



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Лабораторна робота № 2

з дисципліни “ Бази даних ”

тема “ Створення додатку бази даних, орієнтованого на взаємодію з СУБД
PostgreSQL”

Виконав
студент II курсу
групи КП-92

Яковлєв Денис Сергійович
(прізвище, ім'я, по батькові)

Перевірів
“ ____ ” “ ____ ” 20__ р.
викладач

Петрашенко А.В.
(прізвище, ім'я, по батькові)

Київ 2020

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

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції 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).

URL репозиторію: <https://github.com/DenisYakovlev/KP92-Yakovlev-BD/tree/master/lab2>

Завдання 1

Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.

```
main x
1. Read items in table
2. Insert item to table
3. Update item in table
4. Read all products in order
5. Read all orders in dates
6. Read all products in dates
7. Read products by restriction
8. Generate some data
9. Exit

command: 1
Enter table name: products
id|name|price|fk_department|fk_category
(1, 'Apple Juice', 30, 1, 1)
(100004, 'Pineapple Juice', 37, 1, 1)
(2, 'Orange Juice', 33, 1, 1)
(3, 'Marlboro Red', 55, 1, 2)

0.0009591579437255859 ms
Press any key to continue

Commands:
1. Read items in table
2. Insert item to table
3. Update item in table
4. Read all products in order
5. Read all orders in dates
6. Read all products in dates
7. Read products by restriction
8. Generate some data
9. Exit

command: 3
Enter item id: 1
Enter table name: products
id|name|price|fk_department|fk_category
Enter data(coma split, dict): price=33
Just updated item with id#1!

0.0467379093170166 ms
```

Run 6: Problems Terminal Python Console TODO

PC W

```
4. Read all products in dates
7. Read products by restriction
8. Generate some data
9. Exit

command: 1
Enter table name: products

Error with displaying items:
('Database select error: \n', 'ОШИБКА: отношение "products" не существует\nLINE 1: SELECT * FROM products\n
table name: products
Press any key to continue

command: 5
Enter date range(start/end): 2020-12/2020-12
Wrong date format. Try again

Enter date range(start/end): 2020-12-12/2020-12-14
Error with displaying items:

command: 2
Enter table name: products
id|name|price|fk_department|fk_category
Enter data(strings must be in columns): 'Pineapple Juice',37,1,1
Just added new item!





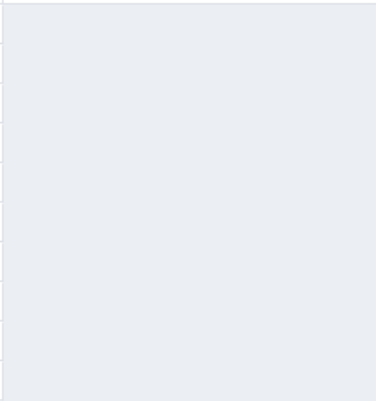
0.22258281707763672 ms
Press any key to continue
```

Завдання 2

Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.

```
1 SELECT * FROM orders
2 WHERE date BETWEEN '2020-06-10' AND '2020-11-12'
```

Data Output Explain Messages Notifications

	 id [PK] integer 	date date 	customer text 	
1	1	2020-08-13	c62cae0f9a32e49c9bb8216bb5c22926	
2	3	2020-10-18	4f5af83406b4733f72a9a079a54f26c8	
3	5	2020-10-13	0363642dd72ba1072876d2b218722ad0	
4	10	2020-07-17	5a38aeab42d0266f4df49c967ea07a3f	
5	12	2020-10-23	d4fba41f7c395839509b8dac12965656	
6	13	2020-07-10	5b108c2a9ccc26aa098b88dbb347e1b8	
7	15	2020-09-05	8743627e4c6e1a23264725c79813554e	
8	16	2020-06-16	9c55be4a470bd2f73c3ec1bac48e3dd8	
9	22	2020-06-29	0ba847782ec622bb9e5008610d8d8f6b	
10	28	2020-08-03	812012497ca54b985443e77a2a1576ff	

```

1 SELECT * FROM products
2 WHERE name LIKE '%ab%'

```

Data Output Explain Messages Notifications

	id [PK] integer	name text	price integer	fk_department integer	fk_category integer
1	24	abec36d68e47392f5f192d38acba9077	191	76956	25746
2	26	781417f2248c07a42afe92ef55aabc02	604	72594	47857
3	30	670852fb818a2ec1489fede09a3b6ab9	841	65539	41410
4	31	8db4ab1b863c8d6ff806652f4823dc62	125	15137	11779
5	45	1b81d1495465ebc7b50ad17e2f2aab82	58	99445	22328
6	48	bed75b9ebd1fd624f91b2d266b06ab22	70	19608	19419
7	51	2357ed2d6b4f41abaffb504e298a3a06	310	87998	33562
8	59	85dc62fcc486ab03e0dadfd4142aca6c	275	30665	31234
9	61	a2e25fb15a18b05391567aab3a1b6afc	800	73090	40043
10	65	6fdff5d49b8cd58961a0a721abff268a	513	39770	9157
11	82	b0fc4e69d70ba8d9406496eab7c97d26	36	22778	4899



```
Query Editor  Query History

1  INSERT INTO categories(name, age_restricted)
2  (SELECT md5(random()::text), (round(random())::int)::boolean FROM generate_series(1,5));
3
4  INSERT INTO departments(name, location)
5  (SELECT md5(random()::text), md5(random()::text) FROM generate_series(1,100000));
6
7  INSERT INTO orders(date, customer)
8  (SELECT timestamp '2020-01-01 20:00:00' + random() *
9  (timestamp '2020-12-31 20:00:00' - timestamp '2020-01-01 10:00:00'),
10 md5(random()::text) FROM generate_series(1,50000));
11
12 INSERT INTO products(name, price, fk_department, fk_category)
13 (SELECT md5(random()::text), floor(random() * 1000 + 1)::int,
14 floor(random() * (SELECT COUNT(*) FROM departments) + 1)::int, floor(random() * (SELECT COUNT(*) FROM categories) + 1)::int FROM generate_series(1,100000));
15
16 INSERT INTO order_products_relations(order_id, product_id)
17 (SELECT floor(random() * (SELECT COUNT(*) FROM orders) + 1)::int, floor(random() * (SELECT COUNT(*) FROM products) + 1)::int FROM generate_series(1,50000));

Data Output  Explain  Messages  Notifications
```

Завдання 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.

```

Structure
  1. Read items in table
  2. Insert item to table
  3. Update item in table
  4. Read all products in order
  5. Read all orders in dates
  6. Read all products in dates
  7. Read products by restriction
  8. Generate some data
  9. Exit

command: 4
Enter order id: 2
(2, 'Orange Juice', 33, 1, 1, 3, 2, 2)

0.02925395965576172 ms
Press any key to continue

Commands:
1. Read items in table
2. Insert item to table
3. Update item in table
4. Read all products in order
5. Read all orders in dates
6. Read all products in dates
7. Read products by restriction
8. Generate some data
9. Exit

command: 5
Enter date range(start/end): 2020-12-12/2020-12-14
(1, datetime.date(2020, 12, 12), 'Oleg')
(2, datetime.date(2020, 12, 13), 'Jack')
(3, datetime.date(2020, 12, 14), 'Dasha')

0.0 ms
Press any key to continue

```



```

Commands:
1. Read items in table
2. Insert item to table
3. Update item in table
4. Read all products in order
5. Read all orders in dates
6. Read all products in dates
7. Read products by restriction
8. Generate some data
9. Exit

command: 6
Enter date range(start/end): 2020-12-12/2020-12-13
(1, 'Apple Juice', 30, 1, 1)
(2, 'Orange Juice', 33, 1, 1)

0.0030176639556884766 ms
Press any key to continue

Commands:
1. Read items in table
2. Insert item to table
3. Update item in table
4. Read all products in order
5. Read all orders in dates
6. Read all products in dates
7. Read products by restriction
8. Generate some data
9. Exit

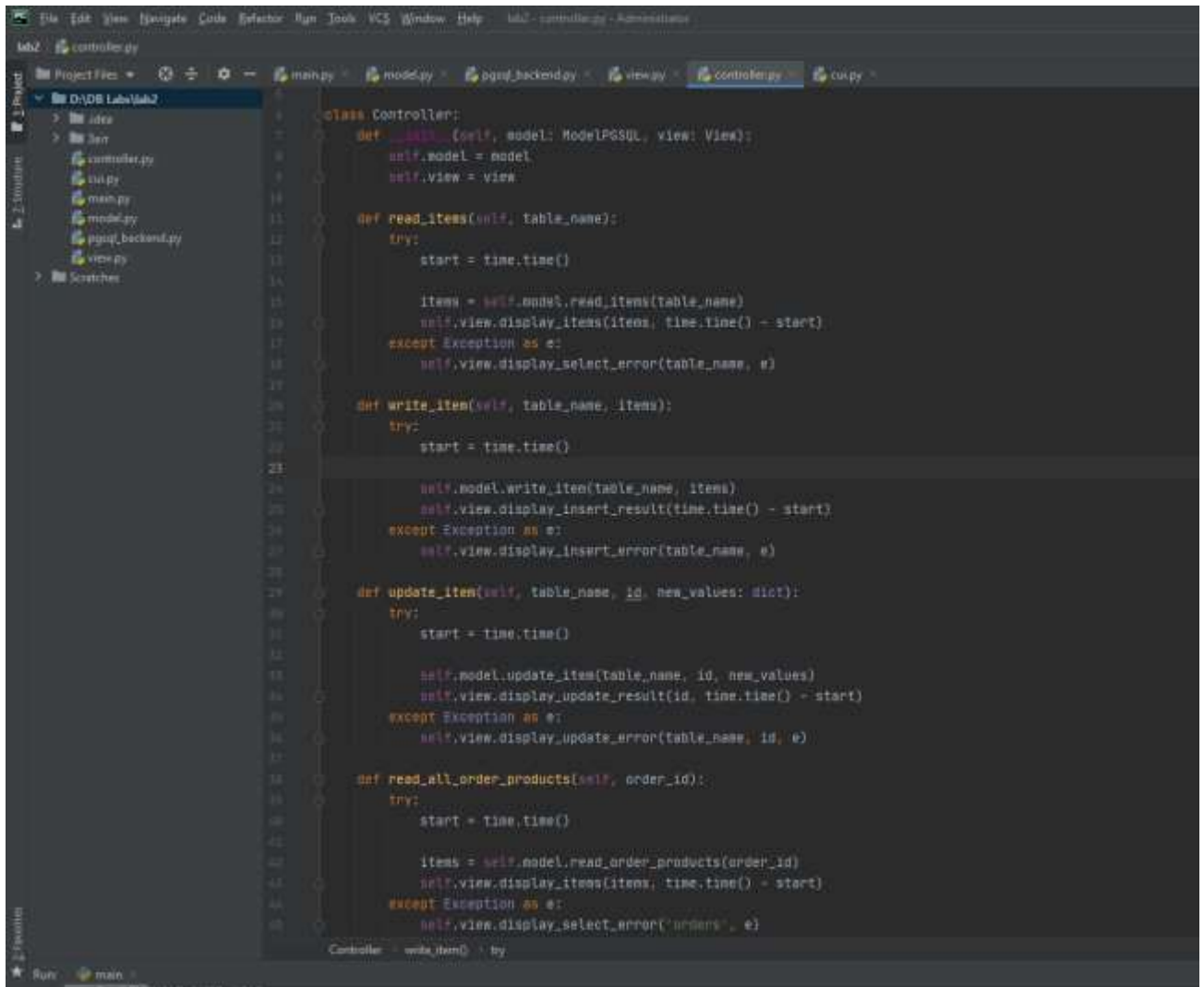
command: 7
Restricted: False
False
(1, 'Apple Juice', 30, 1, 1, 1, 'Juice', False)
(100004, 'Pineapple Juice', 37, 1, 1, 1, 'Juice', False)
(2, 'Orange Juice', 33, 1, 1, 1, 'Juice', False)

0.002017498016357422 ms
Press any key to continue

```

Завдання 4

Програмний код виконати згідно шаблону MVC (модель-подання-контролер).



```
1 class Controller:
2     def __init__(self, model: ModelPGSQL, view: View):
3         self.model = model
4         self.view = view
5
6     def read_items(self, table_name):
7         try:
8             start = time.time()
9
10            items = self.model.read_items(table_name)
11            self.view.display_items(items, time.time() - start)
12        except Exception as e:
13            self.view.display_select_error(table_name, e)
14
15    def write_item(self, table_name, items):
16        try:
17            start = time.time()
18
19            self.model.write_item(table_name, items)
20            self.view.display_insert_result(time.time() - start)
21        except Exception as e:
22            self.view.display_insert_error(table_name, e)
23
24    def update_item(self, table_name, id, new_values: dict):
25        try:
26            start = time.time()
27
28            self.model.update_item(table_name, id, new_values)
29            self.view.display_update_result(id, time.time() - start)
30        except Exception as e:
31            self.view.display_update_error(table_name, id, e)
32
33    def read_all_order_products(self, order_id):
34        try:
35            start = time.time()
36
37            items = self.model.read_order_products(order_id)
38            self.view.display_items(items, time.time() - start)
39        except Exception as e:
40            self.view.display_select_error('orders', e)
```



```
lab2 - model.py - Administrator
File Edit View Navigate Code Refactor Run Tools VCS Window Help
lab2 model.py
Project Files
Project
  DADB Labs/lab2
    .idea
    .git
    controller.py
    cul.py
    main.py
    model.py
    postgresql_backend.py
    view.py
  Scratches
1 from postgresql_backend import DataBase
2
3
4 class ModelPGSQL:
5     def __init__(self, dbname, password, user):
6         self.db = DataBase(dbname, password, user)
7
8     def read_items(self, table_name):
9         return self.db.select_all(table_name)
10
11     def write_item(self, table_name, item):
12         self.db.insert(table_name, item)
13
14     def write_items(self, table_name, items):
15         self.db.insert_many(table_name, items)
16
17     def update_item(self, table_name, id, new_values):
18         self.db.update(table_name, id, new_values)
19
20     def read_order_products(self, order_id: int):
21         return self.db.select_order_products(order_id)
22
23     def read_orders_between(self, start, end):
24         return self.db.select_orders_between(start, end)
25
26     def read_products_between(self, start, end):
27         return self.db.select_products_between(start, end)
28
29     def read_items_with_restriction(self, restriction: bool):
30         return self.db.select_product_by_restriction(restriction)
31
32     def read_columns_of_table(self, table_name):
33         return self.db.select_table_columns_name(table_name)
34
35     def generate(self):
36         self.db.generate()
```

```
lab2 - psycopg_backend.py - Administrator
File Edit View Navigate Code Refactor Run Tools VCS Window Help lab2 - psycopg_backend.py
lab2 psycopg_backend.py
Project Files
D:\DB Labs\lab2
  > idea
  > 3air
    controller.py
    cui.py
    main.py
    model.py
    psycopg_backend.py
    view.py
  > Scratches
1
2
3
4
5 class DataBase:
6     def __init__(self, dbname, password, user):
7         try:
8             self.conn = psycopg2.connect(dbname=dbname, password=password,
9                                           user=user, host='localhost')
10            self.cursor = self.conn.cursor()
11        except OperationalError as err:
12            raise Exception('Init error occurred:\n', str(err))
13
14    def __del__(self):
15        self.close()
16
17    def is_open(self):
18        return not self.conn.closed()
19
20    def close(self):
21        self.conn.close()
22        self.cursor.close()
23
24    def select_all(self, table_name):
25        try:
26            self.cursor.execute(f'SELECT * FROM {table_name}')
27            items = self.cursor.fetchall()
28            return items
29
30        except Exception as err:
31            raise Exception('DataBase select error: \n', str(err))
32
33    def insert(self, table_name, item):
34        try:
35            values = ','.join(map(str, item))
36            self.cursor.execute(f'INSERT INTO {table_name} VALUES (default, {values})')
37
38            self.conn.commit()
39        except Exception as err:
40            raise Exception('DataBase insert error:\n', str(err))
41
42    def insert_many(self, table_name, items):
43        try:
44            values = ','.join(str(item) for item in items)
```