ms'.format(end))
        print('Selected')
        cursor.close()
        connection.connectionlost(connect)
        return datas

    @staticmethod
    def selectionthree(title):
        connect = connection.connection()
        cursor = connect.cursor()
        select = """select name, nameof, size, colour,
name from (select c.name, p.nameof, p.size, p.colour
from
```

```
folders p left join email c on
c.gmailID=p.id_mainfolder

where c.name LIKE '{}' group by c.name, p.nameof,
p.size, p.colour) as foo
                        """.format(title)
        print("SQL query => ", select)
        beg = int(time.time() * 1000)
        cursor.execute(select)
        end = int(time.time() * 1000) - beg
        datas = cursor.fetchall()
        print('Time of request {} ms'.format(end))
        print('Selected')
        cursor.close()
        connection.connectionlost(connect)
        return datas



    @staticmethod
    def randomik(table, kolvo):
            connect = connection.connection()
            cursor = connect.cursor()
            check = True
            while check:
                if table == 1:
```

```
                    insert = "INSERT INTO email
(Creator, Name) select chr(trunc(65 +
random()*26)::int)||chr(trunc(65 + r" \
                                "andom()*26)::int), " \
                                "chr(trunc(65 +
random()*26)::int)||chr(trunc(65 + random()*26)::int) "
\
                                "from
generate_series(1,{})".format(kolvo)
                    cursor.execute(insert)
                    check = False
                elif table == 2:
                    res = 0
                    while (True):
                        insert = "INSERT INTO
folders(Size, Colour, Nameof, id_mainfolder) select
random() * 256," \
                                "random() * 256," \
                                "chr(trunc(65 +
random()*26)::int)||chr(trunc(65 + random()*26)::int),"
\
                                "(select gmailID from email
order by random() limit 1)"\
                                "from generate_series(1,1)"
                        cursor.execute(insert)
                        res = res + 1
                        if(res == kolvo):
                            break
```

```
                    check = False
                elif table == 3:
                    res = 0
                    while (res != kolvo):
                        insert = "INSERT INTO users
(adress, place, id_mail) select " \
                                 "chr(trunc(65 +
random()*25)::int)||chr(trunc(65 + " \
                                 "random()*25)::int), "
\
                                 "chr(trunc(65 +
random()*25)::int)||chr(trunc(65 + random()*25)::int),"
\
                                 "(select gmailID from
email order by random() limit 1) " \
                                 "from
generate_series(1,1)"
                        cursor.execute(insert)
                        res = res + 1

                    check = False
                elif table == 4:
                    res = 0
                    while (res != kolvo):
                        insert = "INSERT INTO
notifications (text, addfiles, title, id_userfkey,
id_sender) select " \
```

```
                                "chr(trunc(65 +
random()*26)::int)||chr(trunc(65 + r" \
                                "andom()*26)::int),
random() * 256," \
                                "chr(trunc(65 +
random()*26)::int)||chr(trunc(65 + random()*26)::int),"
\
                                "(select userID from users
order by random() limit 1)," \
                                "random() * 256 " \
                                "from generate_series(1,1)"
                            cursor.execute(insert)
                            res = res + 1
                        check = False
                        check = False
                print(Tables[table])
                print("SQL query => ", insert)
                connect.commit()
                print('Inserted randomly')
                cursor.close()
                connection.connectionlost(connect)
```

existingtable() – перевірка на існування таблиці.

Outputonetable() – вивід вибраної таблиці.

insertbyusertoEmail() – додавання рядків у таблицю Email.

insertbyusertoFolders()– додавання рядків у таблицю Folders.

insertbyusertoUsers()– додавання рядків у таблицю Users.

insertbyusertoNotifications()– додавання рядків у таблицю Notifications.

deleteEmail() – видалення рядків з Email.

deleteFolders()– видалення рядків з Folders.

deleteUsers()– видалення рядків з Users.

deleteNotifications()– видалення рядків з Notifications.

UpdateEmail()– редагування рядка в Email.

UpdateFolders()– редагування рядка в Folders.

UpdateUsers()– редагування рядка в Users.

UpdateNotifications()– редагування рядка в Notifications.

selectionone()– пошуковий запис 1.

selectiontwo()– пошуковий запис 2.

selectionthree()– пошуковий запис 3.

randomik() – заповнення таблиць випадковими даними.