



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

Іус І. О.

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).

Мова програмування: Python. Використані

бібліотеки: psycorg2, time

“Сутність-зв’язок”

Сутності:

- Учні
- Оцінки
- Предмети
- Вчителі

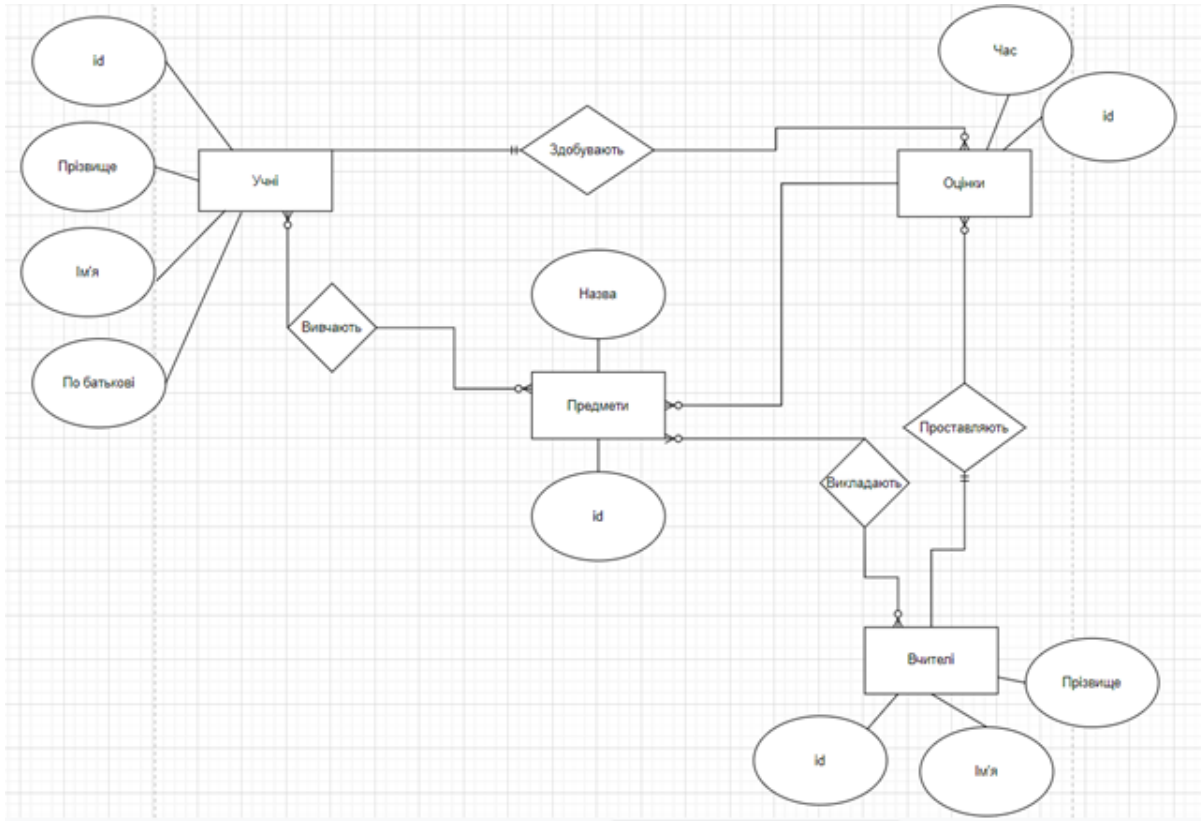
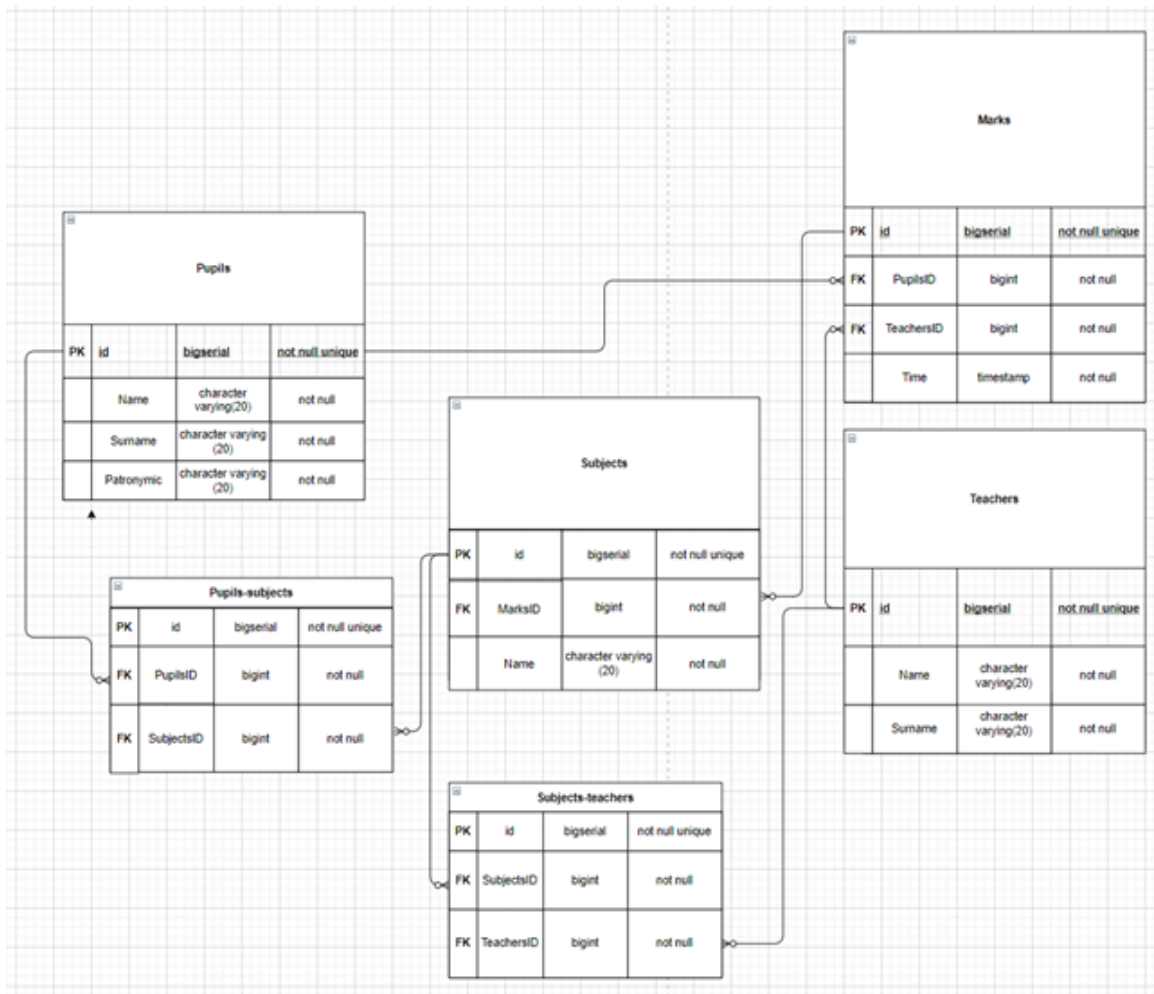


Схема бази даних у графічному вигляді:



Опис бази даних:

У даному випадку маємо 4 сутності: оцінки, вчителі, учні, предмети.

Перша сутність “Предмети ” потрібна для ведення обліку предметів, які вивчаються у школі. У собі містить назву предмета та його ID.

Друга сутність – “Вчителі”. Використовується для ведення обліку вчителів навчального закладу шляхом ідентифікації. Також містить інформацію про ім'я та прізвище вчителя а також ID.

Третя сутність називається “Учні ”. Використовується для ведення обліку усіх учнів, що навчаються у закладі. У собі має такі характеристики, як: ID, ім'я, прізвище та ім'я по-батькові учня.

Четверта сутність – “Оцінки”. Необхідна для ведення обліку оцінок, які отримали учні упродовж навчання. Має такі характеристики як час проставлення 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 Subjects та дочірньої Marks

Запис у Subjects:

```
Your choice is: 3

1          => Subjects
2          => Teachers
3          => Pupils
4          => Marks
5          => PupilsSubjects
6          => TeachersSubjects

Choose your table: 1
name = TestInsert
marksID = 1
['NOTICE: Record added\n']
1 => Continue insertion, 2 => Stop insertion => 1
```

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

```
6          => TeachersSubjects

Choose your table: 1
name = TestWrong
marksID = 6666
['NOTICE: Mark with ID=6666 is not exists\n']
1 => Continue insertion, 2 => Stop insertion => |
```

Таблиця Subjects після Insert

Choose your table: 1

SQL query => select * from public."Subjects"

id	name	marksID
2	LQ	1
3	MZ	1
5	testSubject	2
1	Updated	1
6	test	2
7	TestInsert	1

Continue to work with db => 1, stop => 2. Your choice =>

Лістинг для Insert:

```
@staticmethod
def insertSubject(f,s,added,notice):
    connect = Database.connect()
    cursor = connect.cursor()
    insert = 'DO $$ BEGIN if (1=1) and exists (select id from public."Marks" where id = {}) '
    'then INSERT INTO public."Subjects"(name, marksID) VALUES ({} ,{}); ' \
    'raise notice {}; else raise notice {}; ' \
    'end if; end $$;'.format(s, f, s, added, notice)
    cursor.execute(insert)
    connect.commit()
    print(connect.notices)
    cursor.close()
    Database.close(connect)

@staticmethod
def insertTeacher(f, s, added, notice):
    connect = Database.connect()
    cursor = connect.cursor()
    insert = 'DO $$ BEGIN IF (1=1) THEN ' \
    'INSERT INTO public."Teachers"(name, surname) values (\'{}\', \'{}\'); ' \
    'RAISE NOTICE {};' \
    ' ELSE RAISE NOTICE {};' \
    'END IF; ' \
```



```

        'END $$;'.format(f, s, added, notice)
    cursor.execute(insert)
    connect.commit()
    print(connect.notices)
    cursor.close()
    Database.close(connect)

@staticmethod
def insertPupil(f,s,t,added,notice):
    connect = Database.connect()
    cursor = connect.cursor()
    insert = 'DO $$ BEGIN IF (1=1) THEN ' \
        'INSERT INTO public."Pupils"("Name", "Patronymic", "Surname") values ({} , {},
    {}); ' \
        'RAISE NOTICE {};' \
        ' ELSE RAISE NOTICE {};' \
        'END IF; ' \
        'END $$;'.format(f,s,t, added, notice)
    cursor.execute(insert)
    connect.commit()
    print(connect.notices)
    cursor.close()
    Database.close(connect)

@staticmethod
def insertMark(f,s,t,added,notice):
    connect = Database.connect()
    cursor = connect.cursor()
    insert = 'DO $$ BEGIN IF EXISTS (select "Id" from public."Pupils" where "Id" = {}) and
exists (select id from public."Teachers" where id = {}) THEN ' \
        'INSERT INTO public."Marks"(time, pupilsID, teachersID) values (\'{}\', {}, {});
    ' \
        'RAISE NOTICE {};' \
        ' ELSE RAISE NOTICE {};' \
        'END IF; ' \
        'END $$;'.format(f, s, t, f, s, added, notice)
    cursor.execute(insert)
    connect.commit()
    print(connect.notices)
    cursor.close()
    Database.close(connect)

```

Update

У нашому випадку редагування ключів є неможливим

1 => Continue update, 2 => Stop update => 1

1	=> Subjects
2	=> Teachers
3	=> Pupils
4	=> Marks
5	=> PupilsSubjects
6	=> TeachersSubjects

Choose your table:1

Row to update where id = 5

New name = *InsertedAndUpdated*

['NOTICE: updated\n']

SQL query => select * from public."Subjects"

id	name	marksID
2	LQ	1
3	MZ	1
1	Updated	1
6	test	2
7	TestInsert	1
5	InsertedAndUpdated2	

При спробі редагувати рядок, якого не існує:

```
1          => Subjects
2          => Teachers
3          => Pupils
4          => Marks
5          => PupilsSubjects
6          => TeachersSubjects
```

Choose your table: **1**

Row to update where id = **7777**

New name = **fake**

['NOTICE: id = 7777 is not present in table.\n']

1 => Continue update, 2 => Stop update => |

Лістинг для Update

@staticmethod

def UpdateSubject(idk, name, updated, notice):

connect = Database.connect()

cursor = connect.cursor()

update = 'DO \$\$ BEGIN IF EXISTS (select id from public."Subjects" where id = {}) THEN '\

'update public."Subjects" set name = {} where id = {};' \

'RAISE NOTICE {};' \

'ELSE RAISE NOTICE {};' \

'END IF; '\

'END \$\$;'.format(idk, name, idk, updated, notice)

cursor.execute(update)

connect.commit()

print(connect.notices)

cursor.close()

Database.close(connect)

@staticmethod

```

def UpdateTeachers(idk, set1, set2, updated, notice):
    connect = Database.connect()
    cursor = connect.cursor()
    update = 'DO $$ BEGIN IF EXISTS (select id from public."Teachers" where id = {})' \
        ' THEN ' \
        'update public."Teachers" set Name = {}, Surname = {} where id = {};' \
        'RAISE NOTICE {};' \
        ' ELSE RAISE NOTICE {};' \
        'END IF;' \
        'END $';'.format(idk, set1, set2, idk, updated, notice)
    cursor.execute(update)
    connect.commit()
    print(connect.notices)
    cursor.close()
    Database.close(connect)

```

@staticmethod

```

def UpdatePupils(idk, name, patronymic, surname, updated, notice):
    connect = Database.connect()
    cursor = connect.cursor()
    update = 'DO $$ BEGIN IF EXISTS (select "Id" from public."Pupils" where "Id" = {})' \
        ' THEN ' \
        'update public."Pupils" set "Name" = {}, "Patronymic" = {}, "Surname" = {} where "Id" = {};' \
        'RAISE NOTICE {};' \
        ' ELSE RAISE NOTICE {};' \
        'END IF;' \
        'END $';'.format(idk, name, patronymic, surname, idk, updated, notice)
    cursor.execute(update)
    connect.commit()
    print(connect.notices)
    cursor.close()
    Database.close(connect)

```

@staticmethod

```

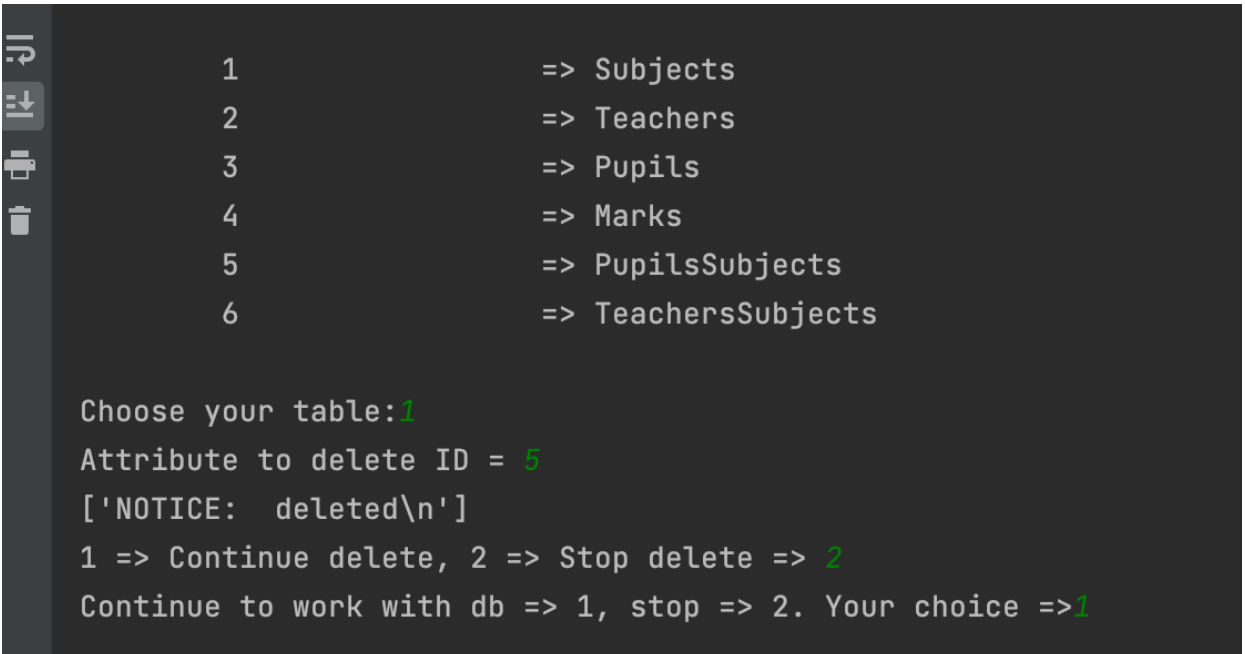
def UpdateMarks(idk, date, updated, notice):
    connect = Database.connect()
    cursor = connect.cursor()
    update = 'DO $$ BEGIN IF EXISTS (select id from public."Marks" where id = {})' \
        ' THEN ' \
        'update public."Marks" set time = \'{}\'' where id = {};' \
        'RAISE NOTICE {};' \
        ' ELSE RAISE NOTICE {};' \
        'END IF;' \
        'END $';'.format(idk, date, idk, updated, notice)

```

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

Delete

На прикладі таблиці Subjects (початковий стан таблиці як після Update)



```
1 => Subjects
2 => Teachers
3 => Pupils
4 => Marks
5 => PupilsSubjects
6 => TeachersSubjects

Choose your table:1
Attribute to delete ID = 5
['NOTICE:  deleted\n']
1 => Continue delete, 2 => Stop delete => 2
Continue to work with db => 1, stop => 2. Your choice =>1
```

Choose your table:1

SQL query => select * from public."Subjects"

id	name	marksID
2	LQ	1
3	MZ	1
1	Updated	1
6	test	2
7	TestInsert	1

При спробі видалення неіснуючого рядка:

1	=> Subjects
2	=> Teachers
3	=> Pupils
4	=> Marks
5	=> PupilsSubjects
6	=> TeachersSubjects

Choose your table:1

Attribute to delete ID = 23453245

['NOTICE: Entered ID is wrong\n']

1 => Continue delete, 2 => Stop delete =>

Лістинг Delete:

@staticmethod

```
def deleteSubject(idk, delete, notice):
```

```
    connect = Database.connect()
```

```
    cursor = connect.cursor()
```

```
    delete = 'DO $$ BEGIN IF EXISTS (select id from public."Subjects" where id = {}) then ' \
```

```
        'delete from public."Marks" where id in (select marksID from public."Subjects" where id = {});' \
```

```
        'delete from public."Subjects" where id = {};' \
```

```
        'raise notice {};' \
```

```
        'else raise notice {};' \
```

```
        'end if;' \
```

```
        'end $$';'.format(idk, idk, idk, delete, notice)
```

```
    cursor.execute(delete)
```

```
    connect.commit()
```

```
    print(connect.notices)
```

```
    cursor.close()
```

```
    Database.close(connect)
```

@staticmethod

```
def deleteTeachers(idk, delete, notice):
```

```
    connect = Database.connect()
```

```
    cursor = connect.cursor()
```

```
    delete = 'DO $$ BEGIN if ' \
```

```
        'exists (select id from public."Teachers" where id = {}) then ' \
```

```
        'delete from public."TeachersSubjects" where teachersID = {};' \
```

```
        'delete from public."Marks" where teachersID = {};' \
```

```
        'delete from public."Teachers" where id = {};' \
```

```
        '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()
```

```
Database.close(connect)
```

```
@staticmethod
```

```
def deletePupils(idk, delete, notice):
```

```
    connect = Database.connect()
```

```
    cursor = connect.cursor()
```

```
    delete = 'DO $$ BEGIN if ' \
```

```
        'exists (select "Id" from public."Pupils" where "Id" = { }) then ' \
```

```
        'delete from public."PupilsSubjects" where pupilsID = { };' \
```

```
        'delete from public."Marks" where pupilsID = { };' \
```

```
        'delete from public."Pupils" where "Id" = { };' \
```

```
        '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()
```

```
    Database.close(connect)
```

```
@staticmethod
```

```
def deleteMarks(idk, delete, notice):
```

```
    connect = Database.connect()
```

```
    cursor = connect.cursor()
```

```
    delete = 'DO $$ BEGIN if exists (select id from public."Marks" where id = { }) then ' \
```

```
        'delete from public."Marks" where id = { };' \
```

```
        'raise notice { };' \
```

```
        'else raise notice { };' \
```

```
        'end if;' \
```

```
        'end $$;'.format(idk, idk, delete, notice)
```

```
    cursor.execute(delete)
```

```
    connect.commit()
```

```
    print(connect.notices)
```



```
cursor.close()
```

```
Database.close(connect)
```

Завдання №2

Передбачити автоматичне пакетне генерування “рандомізованих” даних:

На прикладі таблиці Pupils:

```
Choose your table: 3
```

```
SQL query => select * from public."Pupils"
```

```
*****
```

id	Name	Patronymic	Surname
1	KQ	WG	CV
4	VL	YT	HB
5	KX	AC	RV
6	AA	RU	EC
7	EI	YG	CJ
8	VK	XH	NR
9	BW	RG	KA
10	Pupok	Ivanovich	Pupil
2	Mikhail	Petrovich	Ivanov

```
*****
```

```
Choose your table: 1
```

```
How much datas do you want to add => 5
```

```
Pupils
```

```
SQL query => INSERT INTO public."Pupils"("Name", "Surname", "Patronymic") select chr(trunc(65 + random()*26)::int)||chr(trunc(65 + random()*26)::int), chr(trunc(65 + random()*26)::int)||chr(trunc(65 + random()*26)::int), chr(trunc(65 + random()*26)::int)||chr(trunc(65 + random()*26)::int)
Inserted randomly
```

```
Choose your table: 3
SQL query => select * from public."Pupils"

*****

id      Name      Patronymic      Surname
1       KQ        WG              CV
4       VL        YT              HB
5       KX        AC              RV
6       AA        RU              EC
7       EI        YG              CJ
8       VK        XH              NR
9       BW        RG              KA
10      Pupok    Ivanovich      Pupil
2       Mikhail Petrovich      Ivanov
11      KA        NY              OZ
12      UG        LI              VT
13      PU        WW              TC
14      VY        OX              BA
15      GQ        QR              RW
```

Лістинг:

```
def randomik(table, kolvo):
    connect = Database.connect()
    cursor = connect.cursor()
    check = True
    while check:
        if table == 1:
            res = 0
            while (True):
                insert = "INSERT INTO public.\"Subjects\"(Name, MarksID) select chr(trunc(65 +
random()*26)::int)||chr(trunc(65 + r" \
                "andom()*26)::int), " \
                "(select id from public.\"Marks\" order by random() limit 1)" \
                " from generate_series(1,{})".format(kolvo)
                cursor.execute(insert)
                res = res + 1
                if(res == int(kolvo)):
                    break
            check = False
        elif table == 2:
            res = 0
            while (True):
                insert = "INSERT INTO public.\"Teachers\"(Name, Surname) 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)
                res = res + 1
                if(res == int(kolvo)):
                    break
            check = False
        elif table == 3:
            res = 0
            while (True):
                insert = "INSERT INTO public.\"Pupils\"(\"Name\", \"Surname\", \"Patronymic\") 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), " \
    "chr(trunc(65 + random()*26)::int)||chr(trunc(65 + random()*26)::int)" \
    "from generate_series(1,{ })".format(kolvo)
cursor.execute(insert)
res = res + 1
if(res == int(kolvo)):
    break
check = False
elif table == 4:
    res = 0
    while (True):
        insert = "INSERT INTO public.\"Marks\"(Time, PupilsID, TeachersID) values(" \
            "(select NOW() + (random() * (NOW()+90 days' - NOW())) + '{' } days')," \
            "(select \"Id\" from public.\"Pupils\" order by random() limit 1)," \
            "(select id from public.\"Teachers\" order by random() limit 1))".format(kolvo)
        cursor.execute(insert)
        res = res + 1
        if (res == int(kolvo)):
            break
        check = False
    elif table == 5:
        res = 0
        while (True):
            insert = "INSERT INTO public.\"PupilsSubjects\"(PupilsID, SubjectsID) values(" \
                "(select \"Id\" from public.\"Pupils\" order by random() limit 1)," \
                "(select id from public.\"Subjects\" order by random() limit 1))".format(kolvo)
            cursor.execute(insert)
            res = res + 1
            if (res == int(kolvo)):
                break
            check = False
    elif table == 6:
        res = 0
        while (True):
            insert = "INSERT INTO public.\"TeachersSubjects\"(TeachersID, SubjectsID) values(" \
                "(select id from public.\"Teachers\" order by random() limit 1)," \
                "(select id from public.\"Subjects\" order by random() limit 1))".format(kolvo)
            cursor.execute(insert)
            res = res + 1
            if (res == int(kolvo)):
                break
            check = False
print(Tables[table])
print("SQL query => ", insert)
connect.commit()
print('Inserted randomly')
cursor.close()
Database.close(connect)

```

Завдання №3

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

```
-----
1 => Show Teachers and related to them Pupils where mark datetime is greater then X and Teachers surname like Y
-----
2 => Show teachers that teach subjects with name like X and post a marks starting from date Y
-----
3 => Show name of ongoing subjects (have teachers, students and marks)
-----
Your choice is 1
Enter time criteria (Y-m-d H:M:S) = 2000-02-01 10:10:10
Enter teachers surname criteria = %
SQL query =>
    select
        public."Teachers".name, public."Teachers".surname,
        public."Pupils"."Name", public."Pupils"."Surname"

    from public."Teachers"
    right join public."Marks" on public."Marks".teachersid = public."Teachers".id
    left join public."Pupils" on public."Marks".pupilsid = public."Pupils"."Id"

    where public."Marks".time > '2000-02-01 10:10:10'
        and public."Teachers".surname like '%'

Time of request 69 ms
Selected
*****

teacher name      teacher surname    pupil name          pupil surname
GX                TC                 KQ                  CV
VX                OA                 Mikhail             Ivanov
*****
```

```

2 => Show teachers that teach subjects with name like X and post a marks starting from date Y
-----
3 => Show name of ongoing subjects (have teachers, students and marks)
-----
Your choice is 2
Enter subject name criteria = %
Enter time criteria (Y-m-d H:M:S) = 2010-01-01 12:24:23
SQL query =>
    select
        public."Teachers".name, public."Teachers".surname,
        public."Subjects".name,
        public."Marks".time

    from public."Subjects"
    join public."Marks" on public."Marks".id = public."Subjects".marksid
    join public."Teachers" on public."Marks".teachersid = public."Teachers".id

    where public."Subjects".name like '%'
        and public."Marks".time > '2010-01-01 12:24:23'

Time of request 75 ms
Selected
*****

teacher name      teacher surname    subjects name        marks time
GX                TC                 LQ                   2018-10-10 12:11:00
GX                TC                 MZ                   2018-10-10 12:11:00
GX                TC                 Updated              2018-10-10 12:11:00
GX                TC                 TestInsert           2018-10-10 12:11:00
*****
```

```

-----
2 => Show teachers that teach subjects with name like X and post a marks starting from date Y
-----
3 => Show name of ongoing subjects (have teachers, students and marks)
-----
Your choice is 3
Enter teachers surname criteria = %
Enter pupils surname criteria = %
SQL query =>
    select public."Subjects".name from public."Subjects"

    join public."Marks" on public."Marks".id = public."Subjects".marksid

    join public."TeachersSubjects"
        on public."TeachersSubjects".subjectsid = public."Subjects".id
        and public."TeachersSubjects".teachersid = public."Marks".teachersid

    join public."PupilsSubjects"
        on public."PupilsSubjects".subjectsid = public."Subjects".id
        and public."PupilsSubjects".pupilsid = public."Marks".pupilsid

    join public."Pupils" on "PupilsSubjects".pupilsid = public."Pupils"."Id"
    join public."Teachers" on public."TeachersSubjects".teachersid = public."Teachers".id

    where public."Teachers".surname like '%'
        and public."Pupils"."Surname" like '%'

    group by public."Subjects".name

```

Time of request 86 ms

Time of request 86 ms

Selected

Name

LQ

Updated

Лістинг:

```
@staticmethod
def selectionone(timestamp, teacherSurname):
    connect = Database.connect()
    cursor = connect.cursor()
    select = """
select
    public."Teachers".name, public."Teachers".surname,
    public."Pupils"."Name", public."Pupils"."Surname"

from public."Teachers"
right join public."Marks" on public."Marks".teachersid = public."Teachers".id
left join public."Pupils" on public."Marks".pupilsid = public."Pupils"."Id"

where public."Marks".time > '{ }'
    and public."Teachers".surname like '{ }'
""".format(timestamp, teacherSurname)
    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()
    Database.close(connect)
    return datas

@staticmethod
def selectiontwo(subj, markTime):
    connect = Database.connect()
    cursor = connect.cursor()
    select = """
select
    public."Teachers".name, public."Teachers".surname,
    public."Subjects".name,
    public."Marks".time

from public."Subjects"
join public."Marks" on public."Marks".id = public."Subjects".marksid
join public."Teachers" on public."Marks".teachersid = public."Teachers".id

where public."Subjects".name like '{ }'
    and public."Marks".time > '{ }'
""".format(subj, markTime)
    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()
    Database.close(connect)
    return datas

@staticmethod
```

```

def selectionthree(teachersS, pupilsS):
    connect = Database.connect()
    cursor = connect.cursor()
    select = """
    select public."Subjects".name from public."Subjects"

        join public."Marks" on public."Marks".id = public."Subjects".marksid

        join public."TeachersSubjects"
            on public."TeachersSubjects".subjectsid = public."Subjects".id
            and public."TeachersSubjects".teachersid = public."Marks".teachersid

        join public."PupilsSubjects"
            on public."PupilsSubjects".subjectsid = public."Subjects".id
            and public."PupilsSubjects".pupilsid = public."Marks".pupilsid

        join public."Pupils" on "PupilsSubjects".pupilsid = public."Pupils"."Id"
        join public."Teachers" on public."TeachersSubjects".teachersid = public."Teachers".id

        where public."Teachers".surname like '{}'
            and public."Pupils"."Surname" like '{}'

        group by public."Subjects".name
    """
    .format(teachersS, pupilsS)
    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()
    Database.close(connect)
    return datas

```

Методи моделі:

existingtable() – перевірка на існування таблиці. Outputonetable()

– вивід вибраної таблиці.

Insert<tablename>() – додавання рядків у таблицях.

Update<tablename>() – оновлення рядків у таблицях.

Delete<tablename>() – видалення рядків у таблицях.

selectionone() – пошуковий запис 1.

selectiontwo() – пошуковий запис 2.

selectionthree() – пошуковий запис 3.

randomik() – заповнення таблиць випадковими даними.

