НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
Кафедра програмного забезпечення та комп'ютерних систем

Лабораторна робота №2

з дисципліни

«Бази даних і засоби управління»

Виконала: студентка III курсу
ФПМ групи КП-83
Сергійчук Сергій Миколайович

Київ – 2019

Ознайомлення з базовими операціями СУБД PostgreSQL

Мета роботи: здобуття практичних навичок проектування та побудови реляційних баз даних та створення прикладних програм з базами даних

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі No1, 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).

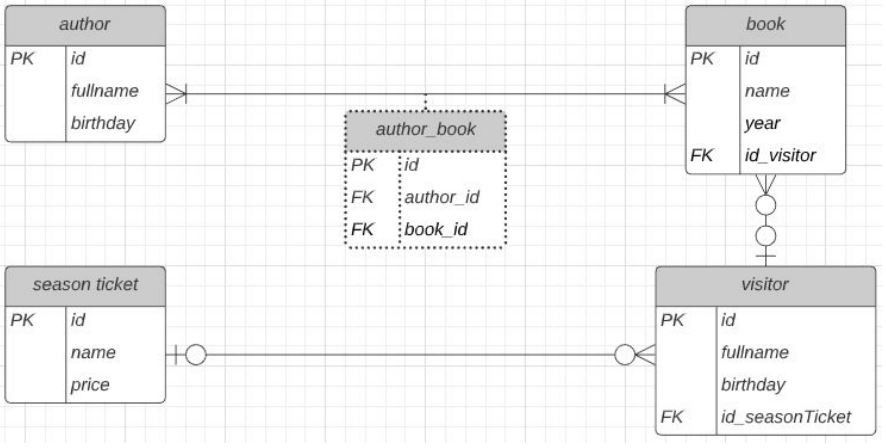
Опис предметної галузі

При проектуванні бази даних «Бібліотека» можна виділити такі сутності: Книга (book), Автор (author), Абонемент(season ticket), Відвідувач(visitor).

Абонементи є декількох видів.Різні користувачі можуть мати однакові абонементи. Один користувач не може одночасно мати більше 1 абонемента.(один до багатьох).

Один автор може написати декілька книг. В однієї книги може бути декілька авторів.(багато до багатьох).

Концептуальна модель учбової предметної області "Бібліотека"



Опис програми

Програма створена за патерном MVC (Model-View-Controller). Складається відповідно з модулів model , view та controller.

У модулі model реалізовані функції , що здійснюють SQL запити до Бази Даних.

У модулі view реалізовані функції виводу даних з таблиць.

У класі Controller реалізовані функції для відповідних меню та допоміжні функції.

Опис структури меню програми

Меню програми можна розглядати як її концептуальну модель

Лістинг програми

```
from controller import Controller
Controller().show_init_menu()
```

Model

```
import psycopg2


class Model:

    def __init__(self):
        try:
            self.connection = psycopg2.connect(dbname='postgres',
user='postgres',
                                               password='1',
host='localhost')
            self.cursor = self.connection.cursor()
        except (Exception, psycopg2.Error) as error:
            print("Error during connection to PostgreSQL", error)


    def get_col_names(self):
        return [d[0] for d in self.cursor.description]

    def set_data_to_db(self, reset):
        if reset:
            f = open("db.txt", "r")
            self.cursor.execute(f.read())
            self.connection.commit()

    def get(self, t_name, condition):
        try:
            query = f'SELECT * FROM {t_name}'

            if condition:
                query += ' WHERE ' + condition

            self.cursor.execute(query)
            return self.get_col_names(), self.cursor.fetchall()
        finally:
            self.connection.commit()

    def insert(self, t_name, columns, values):
        try:
            query = f'INSERT INTO {t_name} ({columns}) VALUES ({values});'

            self.cursor.execute(query)
        finally:
            self.connection.commit()

    def delete(self, t_name, condition):
        try:
```

```
            query = f'DELETE FROM {t_name} WHERE {condition};'

            self.cursor.execute(query)
        finally:
            self.connection.commit()

    def update(self, t_name, condition, statement):
        try:
            query = f'UPDATE {t_name} SET {statement} WHERE {condition}'

            self.cursor.execute(query)
        finally:
            self.connection.commit()

    def search_book_by_author(self, author_id):
        try:

            query = f'''
            SELECT * from book
            WHERE id in(
                SELECT book_id FROM author_book
                WHERE author_id =  {author_id}
            );'''
            self.cursor.execute(query)
            return self.get_col_names(), self.cursor.fetchall()
        finally:
            self.connection.commit()

    def search_author_by_book(self, book_id):

        try:
            query = f'''
                    SELECT * from author
                    WHERE id in(
                        SELECT author_id FROM author_book
                        WHERE book_id =  {book_id}
                        );'''
            self.cursor.execute(query)
            return self.get_col_names(), self.cursor.fetchall()
        finally:
            self.connection.commit()

    def fill_book_by_random_data(self):
        sql = """
        CREATE OR REPLACE FUNCTION randomDepartments()
            RETURNS void AS $$
        DECLARE
            step integer  := 0;
        BEGIN
            LOOP EXIT WHEN step > 10;
                INSERT INTO book (name, year)
                VALUES (
                    substring(md5(random()::text), 1, 10),
                    (random() * (2000 - 1800) + 1800)::integer
                );
                step := step + 1;
```

```
            END LOOP ;
        END;
        $$ LANGUAGE PLPGSQL;
        SELECT randomDepartments();
        """
        try:
            self.cursor.execute(sql)
        finally:
            self.connection.commit()
```

View

```
class View:
    def print(self, data):
        titles, rows = data
        line_len = 30 * len(titles)

        self.print_separator(line_len)
        self.print_row(titles)
        self.print_separator(line_len)

        for row in rows:
            self.print_row(row)
        self.print_separator(line_len)

    @staticmethod
    def print_row(row):
        for col in row:
            print(str(col).rjust(26, ' ') + '   |', end='')
        print('')

    @staticmethod
    def print_separator(length):
        print('-' * length)
```

Controller

```
from consolemenu import SelectionMenu

from model import Model
from view import View

TABLES_NAMES = ['author', 'book', 'author_book', 'visitor',
'season_ticket']
TABLES = {
    'author': ['id', 'fullname', 'birthday'],
    'book': ['id', 'name', 'year'],
    'author_book': ['id', 'author_id', 'book_id'],
    'visitor': ['id', 'fullname', 'birthday', 'id_season_ticket'],
    'season_ticket': ['id', 'name', 'price']
}
```

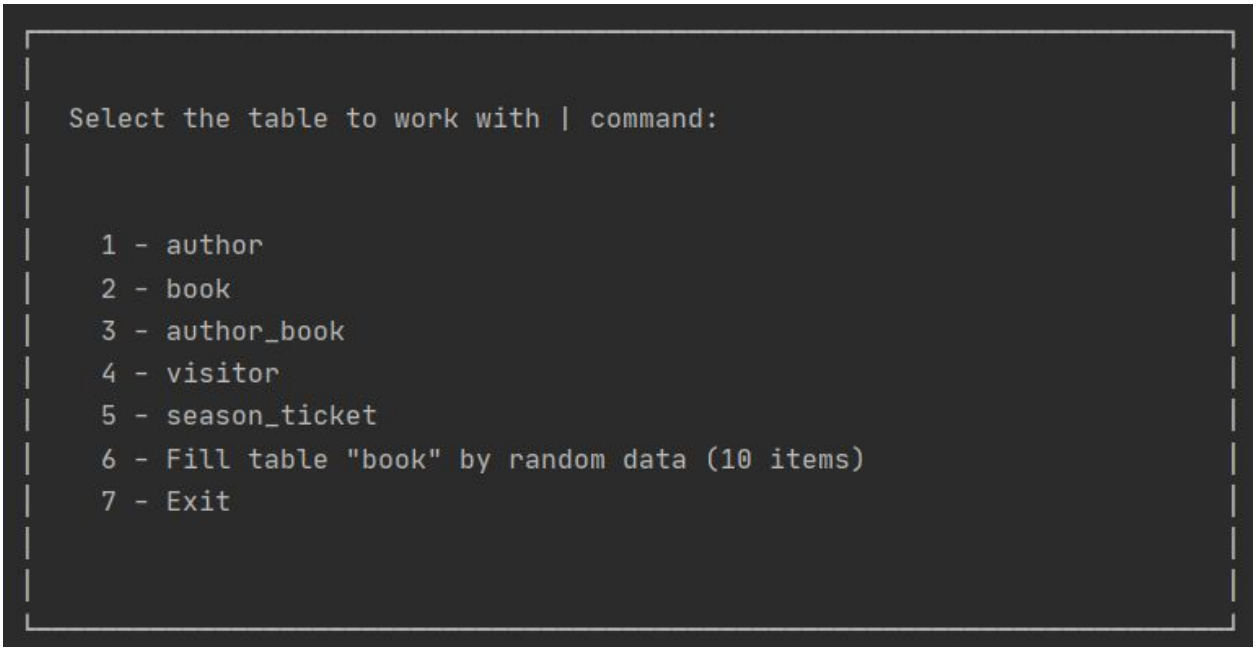
```
def get_input(msg, table_name=''):
    print(msg)
    if table_name:
        print(' | '.join(TABLES[table_name]), end='\n\n')
    return input()


def get_insert_input(msg, table_name):
    print(msg)

    print(' | '.join(TABLES[table_name]), end='\n\n')
    return input(), input()


def press_enter():
    input()


class Controller:

    def __init__(self):
        self.model = Model()
        self.view = View()
        self.model.set_data_to_db(1)

    def show_init_menu(self, msg=''):
        selection_menu = SelectionMenu(
            TABLES_NAMES + ['Fill table "book" by random data (10 items)'],
            title='Select the table to work with | command:', subtitle=msg)

        selection_menu.show()

        index = selection_menu.selected_option
        if index < len(TABLES_NAMES):
            table_name = TABLES_NAMES[index]
            self.show_entity_menu(table_name)
        elif index == 5:
            self.fill_by_random()
        else:
            print('Bye, have a beautiful day!')

    def show_entity_menu(self, table_name, msg=''):
        options = ['Get', 'Delete', 'Update', 'Insert']

        functions = [self.get, self.delete, self.update, self.insert]

        if table_name == 'author':
            options.append('Search book by author')
            functions.append(self.search_book_by_author)
        elif table_name == 'book':
            options.append('Search authors that wrote the book')
            functions.append(self.search_author_by_book)
```

```
        selection_menu = SelectionMenu(options, f'Name of table:
{table_name}', exit_option_text='Back', subtitle=msg)
        selection_menu.show()
        try:
            function = functions[selection_menu.selected_option]
            function(table_name)
        except IndexError:
            self.show_init_menu()

    def get(self, table_name):
        try:
            condition = get_input(
                f'GET {table_name}\nEnter condition (SQL) or leave empty:',
table_name)
            data = self.model.get(table_name, condition)
            self.view.print(data)
            press_enter()
            self.show_entity_menu(table_name)
        except Exception as err:
            self.show_entity_menu(table_name, str(err))

    def insert(self, table_name):
        try:
            columns, values = get_insert_input(
                f"INSERT {table_name}\nEnter columns divided with commas,
then do the same for values in format: ['value1', 'value2', ...]",
                table_name)

            self.model.insert(table_name, columns, values)
            self.show_entity_menu(table_name, 'Insert is successful!')
        except Exception as err:
            self.show_entity_menu(table_name, str(err))

    def delete(self, table_name):
        try:
            condition = get_input(
                f'DELETE {table_name}\n Enter condition (SQL):',
table_name)

            self.model.delete(table_name, condition)
            self.show_entity_menu(table_name, 'Delete is successful')
        except Exception as err:
            self.show_entity_menu(table_name, str(err))

    def update(self, table_name):
        try:
            condition = get_input(
                f'UPDATE {table_name}\nEnter condition (SQL):', table_name)

            statement = get_input(
                "Enter SQL statement in format [<key>='<value>']",
table_name)

            self.model.update(table_name, condition, statement)
            self.show_entity_menu(table_name, 'Update is successful')
        except Exception as err:
```

```
            self.show_entity_menu(table_name, str(err))

    def search_book_by_author(self, table_name):
        try:
            author_id = get_input(
                'Search book by author\'s id are: \nEnter id divided with
commas:')

            data = self.model.search_book_by_author(author_id)
            self.view.print(data)
            press_enter()
            self.show_entity_menu(table_name)
        except Exception as err:
            self.show_entity_menu(table_name, str(err))

    def search_author_by_book(self, table_name):
        try:
            book_id = get_input('Search authors that wrote the book.\nEnter
book\'s id:')

            data = self.model.search_author_by_book(book_id)
            self.view.print(data)
            press_enter()
            self.show_entity_menu(table_name)
        except Exception as err:
            self.show_entity_menu(table_name, str(err))

    def fill_by_random(self):
        try:
            self.model.fill_book_by_random_data()

            self.show_init_menu('Generated successfully')

        except Exception as err:
            self.show_init_menu(str(err))
```

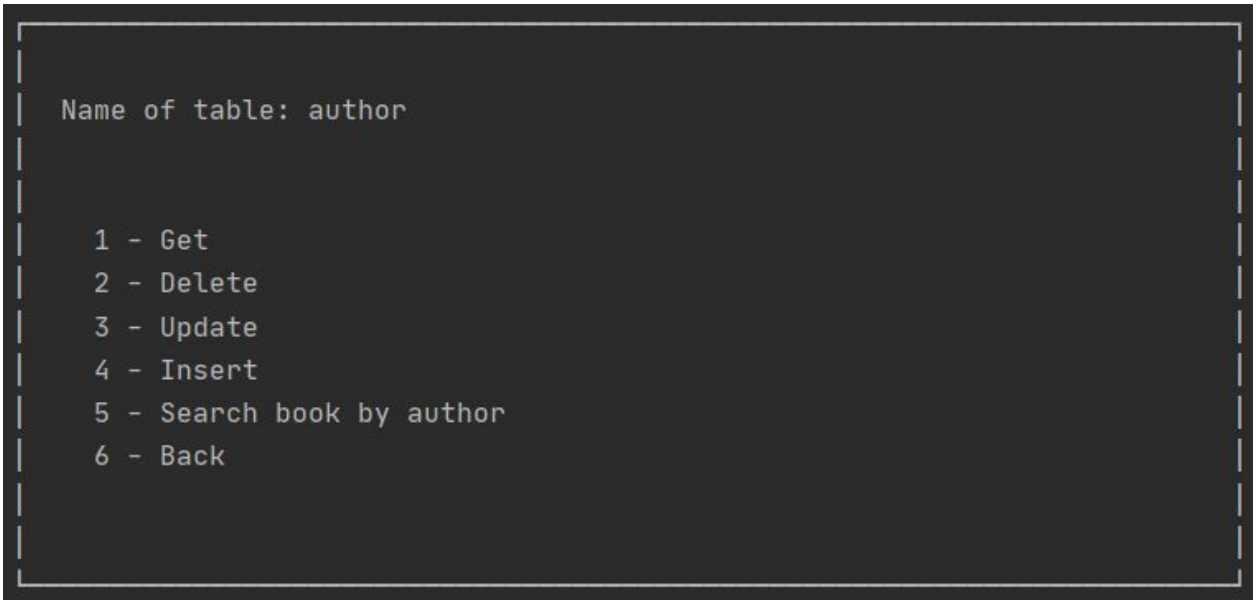
Скріншот результатів виконання операції вилучення

INITIAL MENU



ENTITY MENU

(task)



GET

id	birthday	fullname
1	1965	A.J.Merzlak
2	1970	V.B.Polonskii
3	1959	J.P.Bevz
4	1981	V.J.Bevz
5	1965	J.K.Rowling
6	1859	A.C.Doyle

(author)

(author де id >1)

```
id = 1
```

id	birthday	fullname
1	1965	A.J.Merzlak

INSERT

```
INSERT author
Enter columns divided with commas, then do the same for values in format: ['value1', 'value2', ...]
id | fullname | birthday

fullname
'fullname'
```

```
Name of table: author

Insert is successful!


  1 - Get
  2 - Delete
  3 - Update
  4 - Insert
  5 - Search book by author
  6 - Back
```

Операція вставки успішна.

```
fullname = 'fullname'
-------------------------------------------------------------------------------------
                    id   |                birthday   |                fullname   |
-------------------------------------------------------------------------------------
                     7   |                    None   |                fullname   |
-------------------------------------------------------------------------------------
```

Новий вставлений запис.

UPDATE

(book де id > 1)

```
UPDATE book
Enter condition (SQL):
id | name | year

id=1
Enter SQL statement in format [<key>='<value>']
id | name | year

name='update'
```

```
GET book
Enter condition (SQL) or leave empty:
id | name | year

name='update'
-------------------------------------------------------------------------------------
                    id   |                  name   |                  year   |
-------------------------------------------------------------------------------------
                     1   |                update   |                  2007   |
-------------------------------------------------------------------------------------
```

DELETE

```
DELETE visitor
 Enter condition (SQL):
id | fullname | birthday | id_season_ticket

id=1


   ┌──────────────────────────────────────────────────────────────────────────────┐
   │                                                                              │
   │  Name of table: visitor                                                      │
   │                                                                              │
   │  Delete is successful                                                        │
   │                                                                              │
   │                                                                              │
   │    1 - Get                                                                   │
   │    2 - Delete                                                                │
   │    3 - Update                                                                │
   │    4 - Insert                                                                │
   │    5 - Back                                                                  │
   │                                                                              │
   │                                                                              │
   └──────────────────────────────────────────────────────────────────────────────┘
```

Знайти всі книги за автором:

```
Search book by author's id are:
Enter id divided with commas:
5
-------------------------------------------------------------------------------
                     id   |                    name   |                    year   |
-------------------------------------------------------------------------------
                      3   |             Harry Potter   |                    1997   |
                      4   |       The Casual Vacancy   |                    2012   |
-------------------------------------------------------------------------------
```

Знайти всіх авторів книги:

```
Search authors that wrote the book.
Enter book's id:
1
-------------------------------------------------------------------------------------------
                        id   |                 birthday   |                 fullname   |
-------------------------------------------------------------------------------------------
                         1   |                     1965   |              A.J.Merzlak   |
                         2   |                     1970   |            V.B.Polonskii   |
-------------------------------------------------------------------------------------------
```

Заповнення таблиці рандомізованими значеннями

Висновок

На лабораторній роботі я здобув практичні навички проектування та побудови реляційних баз даних та створення прикладних програм з базами даних