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”*

Виконала: студентка 3 курсу

ФПМ групи КВ-23

Приймак Юліанна Павлівна

Перевірив: Петрашенко А.В.

Київ – 2024

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

***Виконання роботи***

*Сутності предметної області*

- Students | Учні  
Призначення: збереження даних про учнів.  
Атрибути: ім’я, прізвище, по батькові, ідентифікатор учня.

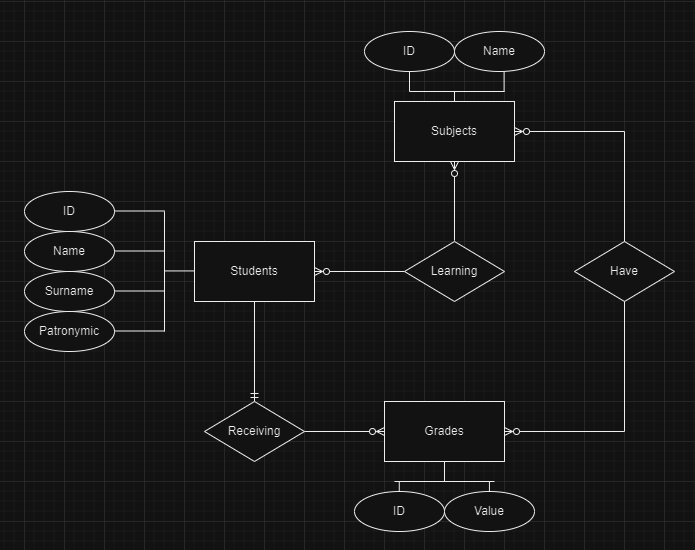
- Subjects | Предмети  
Призначення: збереження даних про предмети.  
Атрибути: назва, ідентифікатор предмета.

- Grades | Оцінки  
Призначення: збереження даних про оцінки.  
Атрибути: значення, ідентифікатор оцінки.

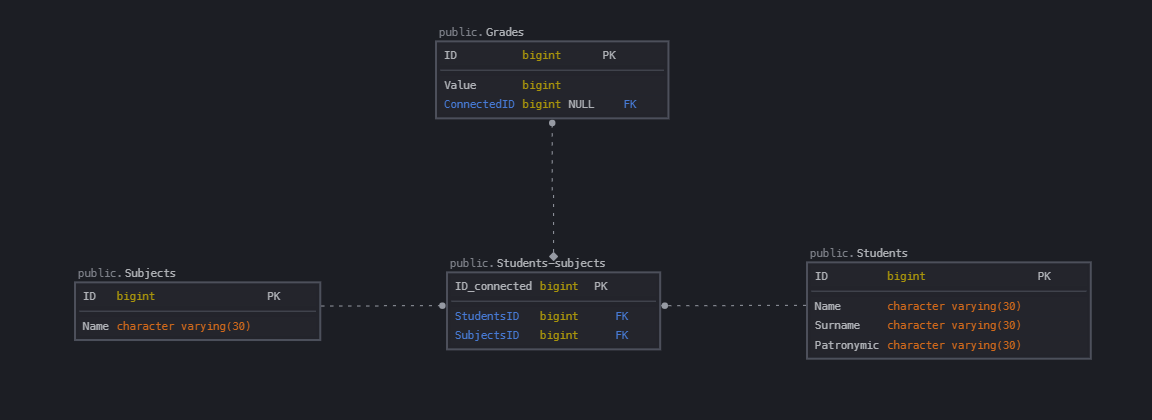
*Зв’язки між сутностями предметної області*

Students – Subjects | Учні – Предмети є зв’язком типу M:N, так як учень може вивчати M предметів, а предмет може вивчатися N студентами.  
Students – Grades | Учні – Оцінки є зв’язком типу 1:N, так як учень може здобути N оцінок, але оцінка може бути здобута лише одним учнем.  
Grades – Subjects | Оцінки – Предмети є зв’язком типу 1:N, так як оцінка може бути здобута по N предметам, але по одному предмету може бути здобута одна оцінка.

*ER-діаграма*



*Логічна модель БД*



*Середовище та компоненти розробки*

Для розробки програми було використано мову програмування Python та середовище розробки PyCharm. Для можливості взаємодії з базою даних PostgreSQL використано бібліотеку psycopg2, що надає API для цього.

***Шаблон проектування***

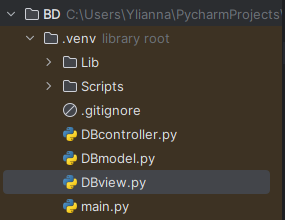
MVC – шаблон проектування, використаний у програмі.

- Model – клас, що є описом логіки для роботи з необхідними даними. Він реалізує всі необхідні операції з даними (додавання, вилучення, оновлення).

- View – клас, що реалізує взаємодію користувача з програмою.

- Controller – клас, що зв’язує користувача і систему, а саме приймає дані введені користувачем, та обробляє їх.

***Структура програми та опис***





Програма розділена на 4 модулі:

main.py – головний файл.

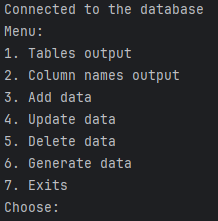
DBmodel – займається регулюванням підключення до бази даних та виконанням запитів.

DBview – виводить результати виконання на екран консолі.

DBcontroller – інтерфейс взаємодії з користувачем.

Як ми бачимо, класи відповідають шаблону проектування MVC.

***Структура меню програми***



1 – Виведення назв таблиць

2 – Виведення назв колонок таблиць

3 – Додавання даних у таблицю

4 – Оновлення даних у таблиці

5 – Видалення даних з таблиці

6 – Генерація даних у таблицю

7 – Завершення програми

***Функції внесення, редагування, видалення, генерування даних***

*Внесення*

def add\_data(self, table\_name, columns, val):  
 c = self.conn.cursor()  
 columns\_str = ', '.join(f'"{col}"' for col in columns)  
 placeholders = ', '.join(['%s'] \* len(val))  
  
 if table\_name == 'Students-subjects':  
 id\_column = 'ID\_connected'  
 else:  
 id\_column = 'ID'  
  
 c.execute(f'SELECT "{id\_column}" FROM "{table\_name}"')  
 existing\_identifiers = c.fetchall()  
 existing\_identifiers = [item for sublist in existing\_identifiers for item in sublist]  
  
 identifier\_index = columns.index(id\_column)  
 val[identifier\_index] = int(val[identifier\_index])  
  
 if val[identifier\_index] in existing\_identifiers:  
 return 2  
  
 if table\_name == 'Students-subjects':  
 c.execute('SELECT "ID" FROM "Students"')  
 student\_ids = [item[0] for item in c.fetchall()]  
 student\_id\_index = columns.index('StudentsID')  
 if int(val[student\_id\_index]) not in student\_ids:  
 return 3  
  
 c.execute('SELECT "ID" FROM "Subjects"')  
 subject\_ids = [item[0] for item in c.fetchall()]  
 subject\_id\_index = columns.index('SubjectsID')  
 if int(val[subject\_id\_index]) not in subject\_ids:  
 return 3  
  
 elif table\_name == 'Grades':  
 c.execute('SELECT "ID\_connected" FROM "Students-subjects"')  
 ss\_ids = [item[0] for item in c.fetchall()]  
 ss\_id\_index = columns.index('ConnectedID')  
 if int(val[ss\_id\_index]) not in ss\_ids:  
 return 3  
  
 c.execute(f'INSERT INTO "{table\_name}" ({columns\_str}) VALUES ({placeholders})', val)  
 self.conn.commit()  
 return 1

*Редагування*

def edit\_data(self, table\_name, column, id, new\_value):  
 c = self.conn.cursor()  
  
 if table\_name == 'Students-subjects':  
 id\_column = 'ID\_connected'  
 else:  
 id\_column = 'ID'  
  
 if column == id\_column:  
 c.execute(f'SELECT "{id\_column}" FROM "{table\_name}"')  
 existing\_identifiers = c.fetchall()  
 existing\_identifiers = [item for sublist in existing\_identifiers for item in sublist]  
 val\_id = int(new\_value)  
 if val\_id in existing\_identifiers:  
 return 2  
  
 elif column in ['StudentsID', 'SubjectsID', 'ConnectedID']:  
 referenced\_table = {  
 'StudentsID': 'Students',  
 'SubjectsID': 'Subjects',  
 'ConnectedID': 'Students-subjects'  
 }[column]  
  
 ref\_id\_column = 'ID\_connected' if referenced\_table == 'Students-subjects' else 'ID'  
 c.execute(f'SELECT "{ref\_id\_column}" FROM "{referenced\_table}"')  
 referenced\_values = [item[0] for item in c.fetchall()]  
 val\_id = int(new\_value)  
 if val\_id not in referenced\_values:  
 return 3  
  
 c.execute(f'UPDATE "{table\_name}" SET "{column}" = %s WHERE "{id\_column}" = %s', (new\_value, id))  
 self.conn.commit()  
 return 1

*Видалення*

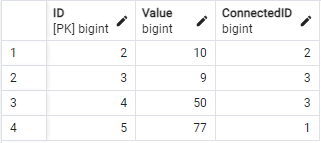
def delete\_data(self, table\_name, id):  
 c = self.conn.cursor()  
  
 try:  
 self.conn.autocommit = False  
  
 if table\_name == 'Students':  
 c.execute('SELECT "ID\_connected" FROM "Students-subjects" WHERE "StudentsID" = %s', (id,))  
 ss\_ids = [row[0] for row in c.fetchall()]  
  
 for ss\_id in ss\_ids:  
 c.execute('DELETE FROM "Grades" WHERE "ConnectedID" = %s', (ss\_id,))  
  
 c.execute('DELETE FROM "Students-subjects" WHERE "StudentsID" = %s', (id,))  
 c.execute('DELETE FROM "Students" WHERE "ID" = %s', (id,))  
  
 elif table\_name == 'Subjects':  
 c.execute('SELECT "ID\_connected" FROM "Students-subjects" WHERE "SubjectsID" = %s', (id,))  
 ss\_ids = [row[0] for row in c.fetchall()]  
  
 for ss\_id in ss\_ids:  
 c.execute('DELETE FROM "Grades" WHERE "ConnectedID" = %s', (ss\_id,))  
  
 c.execute('DELETE FROM "Students-subjects" WHERE "SubjectsID" = %s', (id,))  
 c.execute('DELETE FROM "Subjects" WHERE "ID" = %s', (id,))  
  
 elif table\_name == 'Students-subjects':  
 c.execute('DELETE FROM "Grades" WHERE "ConnectedID" = %s', (id,))  
 c.execute('DELETE FROM "Students-subjects" WHERE "ID\_connected" = %s', (id,))  
  
 elif table\_name == 'Grades':  
 c.execute('DELETE FROM "Grades" WHERE "ID" = %s', (id,))  
  
 self.conn.commit()  
 return 1  
  
 except Exception as e:  
 self.conn.rollback()  
 print(f"Error during deletion: {e}")  
 return 0  
 finally:  
 self.conn.autocommit = True

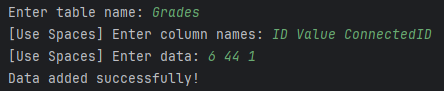
*Генерування*

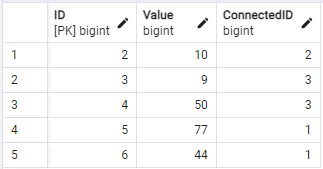
def generate\_data(self, table\_name, count):  
 c = self.conn.cursor()  
 c.execute(f"SELECT column\_name, data\_type FROM information\_schema.columns WHERE table\_name = %s", (table\_name,))  
 columns\_info = c.fetchall()  
  
 for i in range(count):  
 columns = []  
 values = []  
  
 if table\_name == 'Students-subjects':  
 c.execute('SELECT "ID" FROM "Students" ORDER BY RANDOM() LIMIT 1')  
 student\_id = c.fetchone()  
 if not student\_id:  
 print(f"Error: No students available for row {i + 1}")  
 continue  
  
 c.execute('SELECT "ID" FROM "Subjects" ORDER BY RANDOM() LIMIT 1')  
 subject\_id = c.fetchone()  
 if not subject\_id:  
 print(f"Error: No subjects available for row {i + 1}")  
 continue  
  
 elif table\_name == 'Grades':  
 c.execute('SELECT "ID\_connected" FROM "Students-subjects" ORDER BY RANDOM() LIMIT 1')  
 connected\_id = c.fetchone()  
 if not connected\_id:  
 print(f"Error: No connected IDs available for row {i + 1}")  
 continue  
  
 for column\_info in columns\_info:  
 column\_name = column\_info[0]  
 column\_type = column\_info[1]  
  
 columns.append(f'"{column\_name}"')  
  
 if column\_name == 'ID\_connected' and table\_name == 'Students-subjects':  
 c.execute('SELECT COALESCE(MAX("ID\_connected"), 0) FROM "Students-subjects"')  
 max\_id = c.fetchone()[0]  
 values.append(str(max\_id + 1))  
  
 elif column\_name == 'ID':  
 c.execute(f'SELECT COALESCE(MAX("ID"), 0) FROM "{table\_name}"')  
 max\_id = c.fetchone()[0]  
 values.append(str(max\_id + 1))  
  
 elif column\_name == 'StudentsID':  
 values.append(str(student\_id[0]))  
  
 elif column\_name == 'SubjectsID':  
 values.append(str(subject\_id[0]))  
  
 elif column\_name == 'ConnectedID':  
 values.append(str(connected\_id[0]))  
  
 elif column\_name == 'Value' and table\_name == 'Grades':  
 values.append(str(random.randint(1, 100)))  
  
 elif column\_type == 'integer':  
 values.append(str(random.randint(1, 100)))  
  
 elif column\_type == 'character varying':  
 if column\_name in ['Name', 'Surname', 'Patronymic']:  
 values.append(f"'Test{i}'")  
 else:  
 values.append(f"'Text\_{i}'")  
  
 if columns and values:  
 insert\_query = f'INSERT INTO "{table\_name}" ({", ".join(columns)}) VALUES ({", ".join(values)})'  
 try:  
 c.execute(insert\_query)  
 self.conn.commit()  
 except psycopg2.Error as e:  
 print(f"Error inserting row {i + 1}: {e}")  
 self.conn.rollback()  
 continue  
  
 print(f"Data generation for table {table\_name} completed")

***Скріншоти результатів виконання програми***

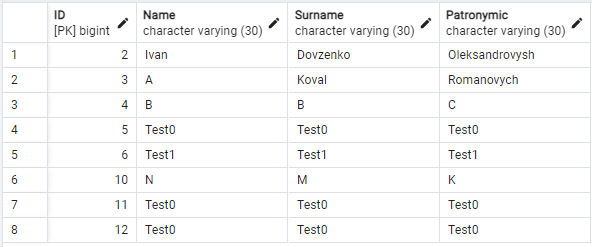
*Внесення*

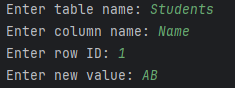


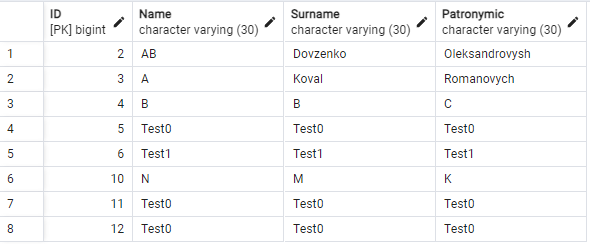




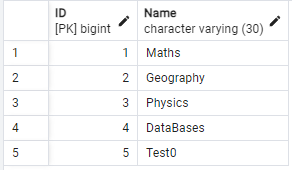
*Редагування*

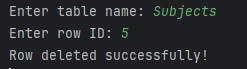


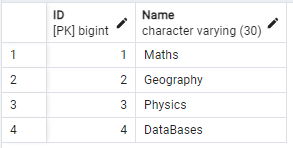




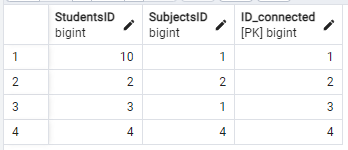
*Видалення*

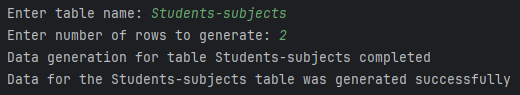


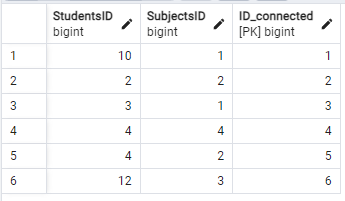




*Генерування*







***Посилання на GitHub: https://github.com/PrYulianna/RGR\_PRYIMAK***