



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:** «Створення додатку бази даних, орієнтованого на взаємодію з  
СУБД PostgreSQL»

Виконала: студентка III курсу

ФПМ групи KB-84

Величко М. М.

Перевірив:

Київ – 2020

## Завдання:

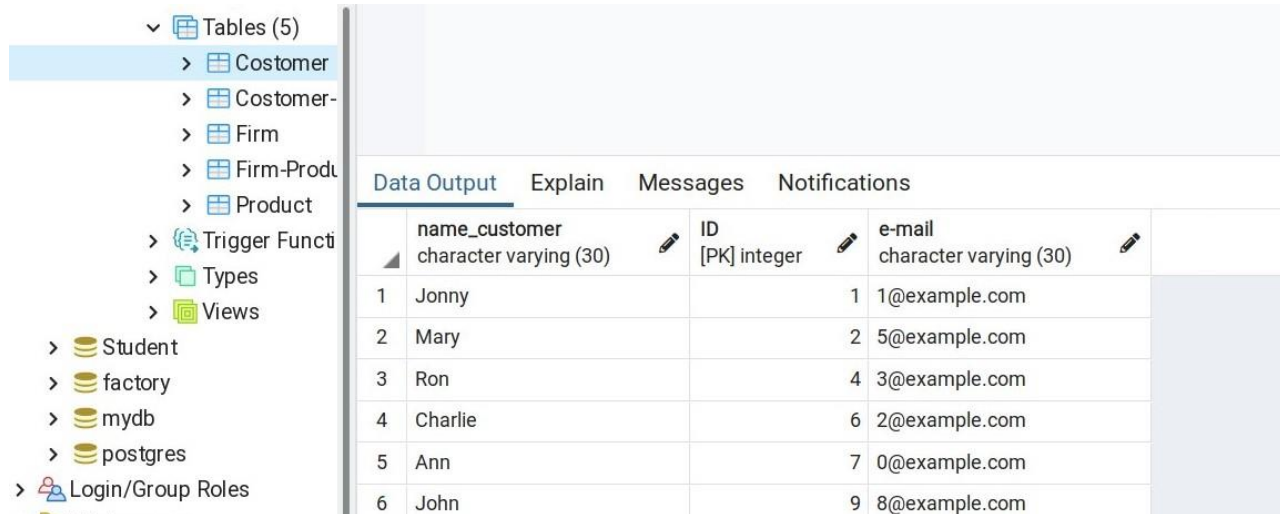
1. Реалізувати функції внесення, редагування та видалення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та 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).

## Посилання на репозиторій:

[https://github.com/maryna-velychko/My\\_proj](https://github.com/maryna-velychko/My_proj)

## Структура бази даних з лабораторної №1:

Costomer:



The screenshot shows a database management interface. On the left, a tree view lists database objects: Tables (5), Costomer, Costomer-, Firm, Firm-Prod, Product, Trigger Functi, Types, Views, Student, factory, mydb, postgres, and Login/Group Roles. The 'Costomer' table is selected. The main area displays the table's structure and data. The structure shows three columns: 'name\_customer' (character varying (30)), 'ID' ([PK] integer), and 'e-mail' (character varying (30)). The data output shows six rows of customer information.

	name_customer character varying (30)	ID [PK] integer	e-mail character varying (30)
1	Jonny		1@example.com
2	Mary		5@example.com
3	Ron		3@example.com
4	Charlie		6@example.com
5	Ann		7@example.com
6	John		8@example.com

Firm:



Columns

Constraints

Indexes

RLS Policies

Rules

Triggers

Firm

Firm-Products

Product

Trigger Functions

Types

Views

Data Output

Explain

Messages

Notifications

	<div>article</div> <div>[PK] integer</div>	<div>ID</div> <div>[PK] integer</div>
1	12	1
2	23	2
3	34	4
4	45	6
5	56	7
6	67	9

### Вимоги до пункту №1 деталізованого завдання:

Ілюстрації обробки виняткових ситуацій (помилки) при введенні/вилучення даних:

Помилка при додаванні даних до таблиці.

```

pythonProject1 - Model.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help

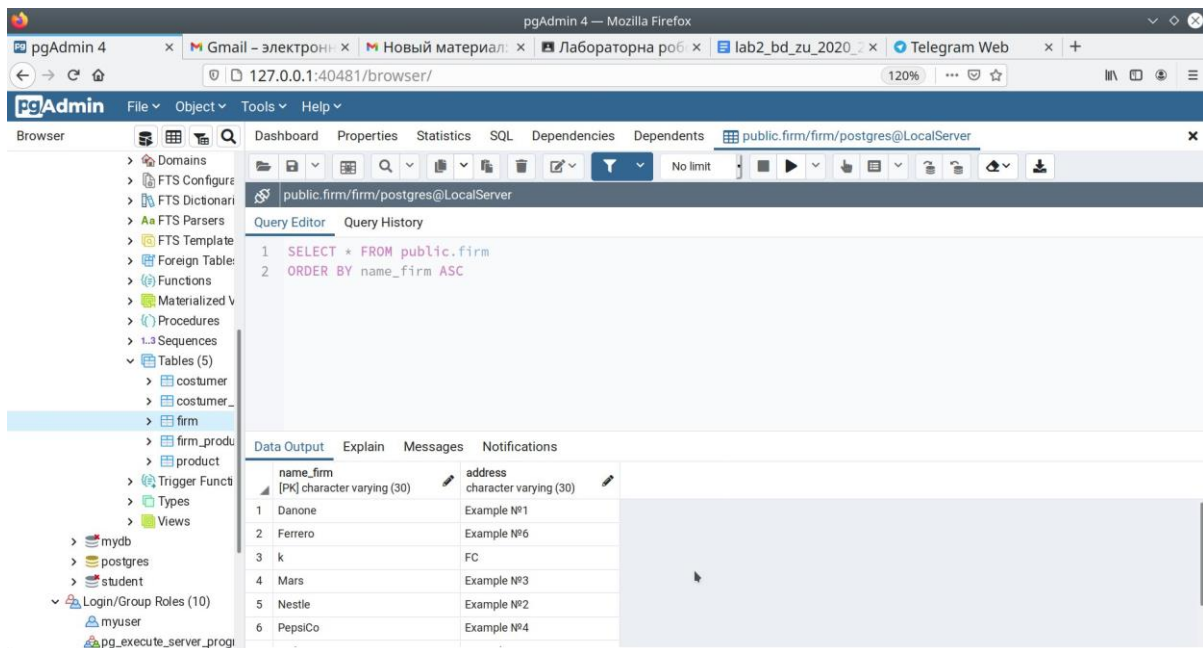
pythonProject1 Model.py
Controller.py View.py Model.py
Run: Controller
Enter:
1 - create
2 - delete
3 - edit
4 - get
5 - return to main menu

1
Choose table:
Customer - press 1
Firm - press 2
Product - press 3
Customer_Product - press 4
Firm_Product - press 5

2
Enter firm name
Danone
Enter firm address

Error, ENTER WRONG DATA
  
```

Атрибут name\_firm є головним ключем таблиці costumer, значення Danone вже існує)



Для інших таблиц також передбачені подібні помилки:

Для редагування:

```

35 def edit(cursor, table, parameter_1, parameter_2, parameter_3):
36     try:
37         if table == 1:
38             cursor.execute("UPDATE costumer SET name_costumer = %, email = %s WHERE id = %s", (parameter_1, parameter_2, [parameter_3])
39         elif table == 2:
40             cursor.execute("UPDATE firm SET name_firm = %, address = %s WHERE name_firm = %s", (parameter_1, parameter_2, parameter_3,))
41         elif table == 3:
42             cursor.execute("UPDATE product SET name_product = %, price = %s WHERE article = %s", (parameter_1, float(parameter_2), parameter_3))
43         elif table == 4:
44             cursor.execute("UPDATE costumer_product SET article = %, id = %s WHERE article = %s", (parameter_1, parameter_2, parameter_3))
45         elif table == 5:
46             cursor.execute("UPDATE firm_product SET name_firm = %, article = %s WHERE article = %s", (parameter_1, parameter_2, parameter_3))
47     except errors.ForeignKeyViolation:
48         print("Error, ENTER WRONG DATA\n")

```

Для додавання:

```

6 def create(cursor, table, parameter_1, parameter_2):
7     try:
8         if table == 1:
9             cursor.execute("INSERT INTO costumer (name_costumer, email) VALUES (%s, %s)", (parameter_1, parameter_2))
10        elif table == 2:
11            cursor.execute("INSERT INTO firm (name_firm, address) VALUES (%s, %s)", (parameter_1, parameter_2))
12        elif table == 3:
13            cursor.execute("INSERT INTO product (name_product, price) VALUES (%s, %s)", (parameter_1, float(parameter_2)))
14        elif table == 4:
15            cursor.execute("INSERT INTO costumer_product (id, article) VALUES (%s, %s)", ([parameter_1], [parameter_2]))
16        elif table == 5:
17            cursor.execute("INSERT INTO firm_product (name_firm, article) VALUES (%s, %s)", (parameter_1, [parameter_2]))
18    except errors.ForeignKeyViolation:
19        print("Error, ENTER WRONG DATA\n")
20

```

Ілюстрації валідації даних при уведенні користувачем:

```

Run: Controller
5 - return to main menu

1
Choose table:
Customer - press 1
Firm - press 2
Product - press 3
Customer_Product - press 4
Firm_Product - press 5
2
Enter costumer name
Polly
Enter costumer e-mail
p@example
Enter:
1 - create
2 - delete
3 - edit
4 - get
5 - return to main menu

```

```

Controller.py x View.py x Model.py
Run: Controller
5 - return to main menu
4
Enter name table:
costumer
firm
product
costumer_product
firm_product
costumer
('Jonny', 1, '1@example.com')
('Mary', 2, '5@example.com')
('John', 9, '8@example.com')
('Ann\n', 7, '0@example.com\n')
('Polly', 19, 'p@example')
Enter:
1 - create
2 - delete
3 - edit
4 - get
5 - return to main menu

```

Вимоги до пункту №2 деталізованого завдання:

Копії екрану (ілюстрації) з фрагментами згенерованих даних таблиць  
Приклад генерації 100 000 записів для таблиці costumer:

pgAdmin

Browser

- FTS Template
- Foreign Table
- Functions
- Materialized V
- Procedures
- Sequences
- Tables (5)
  - costumer
  - costumer\_
  - firm
  - firm\_produ
  - product
- Trigger Funct
- Types
- Views
- mydb
  - postgres
  - student
  - Login/Group Roles (10)
    - myuser
    - pg\_execute\_server\_progr
    - pg\_monitor
    - pg\_read\_all\_settings
    - pg\_read\_all\_stats
    - pg\_read\_server\_files

Dashboard Properties Statistics SQL Dependencies Dependents public.costumer/firm/postgres@LocalServer

Query Editor Query History

```

1 SELECT * FROM public.costumer
2 ORDER BY id ASC

```

Data Output Explain Messages Notifications

name_costumer	id	email
character varying (100)	[PK] integer	character varying (100)
100001 YY	110015	DS
100002 LO	110016	BW
100003 UE	110017	UI
100004 RE	110018	RH
100005 QP	110019	IR

Копії SQL-запитів:

Кількість значень, що згенерують функції задається користувачем (параметр number)

```
58 def random_generation(cursor, table, number):
59     try:
60         if table == 1:
61             cursor.execute("INSERT INTO costumer (name_costumer, email) "
62                             " SELECT chr(trunc(65 + random()*25)::int)|| chr(trunc(65 + random()*25)::int),"
63                             " chr(trunc(65 + random()*25)::int)|| chr(trunc(65 + random()*25)::int)"
64                             " FROM generate_series(1, %s)", [number])
65         elif table == 2:
66             cursor.execute("INSERT INTO firm (name_firm, address) "
67                             " SELECT chr(trunc(65 + random()*25)::int)|| chr(trunc(65 + random()*25)::int),"
68                             " chr(trunc(65 + random()*25)::int)|| chr(trunc(65 + random()*25)::int)"
69                             " FROM generate_series(1, %s)", [number])
70         elif table == 2:
71             cursor.execute("INSERT INTO firm (name_firm, address) "
72                             " SELECT chr(trunc(65 + random()*25)::int)|| chr(trunc(65 + random()*25)::int),"
73                             " chr(trunc(65 + random()*25)::int)|| chr(trunc(65 + random()*25)::int)"
74                             " FROM generate_series(1, %s)", [number])
75         elif table == 3:
76             cursor.execute("INSERT INTO product (name_product, price) "
77                             " SELECT chr(trunc(65 + random()*25)::int)|| chr(trunc(65 + random()*25)::int),"
78                             " trunc(random()*100 + random())::real FROM generate_series(1, %s)",
79                             [number])
80         elif table == 4:
81             cursor.execute("INSERT INTO costumer_product (article, id) SELECT article, id FROM product TABLESAMPLE"
82                             " bernoulli(%s), costumer TABLESAMPLE bernoulli(%s)", (number, number))
83     except:
```

Користувач задає процент вибірки значень з ключових таблиць:

```
80         elif table == 4:
81             cursor.execute("INSERT INTO costumer_product (article, id) SELECT article, id FROM product TABLESAMPLE"
82                             " bernoulli(%s), costumer TABLESAMPLE bernoulli(%s)", (number, number,))
83         elif table == 5:
84             cursor.execute("INSERT INTO firm_product (name_firm, article) SELECT name_firm, article"
85                             " FROM firm TABLESAMPLE"
86                             " bernoulli(%s), costumer TABLESAMPLE bernoulli(%s)", (number, number,))
87     except psycopg2.errors.UniqueViolation:
88         print("Error\n")
89
90
91 def select_function(cursor, parameter, item):
92     random_generation()
93     try:
94         if table == 1:
```

Вимоги до пункту №3 деталізованого завдання:

Ілюстрації уведення пошукового запиту та результатів виконання запитів:  
Запит 1(Вивести ім'я клієнта, що придбав товар з заданим артикулом):



```
Enter:
1 - for testing task
2 - for task 2
3 - for 3
4 - exit

3
Enter: 1, 2 or 3 for testing select
1
Enter the product article to display the name of the customer who bought it
12
Name costumer      I
('Jonny',)
Request processing time  0.035886386999865936
Enter:
1 - for testing task
2 - for task 2
3 - for 3
4 - exit
```

Запит 2(Вивести адресу фірми, що постачає товар з заданим артикулом):

```
Enter:
1 - for testing task
2 - for task 2
3 - for 3
4 - exit

3
Enter: 1, 2 or 3 for testing select
2
Enter the product article to display the address of the firm who produces it
12
Name firm Address   I
('Example M1',)
Request processing time  0.004365266000604606
Enter:
1 - for testing task
2 - for task 2
3 - for 3
4 - exit
```

Запит 3(Виведення загальної суми покупок клієнта з вказаним айді)

```
Enter:
1 - for testing task
2 - for task 2
3 - for 3
4 - exit

3
Enter: 1, 2 or 3 for testing select
3
Enter the customer ID to find out the total cost of his purchases
2
Price
(53.7,)
Request processing time  0.2125439940000433
Enter:
1 - for testing task
2 - for task 2
3 - for 3
4 - exit
```

Копії SQL-запитів:

```
94 cursor.execute("with table_1 as (SELECT id FROM costumer_product"
95               " where costumer_product.article = %s)"
96               " SELECT name_costumer FROM costumer inner join table_1 on costumer.id = table_1.id",
97               [parameter])
```



```
185 cursor.execute("with table_1 as (SELECT name_firm FROM firm_product"
186                 " where firm_product.article = %s)"
187                 " SELECT address FROM firm inner join table_1 on firm.name_firm = table_1.name_firm",
188                 [parameter])
```

```
116 cursor.execute("with table_1 as (SELECT article FROM costumer_product"
117                 " where costumer_product.id = %s)"
118                 " SELECT sum(price) FROM product inner join table_1 on product.article = table_1.article",
119                 [parameter])
```