Е

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп'ютерних систем

Розрахунково-графічна робота

з дисципліни **Бази даних і засоби управління**

на тему: "Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL"

Виконав:
студент III курсу
групи КВ-21
Саюк В.А.
Перевірив:

Київ – 2024

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

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 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).

Опис предметної області

Предметна область – *онлайн-платформа для зберігання та пошуку наукових публікацій.*

Вона охоплює набір засобів і інструментів, які спрямовані на ефективне управління та доступ до наукових досліджень і статей. Вона дозволяє зберігати різноманітні типи публікацій, включаючи журнальні статті, тези конференцій, наукові звіти та дисертації, надаючи можливість їх швидкого пошуку за ключовими словами, авторами або темами. Ця система спрощує доступ до актуальних наукових знань, підтримуючи дослідників, викладачів та студентів у пошуку необхідних матеріалів. Завдяки аналітичним інструментам платформи користувачі можуть відстежувати популярність публікацій, цитування та останні тенденції в певних галузях науки, що сприяє розвитку дослідницької діяльності та наукової співпраці.

Опис сутностей

Для побудови бази даних для *онлайн-платформи зберігання та пошуку наукових публікацій*, були виділені такі сутності:

Collection (Збірник)

- Атрибути:
- *Collection_id*: Ідентифікатор збірника (унікальний ключ для кожного збірника).
- *name*: Назва збірника, яка визначає його унікальність або зміст (наприклад, "Збірник наукових праць").
- *type*: Формат видання збірника, що може включати підручники, журнальні статті,

матеріали конференцій тощо.
- *view*: Статус збірника, який вказує на географічну спрямованість: всеукраїнський (національний) або міжнародний.

Призначення:

Збірник представляє собою організовану колекцію наукових публікацій, підручників або інших навчальних матеріалів. Він дозволяє структурувати та систематизувати публікації за певними критеріями, що полегшує доступ до необхідної інформації.

Edition (Видання)

- Атрибути:
- *Edition_id*: Унікальний ідентифікатор видання.
- *name*: Назва видання, що може вказувати на тему або серію публікацій (наприклад, "Науковий вісник" або "Журнал з фізики").
- *branch*: Галузь або науковий напрямок, до якого належить видання (наприклад, фізика, біологія, історія тощо).
- *number_of_pages*: Кількість сторінок у виданні, що може вказувати на обсяг матеріалу.
- *languages*: Мови, якими видання доступне (наприклад, українська, англійська, німецька).

Призначення:

Видання є основною одиницею публікації наукових або освітніх матеріалів. Його мета полягає у наданні інформації та досліджень у певній галузі, допомагаючи розповсюджувати знання серед наукової спільноти або студентів.

Author (Автор)

- Атрибути:
- *Author_ID*: Унікальний ідентифікатор автора.
- *name*: Ім'я автора.
- *surname*: Прізвище автора.

- Призначення:

Автор є ключовою фігурою у створенні наукових публікацій або навчальних матеріалів. Він несе відповідальність за зміст публікації, її якість та наукову значущість. В системі обліку автор дозволяє відстежувати внесок конкретної особи у певні публікації та спрощує пошук робіт за іменем чи прізвищем автора.

Опис зв’язків між сутностями

Ці три сутності(Author,Edition, Collection) взаємопов'язані через тернарний зв'язок, що представляє відносини *"багато до багатьох" (N:M).*

Це означає, що один і той самий автор може бути залучений до створення численних публікацій, які видаються в різних збірниках. Водночас одна публікація може мати кілька авторів і бути частиною одного збірника.

Такий тип зв'язку дозволяє гнучко керувати даними про авторів, видання та збірники, адже публікація може бути результатом колективної роботи кількох науковців, а також може входити до різних збірників або випусків. Така модель також спрощує роботу з багатоспрямованими дослідженнями та міждисциплінарними проєктами, де участь

різних авторів та вихід у кількох збірниках є нормою.

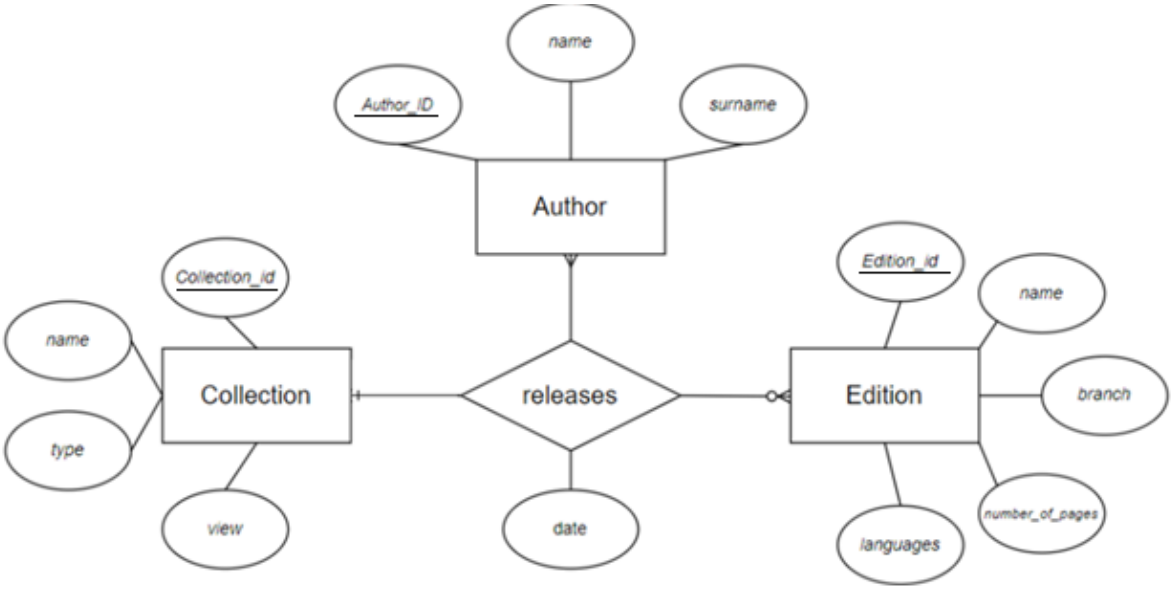


Рисунок 1 – ER-діаграма, побудована за нотацією “Вороняча лапка”

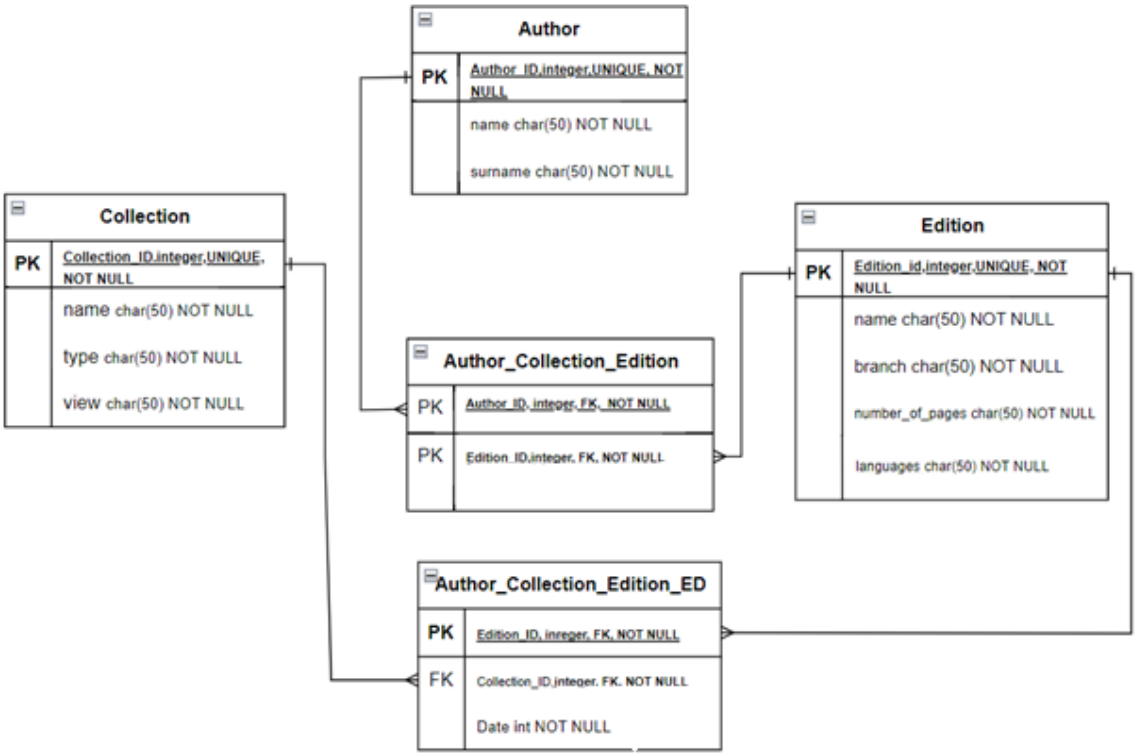
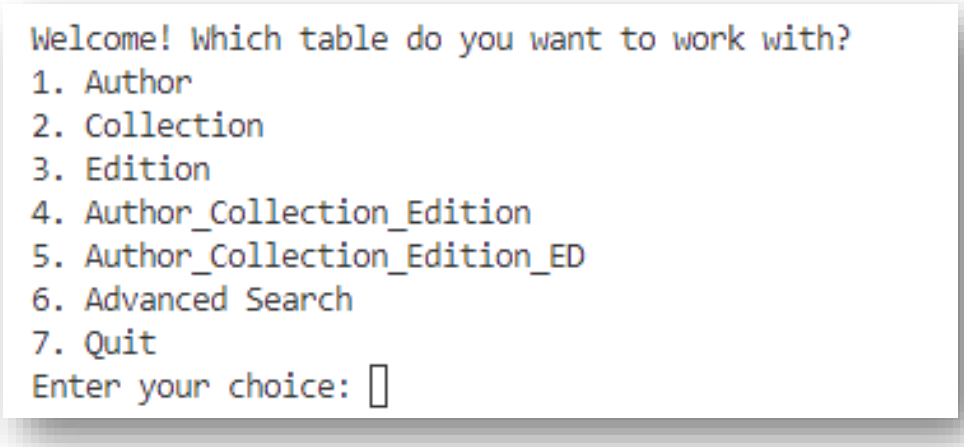


Рисунок.2 Схема бази даних

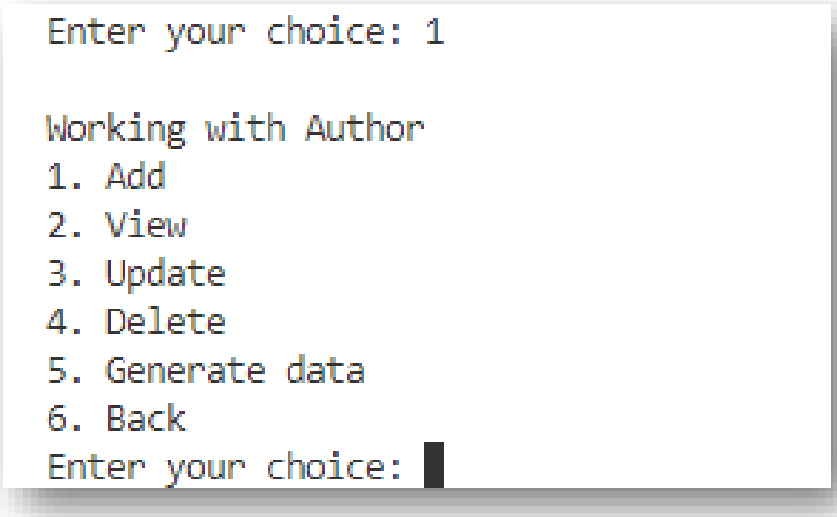
Програма

Вигляд початкового меню:



Під час запуску відображається початкове меню в якому пропонується обрати таблицю для подальшої роботи (1-5) або запити (6).

Обравши таблицю, відображаються наступне меню:



Для таблиць доступні додавання (1), перегляд (2), оновлення (3), вилучення (4) та генерування даних (5). Опція виходу (6).

Додамо автора та видання:

```
Working with Author
1. Add
2. View
3. Update
4. Delete
5. Generate data
6. Back
Enter your choice: 1
Enter author's name: John
Enter author's surname: Doe
Added successfully!

Working with Author
1. Add
2. View
3. Update
4. Delete
5. Generate data
6. Back
Enter your choice: 6

Welcome! Which table do you want to work with?
1. Author
2. Collection
3. Edition
4. Author_Collection_Edition
5. Author_Collection_Edition_ED
6. Advanced Search
7. Quit
Enter your choice: 3

Working with Edition
1. Add
2. View
3. Update
4. Delete
5. Generate data
6. Back
Enter your choice: 1
Enter edition's name: 1984
Enter edition's branch: Fiction
Enter number_of_pages: 328
Enter languages: English, Ukrainian
Added successfully!

Working with Edition
1. Add
2. View
3. Update
4. Delete
5. Generate data
6. Back
Enter your choice:
```

Результат вставки в дочірню таблицю даних яких не існує в батьківських:

Додам дані в таблицю Author_Collection_Edition з неіснуючим ID

```
Working with Author_Collection_Edition
1. Add
2. View
3. Delete
4. Generate data
5. Back
Enter your choice: 1
Enter Author_ID: 23423
Enter Edition_ID: 4
ERROR: Foreign key violation.
insert or update on table "Author_Collection_Edition" violates foreign key constraint "Author_Collection_Edition_author_id_fkey"
DETAIL:  Key (author_id)=(23423) is not present in table "Author".
```

У випадку ж додавання існуючого помилка не виникає:

```
Working with Author_Collection_Edition
1. Add
2. View
3. Delete
4. Generate data
5. Back
Enter your choice: 1
Enter Author_ID: 1
Enter Edition_ID: 1
Added successfully!
```

Перейдемо до генерації даних:

Генерація 3 випадкових збірників:

```
Welcome! Which table do you want to work with?
1. Author
2. Collection
3. Edition
4. Author_Collection_Edition
5. Author_Collection_Edition_ED
6. Advanced Search
7. Quit
Enter your choice: 2

Working with Collection
1. Add
2. View
3. Update
4. Delete
5. Generate data
6. Back
Enter your choice: 5
Enter number of data to generate: 3
Generated successfully!

Working with Collection
1. Add
2. View
3. Update
4. Delete
5. Generate data
6. Back
Enter your choice: 2
Collections:
+---------------+----------+---------+------------+
| collection_id |   name   |  type   |    view    |
+---------------+----------+---------+------------+
|       1       | hCREoOTm | Poetry  |   Paper    |
|       2       | NNEqdkgQ | Science | Electronic |
|       3       | ewgoFBkN | Science |   Paper    |
+---------------+----------+---------+------------+
```

Додам запис в Author_Collection_Edition та Author_Collection_Edition_ED:

```
Working with Author_Collection_Edition
1. Add
2. View
3. Delete
4. Generate data
5. Back
Enter your choice: 2
Author_Collection_Edition:
+-----------+------------+
| author_id | edition_id |
+-----------+------------+
|     1     |     1      |
+-----------+------------+
```

```
Working with Author_Collection_Edition_ED
1. Add
2. View
3. Delete
4. Generate data
5. Back
Enter your choice: 1
Enter Edition_ID: 1
Enter Collection_ID: 1
Enter Date (YYYY-MM-DD): 1949-06-08
Added successfully!

Working with Author_Collection_Edition_ED
1. Add
2. View
3. Delete
4. Generate data
5. Back
Enter your choice: 2
Author_Collection_Edition_ED:
+------------+---------------+------------+
| edition_id | collection_id |    date    |
+------------+---------------+------------+
|     1      |       1       | 1949-06-08 |
+------------+---------------+------------+
```

Тепер перевіримо програму на вилучення з батьківської таблиці даних:

Було реалізовано каскадне видалення. Якщо запис про автора видаляється з таблиці, то будуть видалені і відповідні записи за таблиці Author_Collection_Edition

Приклад:

```
Working with Author
1. Add
2. View
3. Update
4. Delete
5. Generate data
6. Back
Enter your choice: 2
Authors:
+-----------+-------+---------+
| author_id | name  | surname |
+-----------+-------+---------+
|     3     | bXWAM | gLfguOZ |
|     4     | rblHg | lweJUsU |
|     5     | wjyzX | ZhnfnMm |
|     6     | VJjfL | MopgVao |
|     7     | fkNVk | RgGLOeU |
+-----------+-------+---------+
```

```
Working with Author_Collection_Edition
1. Add
2. View
3. Delete
4. Generate data
5. Back
Enter your choice: 2
Author_Collection_Edition:
+-----------+------------+
| author_id | edition_id |
+-----------+------------+
|     3     |     1      |
+-----------+------------+
```

Тепер видалю запис з Author і перегляну, чи залишиться запис у Author_collection_edition:

```
Working with Author
1. Add
2. View
3. Update
4. Delete
5. Generate data
6. Back
Enter your choice: 4
Enter author_id: 3
Deleted successfully!
```

```
Welcome! Which table do you want to work with?
1. Author
2. Collection
3. Edition
4. Author_Collection_Edition
5. Author_Collection_Edition_ED
6. Advanced Search
7. Quit
Enter your choice: 4

Working with Author_Collection_Edition
1. Add
2. View
3. Delete
4. Generate data
5. Back
Enter your choice: 2
Author_Collection_Edition:
+-----------+------------+
| author_id | edition_id |
+-----------+------------+
+-----------+------------+
```

Генерація 100 000 даних у таблицю Author:

```
Welcome! Which table do you want to work with?
1. Author
2. Collection
3. Edition
4. Author_Collection_Edition
5. Author_Collection_Edition_ED
6. Advanced Search
7. Quit
Enter your choice: 1

Working with Author
1. Add
2. View
3. Update
4. Delete
5. Generate data
6. Back
Enter your choice: 5
Enter number of data to generate: 100000
Generated successfully!
```

Query | Query History

```
1 SELECT * FROM public."Author"
2 ORDER BY author_id DESC LIMIT 100
3
```

Data Output | Messages | Notifications

	author_id [PK] integer	name character varying	surname character varying
1	100007	HfYNj	RVnvJcV
2	100006	CKFIJ	LefAXsn
3	100005	WliTJ	MBMkpDS
4	100004	dBtBx	QcybVdH
5	100003	DTlGU	oiZjDYr
6	100002	qXrXQ	aqWpNwE
7	100001	FimuL	uTTpVrs
8	100000	mEcjA	ispZLUW
9	99999	jeSix	CzLBxBs
10	99998	mZIFv	NcWqprF
11	99997	KaGFi	AcuBgCL

Згенеруємо дані і для таблиці Edition та Author_Collection_Edition:

```
Welcome! Which table do you want to work with?
1. Author
2. Collection
3. Edition
4. Author_Collection_Edition
5. Author_Collection_Edition_ED
6. Advanced Search
7. Quit
Enter your choice: 3

Working with Edition
1. Add
2. View
3. Update
4. Delete
5. Generate data
6. Back
Enter your choice: 5
Enter number of data to generate: 10000
Generated successfully!
```

```
Working with Author_Collection_Edition
1. Add
2. View
3. Delete
4. Generate data
5. Back
Enter your choice: 4
Enter number of data to generate: 10000
Generated successfully!
```

Query | Query History

```
1 SELECT * FROM public."Edition"
2 ORDER BY edition_id DESC LIMIT 100
3
```

Data Output | Messages | Notifications

	edition_id [PK] integer	name character varying	branch character varying	number_of_pages integer	languages character varying
1	10001	aGLmYI	Poetry	785	Ukrainian
2	10000	iAkfau	Non-Fiction	622	French
3	9999	FwlGlH	Science	414	French
4	9998	WgXcbk	Fiction	665	French
5	9997	EdlkDW	Science	920	English
6	9996	Jgzeis	Poetry	92	French
7	9995	szmmcV	Science	655	English
8	9994	DdMvoL	Fiction	341	Ukrainian

Query | Query History

```
1 SELECT * FROM public."Author_Collection_Edition"
2 ORDER BY author_id DESC, edition_id DESC LIMIT 100
3
```

Data Output | Messages | Notifications

	author_id [PK] integer	edition_id [PK] integer
1	100004	4068
2	100004	3049
3	100001	9767
4	100001	8552
5	99989	1913
6	99983	4579
7	99981	4445
8	99953	5199
9	99948	1860
10	99941	8232

SQL запити, які використовувались для генерації:

```
if table_name == "Author":
                # Insert random authors
                for _ in range(num):
                    name = ''.join(random.choice(string.ascii_letters) for _ in
range(5))
                    surname = ''.join(random.choice(string.ascii_letters) for _
in range(7))
                    c.execute('INSERT INTO "Author" ("name","surname") VALUES
(%s,%s)', (name, surname))

            elif table_name == "Collection":
                for _ in range(num):
                    col_name = ''.join(random.choice(string.ascii_letters) for _
in range(8))
                    col_type = random.choice(["Literature", "Fiction", "Poetry",
"Science"])
                    col_view = random.choice(["Paper", "Electronic"])
                    c.execute('INSERT INTO "Collection" ("name","type","view")
VALUES (%s,%s,%s)', (col_name, col_type, col_view))

            elif table_name == "Edition":
                for _ in range(num):
                    ed_name = ''.join(random.choice(string.ascii_letters) for _
in range(6))
                    branch = random.choice(["Fiction", "Poetry", "Non-Fiction",
"Science"])
                    pages = random.randint(50, 1000)
                    languages = random.choice(["English", "Ukrainian", "French"])
                    c.execute('INSERT INTO "Edition"
("name","branch","number_of_pages","languages") VALUES (%s,%s,%s,%s)', (ed_name,
branch, pages, languages))

            elif table_name == "Author_Collection_Edition":
                # For linking table, we need existing authors and editions
                c.execute('SELECT "author_id" FROM "Author"')
                authors = [row[0] for row in c.fetchall()]
                c.execute('SELECT "edition_id" FROM "Edition"')
                editions = [row[0] for row in c.fetchall()]
                if not authors or not editions:
                    self.view.show_message("No authors or editions to link.
Generate them first.")
                    return
                for _ in range(num):
                    aid = random.choice(authors)
                    eid = random.choice(editions)
                    try:
                        c.execute('INSERT INTO "Author_Collection_Edition"
("author_id","edition_id") VALUES (%s,%s)', (aid, eid))
                    except psycopg2.errors.UniqueViolation:
```

```
                        self.conn.rollback()  # ignore duplicates
                        continue

            elif table_name == "Author_Collection_Edition_ED":
                # We need existing editions and collections
                c.execute('SELECT "edition_id" FROM "Edition"')
                editions = [row[0] for row in c.fetchall()]
                c.execute('SELECT "collection_id" FROM "Collection"')
                collections = [row[0] for row in c.fetchall()]
                if not editions or not collections:
                    self.view.show_message("No editions or collections to link.
Generate them first.")
                    return
                for _ in range(num):
                    eid = random.choice(editions)
                    cid = random.choice(collections)
                    # Random date
                    year = random.randint(1800, 2020)
                    month = random.randint(1,12)
                    day = random.randint(1,28)
                    date_str = f"{year}-{month:02d}-{day:02d}"
                    try:
                        c.execute('INSERT INTO "Author_Collection_Edition_ED"
("edition_id","collection_id","date") VALUES (%s,%s,%s)', (eid, cid, date_str))
                    except psycopg2.errors.UniqueViolation:
                        self.conn.rollback()
                        continue
```

Перейдемо до SQL запиту.

Я вирішив реалізувати запит, який виконуватиме пошук за певною комбінацією літер в прізвищі автора:

```
def advanced_search(self, surname_pattern, min_pages, max_pages, start_date,
end_date):
        # Example advanced query
        # Joins across multiple tables and applies filters on numeric range,
string pattern, and date range
        c = self.conn.cursor()
        sql = """
            SELECT a.name, a.surname, ed.name as edition_name,
ed.number_of_pages, aced.date, c.name as collection_name
            FROM "Author" a
            JOIN "Author_Collection_Edition" ace ON a.author_id = ace.author_id
            JOIN "Edition" ed ON ace.edition_id = ed.edition_id
            JOIN "Author_Collection_Edition_ED" aced ON ed.edition_id =
aced.edition_id
            JOIN "Collection" c ON aced.collection_id = c.collection_id
```

```
            WHERE a.surname ILIKE %s
              AND ed.number_of_pages BETWEEN %s AND %s
              AND aced.date BETWEEN %s AND %s
            ORDER BY a.surname, ed.number_of_pages;
        """
        c.execute(sql, (f"%{surname_pattern}%", min_pages, max_pages, start_date,
end_date))
        rows = c.fetchall()
        return rows
```

Для цього було згенеровано дані для кожної таблиці, після чого вводячи уточнюючі дані для пошукового запиту отримую відповідні дані, які мають записи в кожній з таблиць:

```
Welcome! Which table do you want to work with?
1. Author
2. Collection
3. Edition
4. Author_Collection_Edition
5. Author_Collection_Edition_ED
6. Advanced Search
7. Quit
Enter your choice: 6
Enter surname pattern (e.g., part of surname): ZZ
Enter minimum number_of_pages: 1
Enter maximum number_of_pages: 500
Enter start date (YYYY-MM-DD): 1900-01-01
Enter end date (YYYY-MM-DD): 2024-01-01
Advanced Search Results:
+-------------+----------------+--------------+-------+------------+-----------------+
| Author Name | Author Surname | Edition Name | Pages |    Date    | Collection Name |
+-------------+----------------+--------------+-------+------------+-----------------+
|    PzQNe    |    aFJiZZY    |    ZMgzss    |  135  | 1968-04-05 |    XEolFBIr     |
|    uNhAK    |    bRwuzzc    |    abYpuz    |  267  | 1991-07-21 |    uheJdzfr     |
|    sarSU    |    JItZzMg    |    YFqfNA    |  155  | 2004-05-13 |    KZvwfWaC     |
|    yvWam    |    jzZQINk    |    eydlqX    |  383  | 1930-06-05 |    fwGwBObJ     |
|    yvWam    |    jzZQINk    |    eydlqX    |  383  | 1978-01-09 |    ogHFpFwZ     |
|    yvWam    |    jzZQINk    |    eydlqX    |  383  | 1971-05-28 |    IesZMCex     |
|    bbMdG    |    OVZzLVm    |    HZrxmP    |  189  | 1923-08-26 |    wRmeunAk     |
|    DQUiy    |    oZZlWGyT    |    WuDHis    |  296  | 2012-08-24 |    bkkGFuGw     |
|    XQUhI    |    VMQCZzh    |    abnhuy    |  163  | 1985-01-21 |    aXPpjuSw     |
|    xYBwD    |    xcrUGZz    |    ymGvCq    |  435  | 1976-01-17 |    EwPvAlLm     |
|    Dxkhl    |    ZZLqPmC    |    uNgooL    |  218  | 1998-02-09 |    WEGOjsOe     |
|    Dxkhl    |    ZZLqPmC    |    uNgooL    |  218  | 1998-04-28 |    AVbEejgp     |
|    Dxkhl    |    ZZLqPmC    |    ttelDH    |  290  | 1986-12-16 |    fouMbwJz     |
|    oIFir    |    zzmIaLr    |    LKzbDN    |  303  | 1941-10-01 |    pQhCOelb     |
|    bsnNj    |    ZZolQMh    |    jLimjG    |  356  | 1907-07-15 |    GecFakXU     |
|    OApsP    |    ZZrDeBf    |    LHsYvk    |  249  | 1945-03-26 |    WHWIcUer     |
|    OApsP    |    ZZrDeBf    |    LHsYvk    |  249  | 1960-03-17 |    muzDVgSF     |
|    OApsP    |    ZZrDeBf    |    LHsYvk    |  249  | 1987-04-18 |    mybsbApd     |
|    HkyWL    |    ZzubBCV    |    DUmcno    |  196  | 1924-09-12 |    bhGsmbek     |
|    jyGHE    |    zzZCSJX    |    LzjJgD    |  321  | 2010-07-16 |    qggAKnNR     |
+-------------+----------------+--------------+-------+------------+-----------------+
```

Код програми

model.py

```
import psycopg2
from view import View
import random
import string


class Model:
    def __init__(self):
        self.conn = psycopg2.connect(
            dbname='science_pub',
            user='postgres',
            password='1111',
            host='localhost',
            port='5432'
        )
        self.view = View()
        # Create tables if not exist
        self.create_tables()

    def create_tables(self):
        c = self.conn.cursor()
        # Check and create Author table
        c.execute("""
            CREATE TABLE IF NOT EXISTS "Author" (
                "Author_ID" SERIAL PRIMARY KEY,
                "name" VARCHAR NOT NULL,
                "surname" VARCHAR NOT NULL
            );
        """)

        # Check and create Collection table
        c.execute("""
            CREATE TABLE IF NOT EXISTS "Collection" (
                "Collection_ID" SERIAL PRIMARY KEY,
                "name" VARCHAR NOT NULL,
                "type" VARCHAR NOT NULL,
                "view" VARCHAR NOT NULL
            );
        """)

        # Check and create Edition table
        c.execute("""
            CREATE TABLE IF NOT EXISTS "Edition" (
                "Edition_ID" SERIAL PRIMARY KEY,
                "name" VARCHAR NOT NULL,
                "branch" VARCHAR NOT NULL,
```

```
                "number_of_pages" INT NOT NULL,
                "languages" VARCHAR NOT NULL
            );
        """)

        # Check and create Author_Collection_Edition table
        c.execute("""
            CREATE TABLE IF NOT EXISTS "Author_Collection_Edition" (
                "Author_ID" INT NOT NULL,
                "Edition_ID" INT NOT NULL,
                PRIMARY KEY ("Author_ID", "Edition_ID"),
                FOREIGN KEY ("Author_ID") REFERENCES "Author" ("Author_ID") ON
DELETE CASCADE,
                FOREIGN KEY ("Edition_ID") REFERENCES "Edition" ("Edition_ID") ON
DELETE CASCADE
            );
        """)

        # Check and create Author_Collection_Edition_ED table
        c.execute("""
            CREATE TABLE IF NOT EXISTS "Author_Collection_Edition_ED" (
                "Edition_ID" INT NOT NULL,
                "Collection_ID" INT NOT NULL,
                "Date" DATE,
                PRIMARY KEY ("Edition_ID", "Collection_ID"),
                FOREIGN KEY ("Edition_ID") REFERENCES "Edition" ("Edition_ID") ON
DELETE CASCADE,
                FOREIGN KEY ("Collection_ID") REFERENCES "Collection"
("Collection_ID") ON DELETE CASCADE
            );
        """)

        self.conn.commit()

    def get_all(self, table_name):
        c = self.conn.cursor()
        c.execute(f'SELECT * FROM "{table_name}"')
        return c.fetchall()

    def add_data(self, table_name, data, pk_columns=None):
        try:
            c = self.conn.cursor()
            columns = ', '.join(data.keys())
            placeholders = ', '.join(['%s'] * len(data))
            sql = f'INSERT INTO "{table_name}" ({columns}) VALUES
({placeholders});'
            c.execute(sql, list(data.values()))
            self.conn.commit()
            self.view.show_message("Added successfully!")
        except psycopg2.errors.ForeignKeyViolation as e:
            self.conn.rollback()
```

```
            self.view.show_message(f"ERROR: Foreign key violation.\n{e}")
        except psycopg2.errors.UniqueViolation as e:
            self.conn.rollback()
            self.view.show_message(f"ERROR: Duplicate key.\n{e}")

    def update_data(self, table_name, data, condition_column, condition_value):
        try:
            if not table_name or not data or not condition_column or
condition_value is None:
                self.view.show_message("Insufficient information to update
data.")
                return
            c = self.conn.cursor()
            set_clause = ", ".join([f"\"{key}\" = %s" for key in data.keys()])
            sql = f'UPDATE "{table_name}" SET {set_clause} WHERE
"{condition_column}" = %s;'
            values = list(data.values())
            values.append(condition_value)
            c.execute(sql, values)
            self.conn.commit()
            self.view.show_message("Updated successfully!")
        except psycopg2.Error as e:
            self.conn.rollback()
            self.view.show_message(str(e))

    def delete_data(self, table_name, cond, val):
        try:
            c = self.conn.cursor()
            sql = f'DELETE FROM "{table_name}" WHERE "{cond}" = %s;'
            c.execute(sql, (val,))
            self.conn.commit()
            self.view.show_message("Deleted successfully!")
        except psycopg2.Error as e:
            self.conn.rollback()
            self.view.show_message(str(e))

    def delete_data_composite(self, table_name, conditions):
        # conditions: list of tuples (cond, val)
        try:
            c = self.conn.cursor()
            where_clause = ' AND '.join([f'"{cond}" = %s' for cond, _ in
conditions])
            values = [val for _, val in conditions]
            sql = f'DELETE FROM "{table_name}" WHERE {where_clause};'
            c.execute(sql, values)
            self.conn.commit()
            self.view.show_message("Deleted successfully!")
        except psycopg2.Error as e:
            self.conn.rollback()
            self.view.show_message(str(e))
```

```
    def generate_data(self, table_name, num):
        # Example random data generation
        c = self.conn.cursor()
        try:
            if table_name == "Author":
                # Insert random authors
                for _ in range(num):
                    name = ''.join(random.choice(string.ascii_letters) for _ in
range(5))
                    surname = ''.join(random.choice(string.ascii_letters) for _
in range(7))
                    c.execute('INSERT INTO "Author" ("name","surname") VALUES
(%s,%s)', (name, surname))

            elif table_name == "Collection":
                for _ in range(num):
                    col_name = ''.join(random.choice(string.ascii_letters) for _
in range(8))
                    col_type = random.choice(["Literature", "Fiction", "Poetry",
"Science"])
                    col_view = random.choice(["Paper", "Electronic"])
                    c.execute('INSERT INTO "Collection" ("name","type","view")
VALUES (%s,%s,%s)', (col_name, col_type, col_view))

            elif table_name == "Edition":
                for _ in range(num):
                    ed_name = ''.join(random.choice(string.ascii_letters) for _
in range(6))
                    branch = random.choice(["Fiction", "Poetry", "Non-Fiction",
"Science"])
                    pages = random.randint(50, 1000)
                    languages = random.choice(["English", "Ukrainian", "French"])
                    c.execute('INSERT INTO "Edition"
("name","branch","number_of_pages","languages") VALUES (%s,%s,%s,%s)', (ed_name,
branch, pages, languages))

            elif table_name == "Author_Collection_Edition":
                # For linking table, we need existing authors and editions
                c.execute('SELECT "Author_ID" FROM "Author"')
                authors = [row[0] for row in c.fetchall()]
                c.execute('SELECT "Edition_ID" FROM "Edition"')
                editions = [row[0] for row in c.fetchall()]
                if not authors or not editions:
                    self.view.show_message("No authors or editions to link.
Generate them first.")
                    return
                for _ in range(num):
                    aid = random.choice(authors)
                    eid = random.choice(editions)
                    try:
```

```
                        c.execute('INSERT INTO "Author_Collection_Edition"
("Author_ID","Edition_ID") VALUES (%s,%s)', (aid, eid))
                    except psycopg2.errors.UniqueViolation:
                        self.conn.rollback()  # ignore duplicates
                        continue

            elif table_name == "Author_Collection_Edition_ED":
                # We need existing editions and collections
                c.execute('SELECT "Edition_ID" FROM "Edition"')
                editions = [row[0] for row in c.fetchall()]
                c.execute('SELECT "Collection_ID" FROM "Collection"')
                collections = [row[0] for row in c.fetchall()]
                if not editions or not collections:
                    self.view.show_message("No editions or collections to link.
Generate them first.")
                    return
                for _ in range(num):
                    eid = random.choice(editions)
                    cid = random.choice(collections)
                    # Random date
                    year = random.randint(1800, 2020)
                    month = random.randint(1,12)
                    day = random.randint(1,28)
                    date_str = f"{year}-{month:02d}-{day:02d}"
                    try:
                        c.execute('INSERT INTO "Author_Collection_Edition_ED"
("Edition_ID","Collection_ID","Date") VALUES (%s,%s,%s)', (eid, cid, date_str))
                    except psycopg2.errors.UniqueViolation:
                        self.conn.rollback()
                        continue

            self.conn.commit()
            self.view.show_message("Generated successfully!")
        except psycopg2.Error as e:
            self.conn.rollback()
            self.view.show_message(str(e))

    def advanced_search(self, surname_pattern, min_pages, max_pages, start_date,
end_date):
        # Example advanced query
        # Joins across multiple tables and applies filters on numeric range,
string pattern, and date range
        c = self.conn.cursor()
        sql = """
            SELECT a."name", a."surname", ed."name" as edition_name,
ed."number_of_pages", aced."Date", c."name" as collection_name
            FROM "Author" a
            JOIN "Author_Collection_Edition" ace ON a."Author_ID" =
ace."Author_ID"
            JOIN "Edition" ed ON ace."Edition_ID" = ed."Edition_ID"
```

```
            JOIN "Author_Collection_Edition_ED" aced ON ed."Edition_ID" =
aced."Edition_ID"
            JOIN "Collection" c ON aced."Collection_ID" = c."Collection_ID"
            WHERE a."surname" ILIKE %s
              AND ed."number_of_pages" BETWEEN %s AND %s
              AND aced."Date" BETWEEN %s AND %s
            ORDER BY a."surname", ed."number_of_pages";
        """
        c.execute(sql, (f"%{surname_pattern}%", min_pages, max_pages, start_date,
end_date))
        rows = c.fetchall()
        return rows
```

view.py

```
from tabulate import tabulate

class View:
    def show_author(self, data):
        print("Authors:")
        headers = ["Author_ID", "Name", "Surname"]
        print(tabulate(data, headers, tablefmt="pretty"))

    def show_collection(self, data):
        print("Collections:")
        headers = ["Collection_ID", "Name", "Type", "View"]
        print(tabulate(data, headers, tablefmt="pretty"))

    def show_edition(self, data):
        print("Editions:")
        headers = ["Edition_ID", "Name", "Branch", "Number_of_pages",
"Languages"]
        print(tabulate(data, headers, tablefmt="pretty"))

    def show_author_collection_edition(self, data):
        print("Author_Collection_Edition:")
        headers = ["Author_ID", "Edition_ID"]
        print(tabulate(data, headers, tablefmt="pretty"))

    def show_author_collection_edition_ed(self, data):
        print("Author_Collection_Edition_ED:")
        headers = ["Edition_ID", "Collection_ID", "Date"]
        print(tabulate(data, headers, tablefmt="pretty"))

    def show_message(self, message):
        print(message)

    def show_main_menu(self):
        self.show_message("\nWelcome! Which table do you want to work with?")
```

```
        self.show_message("1. Author")
        self.show_message("2. Collection")
        self.show_message("3. Edition")
        self.show_message("4. Author_Collection_Edition")
        self.show_message("5. Author_Collection_Edition_ED")
        self.show_message("6. Advanced Search")
        self.show_message("7. Quit")
        return input("Enter your choice: ")

    def show_table_menu(self, table_name):
        self.show_message(f"\nWorking with {table_name}")
        self.show_message("1. Add")
        self.show_message("2. View")
        if table_name != "Author_Collection_Edition" and table_name !=
"Author_Collection_Edition_ED":
            self.show_message("3. Update")
            self.show_message("4. Delete")
            self.show_message("5. Generate data")
            self.show_message("6. Back")
        else:
            # Linking tables might be add/view/delete/generate only
            self.show_message("3. Delete")
            self.show_message("4. Generate data")
            self.show_message("5. Back")
        return input("Enter your choice: ")

    def get_data_input(self, table_name):
        if table_name == "Author":
            name = input("Enter author's name: ")
            surname = input("Enter author's surname: ")
            return {"name": name, "surname": surname}
        elif table_name == "Collection":
            cname = input("Enter collection's name: ")
            ctype = input("Enter collection's type: ")
            cview = input("Enter collection's view: ")
            return {"name": cname, "type": ctype, "view": cview}
        elif table_name == "Edition":
            ename = input("Enter edition's name: ")
            branch = input("Enter edition's branch: ")
            pages = int(input("Enter number_of_pages: "))
            langs = input("Enter languages: ")
            return {"name": ename, "branch": branch, "number_of_pages": pages,
"languages": langs}
        elif table_name == "Author_Collection_Edition":
            aid = int(input("Enter Author_ID: "))
            eid = int(input("Enter Edition_ID: "))
            return {"Author_ID": aid, "Edition_ID": eid}
        elif table_name == "Author_Collection_Edition_ED":
            eid = int(input("Enter Edition_ID: "))
            cid = int(input("Enter Collection_ID: "))
            date = input("Enter Date (YYYY-MM-DD): ")
```

```
            return {"Edition_ID": eid, "Collection_ID": cid, "Date": date}

    def get_update_input(self, table_name, pk_val):
        if table_name == "Author":
            name = input("Enter new author's name: ")
            surname = input("Enter new author's surname: ")
            return {"name": name, "surname": surname}
        elif table_name == "Collection":
            cname = input("Enter new collection's name: ")
            ctype = input("Enter new collection's type: ")
            cview = input("Enter new collection's view: ")
            return {"name": cname, "type": ctype, "view": cview}
        elif table_name == "Edition":
            ename = input("Enter new edition's name: ")
            branch = input("Enter new branch: ")
            pages = int(input("Enter new number_of_pages: "))
            langs = input("Enter new languages: ")
            return {"name": ename, "branch": branch, "number_of_pages": pages,
"languages": langs}

    def get_pk(self, table_name):
        if table_name == "Author":
            val = int(input("Enter Author_ID: "))
            return "Author_ID", val
        elif table_name == "Collection":
            val = int(input("Enter Collection_ID: "))
            return "Collection_ID", val
        elif table_name == "Edition":
            val = int(input("Enter Edition_ID: "))
            return "Edition_ID", val

    def get_pk_composite(self, table_name):
        if table_name == "Author_Collection_Edition":
            aid = int(input("Enter Author_ID: "))
            eid = int(input("Enter Edition_ID: "))
            return [("Author_ID", aid), ("Edition_ID", eid)]
        elif table_name == "Author_Collection_Edition_ED":
            eid = int(input("Enter Edition_ID: "))
            cid = int(input("Enter Collection_ID: "))
            return [("Edition_ID", eid), ("Collection_ID", cid)]

    def get_num(self):
        return int(input("Enter number of data to generate: "))

    def advanced_search_input(self):
        surname_pattern = input("Enter surname pattern (e.g. part of surname): ")
        min_pages = int(input("Enter minimum number_of_pages: "))
        max_pages = int(input("Enter maximum number_of_pages: "))
        start_date = input("Enter start date (YYYY-MM-DD): ")
        end_date = input("Enter end date (YYYY-MM-DD): ")
        return surname_pattern, min_pages, max_pages, start_date, end_date
```

```
    def show_advanced_search_results(self, data):
        print("Advanced Search Results:")
        headers = ["Author Name", "Author Surname", "Edition Name", "Pages",
"Date", "Collection Name"]
        print(tabulate(data, headers, tablefmt="pretty"))
```

controller.py

```
from model import Model
from view import View

class Controller:
    def __init__(self):
        self.model = Model()
        self.view = View()

    def run(self):
        while True:
            choice = self.view.show_main_menu()
            if choice == '1':
                self.handle_table("Author")
            elif choice == '2':
                self.handle_table("Collection")
            elif choice == '3':
                self.handle_table("Edition")
            elif choice == '4':
                self.handle_table("Author_Collection_Edition")
            elif choice == '5':
                self.handle_table("Author_Collection_Edition_ED")
            elif choice == '6':
                self.advanced_search()
            elif choice == '7':
                break

    def handle_table(self, table_name):
        while True:
            choice = self.view.show_table_menu(table_name)
            if table_name in ["Author", "Collection", "Edition"]:
                if choice == '1':
                    self.add(table_name)
                elif choice == '2':
                    self.view_table(table_name)
                elif choice == '3':
                    self.update(table_name)
                elif choice == '4':
                    self.delete(table_name)
                elif choice == '5':
                    self.generate_data(table_name)
```

```
                elif choice == '6':
                    break
            else:
                # For linking tables (Author_Collection_Edition,
Author_Collection_Edition_ED)
                if choice == '1':
                    self.add(table_name)
                elif choice == '2':
                    self.view_table(table_name)
                elif choice == '3':
                    self.delete(table_name, composite=True)
                elif choice == '4':
                    self.generate_data(table_name)
                elif choice == '5':
                    break

    def add(self, table_name):
        data = self.view.get_data_input(table_name)
        self.model.add_data(table_name, data)

    def view_table(self, table_name):
        data = self.model.get_all(table_name)
        if table_name == "Author":
            self.view.show_author(data)
        elif table_name == "Collection":
            self.view.show_collection(data)
        elif table_name == "Edition":
            self.view.show_edition(data)
        elif table_name == "Author_Collection_Edition":
            self.view.show_author_collection_edition(data)
        elif table_name == "Author_Collection_Edition_ED":
            self.view.show_author_collection_edition_ed(data)

    def update(self, table_name):
        pk, val = self.view.get_pk(table_name)
        new_data = self.view.get_update_input(table_name, val)
        self.model.update_data(table_name, new_data, pk, val)

    def delete(self, table_name, composite=False):
        if composite:
            conditions = self.view.get_pk_composite(table_name)
            self.model.delete_data_composite(table_name, conditions)
        else:
            pk, val = self.view.get_pk(table_name)
            self.model.delete_data(table_name, pk, val)

    def generate_data(self, table_name):
        num = self.view.get_num()
        self.model.generate_data(table_name, num)

    def advanced_search(self):
```

```
        surname_pattern, min_pages, max_pages, start_date, end_date =
self.view.advanced_search_input()
        data = self.model.advanced_search(surname_pattern, min_pages, max_pages,
start_date, end_date)
        self.view.show_advanced_search_results(data)
```

main.py

```
from controller import Controller

if __name__ == "__main__":
    controller = Controller()
    controller.run()
```