

**Міністерство освіти та науки України**  
**Національний технічний університет України**  
**“Київський політехнічний інститут імені Ігоря Сікорського”**  
**Факультет прикладної математики**

**Розрахунково-графічна робота**  
**із дисципліни**

**“Бази даних та засоби управління”**

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

**Виконав: Пилипенко Тимофій.**

**Студент групи КВ-32**

**Telegram: @NeLern**

**Київ 2025**

Загальне завдання роботи полягає у наступному:

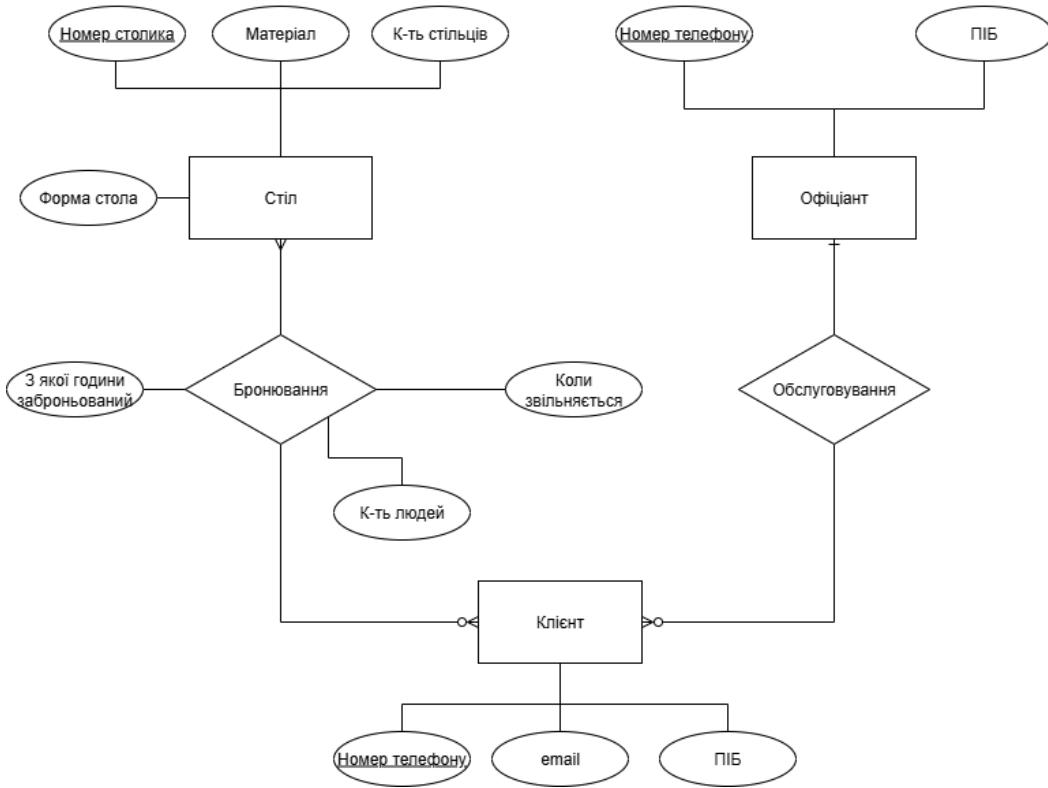
1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами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

Репозиторій: <https://github.com/LernL/BD>;

Мал. 1.1. Діаграма сутність-зв'язок бази даних

### Ресторанний портал для бронювання столиків

-Стіл(Номер столика, кількість стільців)  
-Офіціант(ПІБ, номер телефону, id офіціанта)  
-Клієнт(ПІБ, номер телефону, email, id клієнта, к-ть людей)



В базі даних є 5 таблиці:

• Клієнт, який взаємодіє зі столиком й офіціантом. Він має такі атрибути: Номер телефону(pk), email й номер телефону офіціанта, який його обслуговує.

- Контакти клієнтів, які має email клієнта(pk) й його ім'я.
- Столик, який має номер столика(pk), к-ть стільців, форма стола, матеріал, з якого він зроблен.
- Офіціант має такі атрибути: ПІБ, номер телефону(pk).
- Бронювання має час бронювання, час кінця бронювання, номер стола, номер клієнта, кількість людей й id(pk) бронювання

Меню має 7 пунктів:

1. Waiter:

- 1.1 Add
- 1.2 List
- 1.3 Update
- 1.4 Delete
- 1.5 Back

2. Clients:

- 2.1 Add
- 2.2 List
- 2.3 Update
- 2.4 Delete
- 2.5 Back

3. Tables:

- 3.1 Add
- 3.2 List
- 3.3 Update
- 3.4 Delete
- 3.5 Back

4. Booking:

- 4.1 Add
- 4.2 List
- 4.3 Update
- 4.4 Delete
- 4.5 Back

5. Contacts:

- 5.1 Add

5.2 List

5.3 Update

5.4 Delete

5.5 Back

6. Analytics:

6.1 By table/hour

6.2 By waiter

6.3 Show booking for table

6.4 Exit

7. Exit

Мова програмування: Python;

Використані бібліотеки: math, psycopg2;

1. Дочірня таблиця Client для Waiter(атрибут phone\_num\_w):

Мал. 1.2. Додавання запису в таблицю Waiter

Waiters: 1) Add 2) List 3) Update 4) Delete 0) Back		
5 > 2		
-----		
Employee number		Name
-----		
111	Sam	

Мал. 1.3. Додавання запису в таблицю Client

Clients: 1) Add 2) List 3) Update 4) Delete 0) Back		
5 > 2		
-----		
Phone number client		Email
-----		
222	aaa@gmail.com	111
1002161011	gomev@bikqwwg.com	292924517865

Мал. 1.4. Спроба видалення зв'язаного запису з таблиці Waiter

Waiters: 1) Add 2) List 3) Update 4) Delete 0) Back		
> 4		
Phone number: 111		
Error: ПОМИЛКА: update або delete в таблиці "waiter" порушує обмеження зовнішнього ключа "client_phone_num_w_fkey3" таблиці "client" DETAIL: На ключ (phone_num_w)=(111) все ще є посилання в таблиці "client".		

Мал. 1.5. Запис в таблиці Client залишилась

Phone number client	Email	Phone number waiter
222	aaa@gmail.com	111
1002161011	gomev@bikqwwg.com	292924517865

Видалення запису з батьківської таблиці неможливе, оскільки в дочірній таблиці існують записи, що посилаються на нього через зовнішній ключ. Для цього зв'язку не встановлено жодної дії при видаленні (ON DELETE NO ACTION або ON DELETE RESTRICT), тому СУБД забороняє видалення, щоб не порушити цілісність даних.

Мал. 1.6. Запис в таблиці Client не існуючого запису з таблиці Waiter неможливе

```
Clients: 1) Add 2) List 3) Update 4) Delete 0) Back
> 1
Phone number client: 444
Email: bbb@gmail.com
Phone number waiter: 333
Error: ПОМИЛКА: insert або update в таблиці "client" порушує обмеження зовнішнього ключа "client_phone_num_w_fkey3"
DETAIL: Ключ (phone_num_w)=(333) не присутній в таблиці "waiter".
```

2. Ілюстрації згенерованих таблиць та відповідних sql записів наведено далі:

Мал. 2.1. Згенерована таблиця waiter

name	phone_num_w
Sam	111
Kwemasrl	1016408524
Fdfdfhy	1031875267
Juvfahf	1033636464
Guryebpl	1042213880
Vaxneraf	1043778215
Dbugpkke	1045038202
Tetehler	1047532003
Rashfuply	1048337961

Sql запит:

```
BEGIN;
TRUNCATE TABLE public.waiter RESTART IDENTITY CASCADE;
INSERT INTO public.waiter (name, phone_num_w)
```

```

SELECT      chr(trunc(65+random()*25)::int) || 
Lower(chr(trunc(65+random()*25)::int)) || chr(trunc(65+random()*25)::int) || 
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) || 
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) || 
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ) AS name,
(random()*1000000000000)+1000000000::bigint
FROM generate_series(1,100000);

```

COMMIT;

Мал. 2.2. Згенерована таблиця contacts\_client

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under the 'Tables' section, the 'contacts\_client' table is selected. The Data Output tab displays the results of the query: 'SELECT \* FROM public.contacts\_client ORDER BY email ASC'. The results show 10 rows of data, each containing an 'email' column (PK) and a 'name' column.

	email [PK] character varying	name character varying
1	aaa@gmail.com	Tom
2	aaagi@exwrl.com	Mmlaoeokt
3	aaajl@ipbjxi.com	Xuwmpqxcb
4	aaajh@pjprfrv.com	Jbleqehvn
5	aaakk@yndyhy.com	Vversdrtt
6	aaala@jmsktp.com	Bdwopyxvt
7	aaaly@vdkrnq.com	Adsvrdrne
8	aaanr@gmhrvvk.com	Tsukiwbkw
9	aaaox@wwwbyvr.com	Xhwevdak

Sql запит:

```

BEGIN;
TRUNCATE TABLE public.contacts_client RESTART IDENTITY CASCADE;
INSERT INTO public.contacts_client (email, name)
SELECT      Lower(      chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) || '@' ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || '.com' ) AS name,
chr(trunc(65+random()*25)::int) || Lower( (
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||

```

```

chr(trunc(65+random()*25)::int)      ||      chr(trunc(65+random()*25)::int)      ||
chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) )
FROM generate_series(1,100000);

```

COMMIT;

Мал. 2.3. Згенерована таблиця client

	email	phone_num_cl	phone_num_w
1	aaa@gmail.com	222	111
2	gomer@qblkqwgg.com	1002161011	292924517865
3	rwlkm@eqksjos.com	1002742588	253162632629
4	yrkts@wabvrra.com	1005664755	362492191356
5	khstv@funnalmh.com	1005665542	422423427089
6	hsygf@oocgtmm.com	1015926995	360901120613
7	kodwh@qbhffn.com	1025227994	977295459465
8	cynco@cujigto.com	1045211955	307311229075
9	wseht@mgojjug.com	1058358980	770882859544

Sql запит:

BEGIN;

```
TRUNCATE TABLE public.client RESTART IDENTITY CASCADE;
```

```

WITH emails AS (
    SELECT email, row_number() OVER () AS id
    FROM public.contacts_client
    ORDER BY random()
    LIMIT 100000
),
waiters AS (
    SELECT phone_num_w, row_number() OVER () AS id
    FROM public.waiter
    ORDER BY random()
    LIMIT 100000
)
INSERT INTO public.client (phone_num_w, email, phone_num_cl)

```

```

SELECT
w.phone_num_w,
e.email,
(random() * 1000000000000 + 1000000000)::bigint
FROM emails e
JOIN waiters w USING (id);

```

COMMIT;

Мал. 2.4. Згенерована таблиця table

num_table	number_chairs	material	table_shape
1	1	2	stone
2	2	7	metal
3	3	4	wood
4	4	4	glass
5	5	6	plastic
6	6	7	stone
7	7	8	wood
8	8	3	stone
9	9	1	wood

Sql запит:

BEGIN;

TRUNCATE TABLE public.table RESTART IDENTITY CASCADE;

```

WITH params AS (
SELECT
ARRAY['wood','metal','plastic','glass','stone'] AS materials,
ARRAY['square','circle','rectangle','oval']      AS shapes
)

```

```

INSERT INTO public.table (num_table, number_chairs, material, table_shape)
SELECT
gs AS num_table,
(floor(random()*8) + 1)::int AS number_chairs,

```

```

params.materials[(floor(random()*array_length(params.materials,1)) + 1)::int]
AS material,
params.shapes[(floor(random()*array_length(params.shapes,1)) + 1)::int]
AS table_shape
FROM generate_series(1, 100000) AS gs
CROSS JOIN params;

```

COMMIT;

Мал. 2.5. Згенерована таблиця booking

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer displays the database schema, including the 'booking' table under the 'Tables (5)' section. In the center, the Query tool contains the following SQL code:

```

SELECT * FROM public.booking
ORDER BY id ASC

```

The Data Output tab shows the results of the query, displaying 10 rows of data from the 'booking' table. The columns and their data types are:

	time_book	end_time	num_table	phone_num_cl	people_num	id [PK]
1	4	5	24100	49915200918	5	1
2	16	17	89396	716125163114	1	2
3	24	24	47869	68344438163	4	3
4	8	9	24094	992766750537	8	4
5	10	11	95494	407182335224	8	5
6	12	13	41700	586612591211	1	6
7	7	8	87895	465241162057	5	7
8	6	7	68018	865950411614	4	8
9	15	16	96439	873945226802	7	9

Sql запит:

BEGIN;

TRUNCATE TABLE public.booking RESTART IDENTITY CASCADE;

```

WITH numtb AS (
  SELECT num_table, row_number() OVER (ORDER BY random()) AS rn
  FROM public."table"
  LIMIT 100000
),
numcl AS (
  SELECT phone_num_cl, row_number() OVER (ORDER BY random()) AS rn
  FROM public.client
  LIMIT 100000
)

```

```

INSERT INTO public.booking (id, time_book, end_time, num_table,
phone_num_cl, people_num)
SELECT
    row_number() OVER (ORDER BY random()) AS id,
    s.time_book,
    LEAST(24, s.time_book + (floor(random() * (24 - s.time_book))::int + 1)) AS end_time,
    s.num_table,
    s.phone_num_cl,
    s.people_num
FROM (
    SELECT nt.num_table, nc.phone_num_cl,
        (floor(random()*24)::int + 1) AS time_book,
        (floor(random()*8)::int + 1) AS people_num,
        nt.rn
    FROM numtb nt
    JOIN numcl nc USING (rn)
) s;

```

COMMIT;

3. Ілюстрації запитів й відповідних sql записів наведено далі:

Мал. 3.1 Пошуковий запит бронювання з 1 до 5 години

num_table	table_shape	material	start_hour	end_hour	avg_people
53	oval	wood	1	4	1
117	square	stone	1	5	3
139	square	stone	1	2	4
210	oval	glass	1	5	2
459	oval	metal	1	5	4
514	circle	glass	1	4	4
570	oval	stone	1	4	4
633	oval	plastic	1	4	4
819	square	stone	1	3	5
841	rectangle	stone	1	2	2
1233	rectangle	wood	1	2	2
1420	rectangle	stone	1	5	6
1453	rectangle	metal	1	3	7
1525	oval	plastic	1	3	7
1542	oval	plastic	1	4	7
1570	square	plastic	1	3	7
1785	rectangle	glass	1	4	8

Sql запит:

```
SELECT
    t.num_table,
    t.table_shape,
    t.material,
    MIN(b.time_book) AS start_hour,
    MAX(b.end_time) AS end_hour,
    ROUND(AVG(b.people_num))::int AS avg_people
FROM booking b
JOIN "table" t ON b.num_table = t.num_table
{where_clause}
GROUP BY t.num_table, t.table_shape, t.material
{having_clause}
ORDER BY start_hour, t.num_table
LIMIT %s
```

### Мал. 3.2. Пошуковий запит по імені офіціанта

The screenshot shows a Python development environment with several windows open:

- Project**: Shows a file tree with a folder named "RGR" containing "code", "idea", and "analytics". Inside "analytics", there are files: controller.py, model.py, and view.py. The "model.py" file is currently selected.
- Code Editor**: The "model.py" file contains the following code:

```
18
19
20
21
22
23     q = f"""
24         SELECT
25             t.num_table,
```
- Terminal**: Displays the output of a command, showing a table of data and a time measurement.

	num_table	start_hour	end_hour	total_people	avg_people
Nphctrmh	2	3	8	8	
Ngupunin	2	5	6	6	
Nuerjuua	2	5	2	2	
Nwukhod	2	4	1	1	
Nyelvbsg	2	4	8	8	
Nyodobvu	2	4	3	3	

Time: 168.38 ms
- Output**: Shows a command-line interface for analytics:

```
Analytics: 1) By table/hour 2) By waiter 3) Show bookings for table 0) Back
> 2
waiter name (partial, enter skip): Aartgeop
hour from (0-23, enter to skip):
hour to (0-23, enter to skip):
    waiter_name | start_hour | end_hour | total_people | avg_people
    -----+-----+-----+-----+-----+
    Aartgeop | 1 | 5 | 1 | 1
Time: 95.00 ms
```
- Bottom Status Bar**: Shows the path "RGR > code > analytics > model.py", and system information like "23:4 (452 chars, 13 line breaks)", "CRLF", "UTF-8", "4 spaces", and "Python 3.12".

Sql запрос:

## SELECT

```
w.name AS waiter_name,  
MIN(b.time_book) AS start_hour,  
MAX(b.end_time) AS end_hour,  
SUM(b.people_num) AS total_people,  
ROUND(AVG(b.people_num))::int AS avg_people  
FROM booking b  
JOIN "table" t ON b.num_table = t.num_table
```

```

LEFT JOIN client c ON b.phone_num_cl = c.phone_num_cl
LEFT JOIN waiter w ON c.phone_num_w = w.phone_num_w
{where_clause}
GROUP BY w.name
{having_clause}
ORDER BY start_hour, w.name
LIMIT %s

```

Мал. 3.3. Пошуковий запит по номеру стола

```

Project: RGR
File: main.py
Line: 72
    LEFT JOIN client c ON b.phone_num_cl = c.phone_num_cl
Line: 73
    LEFT JOIN waiter w ON c.phone_num_w = w.phone_num_w
Line: 74
    {where_clause}
Line: 75
    GROUP BY w.name
Line: 76
    {having_clause}
Line: 77
    ORDER BY start_hour, w.name
Line: 78
    LIMIT %s
Line: 79
    """
Line: 80
    exec_params = list(params)
Line: 81
Model: search_by_waiter()

```

```

Run: main
-----
Aartgeop | 1 | 5 | 1 | 1
Time: 95.00 ms
Analytics: 1) By table/hour 2) By waiter 3) Show bookings for table 0) Back
> 3
Invalid choice
Analytics: 1) By table/hour 2) By waiter 3) Show bookings for table 0) Back
> 3
num_table: 56
id | start_hour | end_hour | phone_num_cl | email | contact_name | waiter_name | people_num
... | ..... | ..... | ..... | ..... | ..... | ..... | .....
6898 | 10 | 11 | 488868892030 | mrtck@mnkjaea.com | Mxkncfoyx | Lvxyynbp | 7
Time: 18.60 ms
Analytics: 1) By table/hour 2) By waiter 3) Show bookings for table 0) Back
>

```

Sql запит:

```

SELECT
    b.id,
    b.time_book AS start_hour,
    b.end_time AS end_hour,
    b.phone_num_cl,
    c.email,
    co.name AS contact_name,
    w.name AS waiter_name,
    b.people_num
FROM booking b
LEFT JOIN client c ON b.phone_num_cl = c.phone_num_cl
LEFT JOIN contacts_client co ON c.email = co.email
LEFT JOIN waiter w ON c.phone_num_w = w.phone_num_w
WHERE b.num_table = %s
ORDER BY b.time_book, b.id

```

LIMIT %s

4. Ілюстрація програмного коду модуля “Model” й його опис:

```
import psycopg2

class Model:
    def __init__(self):
        self.conn = psycopg2.connect("dbname=postgres user=postgres
password=1234adminL host=localhost port=5432")
        self.conn.autocommit = True
        self.create_table()

    def create_table(self):
        with self.conn.cursor() as c:
            c.execute("""
CREATE TABLE IF NOT EXISTS waiter (
    name varchar(20) NOT NULL,
    phone_num_w integer NOT NULL PRIMARY KEY,
    UNIQUE (name)
);
""")
            self.conn.commit()

    def add(self, phone_num_w, name):
        with self.conn.cursor() as c:
            c.execute("INSERT INTO waiter (phone_num_w, name) VALUES
(%s,%s)", (int(phone_num_w), name))
            self.conn.commit()

    def list_all(self):
        with self.conn.cursor() as c:
            c.execute("SELECT phone_num_w, name FROM waiter ORDER BY
phone_num_w")
            return c.fetchall()

    def update(self, phone_num_w, new_name):
        with self.conn.cursor() as c:
            c.execute(
```

```
"UPDATE waiter SET name=%s WHERE phone_num_w=%s",
(new_name, int(phone_num_w))
)
self.conn.commit()

def delete(self, phone_num_w):
    with self.conn.cursor() as c:
        c.execute("DELETE FROM waiter WHERE phone_num_w=%s",
(int(phone_num_w),))
    self.conn.commit()

def get_input(self):
    name = input("Name: ").strip()
    return name

def get_key(self):
    return input("Phone number: ").strip()
```

Короткий опис:

\_\_init\_\_(self)

Ініціалізує підключення до бази даних PostgreSQL та викликає метод створення таблиці.

**create\_table(self)**

Створює таблицю, якщо вона ще не існує.

**add(self, ...)**

Додає новий запис у таблицю з указаними параметрами.

**list\_all(self)**

Повертає всі записи з таблиці.

**update(self)**

Оновлює запис таблиці.

**delete(self, phone\_num\_w)**

Видаляє запис із таблиці.

`get_input(self)`

Зчитує з клавіатури значення для додавання або оновлення запису.

`get_key(self)`

Зчитує з клавіатури значення ключового поля.