Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ОТЧЕТ

по лабораторной работе № 1

на тему

РАЗРАБОТКА ТРЕБОВАНИЙ К СХЕМЕ ДАННЫХ И ПОЛЬЗОВАТЕЛЬСКОМУ ИНТЕРФЕЙСУ ПРИКЛАДНОЙ ПРОГРАММЫ. ПРАКТИЧЕСКОЕ ЗНАКОМСТВО С ИНТЕРФЕЙСОМ POSTGRESQL. ВАРИАНТ №11 (ШКОЛА)

Студент: А.Н. Климович

Преподаватель: Ю.Ю. Желтко

МИНСК 2024

**1 ЦЕЛЬ РАБОТЫ**

1. Научиться разрабатывать схему базы данных (БД) и требования к пользовательскому интерфейсу (UI).

2. Ознакомиться с основными командами и интерфейсом работы в PostgreSQL.

**2 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ**

**2.1 Разработка схемы данных**

1. Спроектировать схему базы данных, включающую основную таблицу, содержащую данные, достаточные для работы пользовательского интерфейса.

2. Добавить 3-4 справочных таблиц (LUT) для заполнения и верификации полей основной таблицы.

3. Не менее двух справочных таблиц должны быть связаны с основной таблицей отношением «многие ко многим».

**2.2 Разработка пользовательского интерфейса**

1. Описать, как пользователи будут взаимодействовать с приложением для работы с базой данных.

2. Интерфейс должен поддерживать следующие функции: добавление, изменение и удаление данных в интерактивном и пакетном режимах.

Интерактивный режим – это режим, в котором пользователь работает с программой в реальном времени, вводя команды вручную и немедленно получая на них отклик.

Пакетный режим – режим, в котором команды или запросы собираются в одном файле или пакете и выполняются все вместе, одним запуском, без участия пользователя в реальном времени.

**2.3 Технические требования**

1. Разработать спецификацию (техническое задание) для базы данных и пользовательского интерфейса:

– описание структуры базы данных (какие таблицы нужны, какие поля они содержат);

– описание связей между таблицами (какие поля связаны и как);

– описание операций, доступных пользователю (что можно добавлять, изменять, удалять);

– описание взаимодействия пользователя с интерфейсом программы.

**2.4 Работа с PostgreSQL**

1. Практически освоить команды работы с PostgreSQL для выполнения операций добавления, изменения и удаления данных в базе данных (БД).

**3 ВЫПОЛНЕНИЕ РАБОТЫ**

**3.1 Разработка схемы данных**

В таблице 3.1 представлено описание сущностей, выделенных в модели «Школа».

Таблица 3.1 – Сущности

|  |  |  |  |
| --- | --- | --- | --- |
| Имя сущности | Имя таблицы, отображающей сущность | Названия атрибутов | Названия полей таблицы |
| Ученик | student | Имя | first\_name |
| Фамилия | last\_name |
| Пол | gender\_type |
| Учитель | teacher | Имя | first\_name |
| Фамилия | last\_name |
| Пол | gender\_type |
| Возраст | age |
| Номер телефона | phone\_no |
| Класс | class | Название класса | class\_name |
| Предмет | subject | Название предмета | subject\_name |

В таблице 3.2 представлено описание связей для сущностей в модели «Школа».

Таблица 3.2 – Связи

|  |  |  |
| --- | --- | --- |
| Связь | Промежуточная таблица | Описание |
| Предметы учащегося | student\_subject | Описывает предметы, которые изучает ученик |
| Учитель для класса | class\_teacher | Описывает учителей, которые ведут предметы в определенных классах |
| Класс ученика | – | Описывает принадлежность ученика какому-либо классу |
| Предмет учителя | – | Описывает предмет как специализацию учителя |
| Классный руководитель | – | Описывает учителя для каждого класса |

По описанным сущностям и связям была разработана реляционная модель «Школа», представленная на рисунке 3.1. Для отображения связей использовалась нотация «Crow’s Foot».

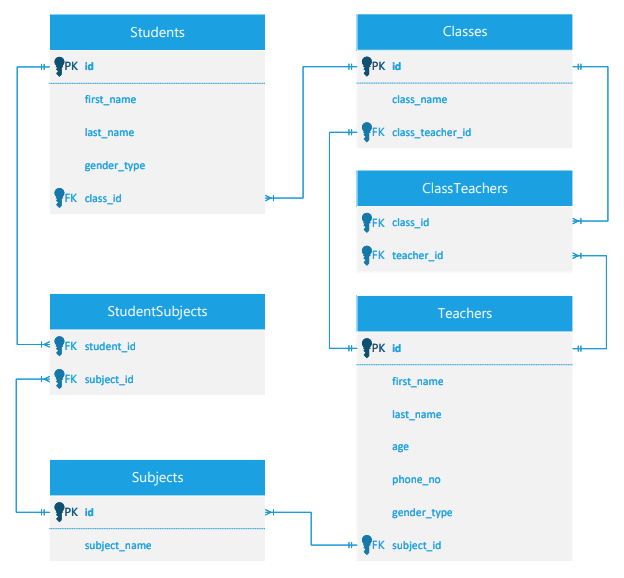


Рисунок 3.1 – Реляционная модель «Школа»

Аналогично была построена модель данных, изображенная на рисунке 3.2.

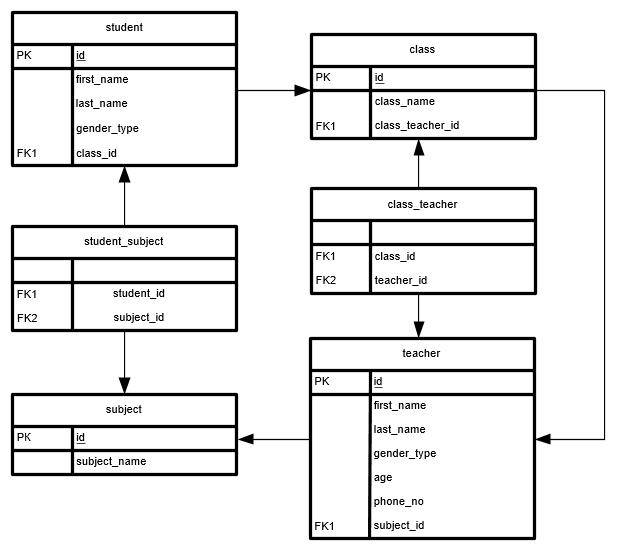


Рисунок 3.2 – Модель данных «Школа»

**3.2 Разработка пользовательского интерфейса**

**3.2.1 Техническое требования**

Для работы с базой данных «Школа» был разработан пользовательский интерфейс (UI) с возможностью взаимодействия исключительно через клавиатуру. Это подразумевает минимизацию необходимости использования графических элементов и внедрение удобной навигации через горячие клавиши и текстовые команды.

Интерфейс должен поддерживать основные функции взаимодействия с базой данных:

* добавление данных;
* просмотр данных;
* изменение данных;
* удаление данных.

Пользователи будут взаимодействовать с программой исключительно с помощью клавиатуры, что требует внедрения системы горячих клавиш и команд. Пример основных взаимодействий:

1. «Tab» – переключение между элементами управления (например, между формами ввода).

2. «Enter» – подтверждение выбора или выполнение команды.

3. «Alt + буква» – открытие соответствующего раздела меню.

4. «Ctrl + буква» – выполнение специфических действий (например, «Ctrl + A» для добавления новой записи, «Ctrl + E» для редактирования, «Ctrl + D» для удаления).

Пример пользовательского интерфейса показан на рисунке 2.2.

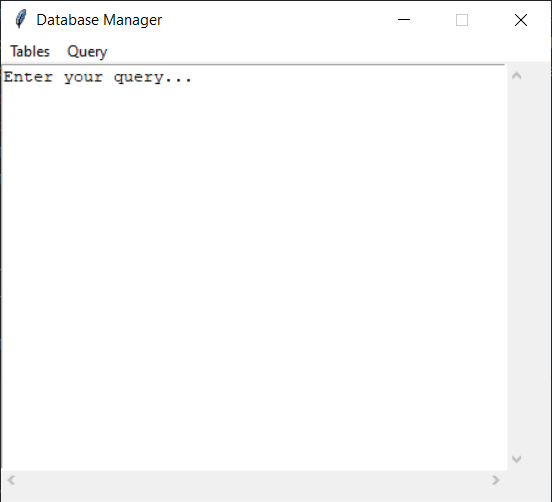


Рисунок 3.3 – Пример пользовательского интерфейса

В меню «Tables» можно выбрать таблицу, с которой дальше можно взаимодействовать (см. рисунок 3.4).

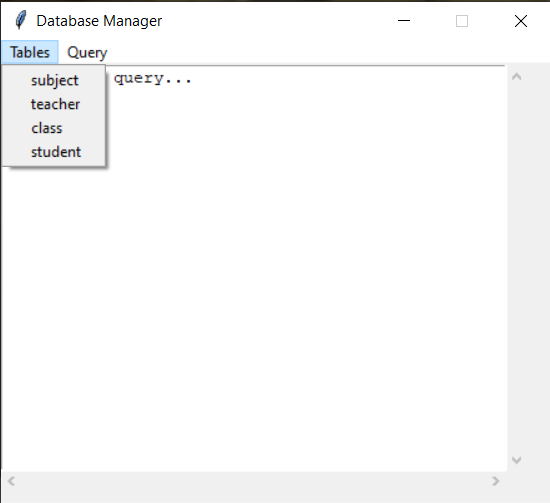


Рисунок 3.4 – Подменю «Tables»

В меню «Query» можно выполнить SQL-запрос либо из текстового редактора приложения, либо из локального файла (см. рисунок 3.5).

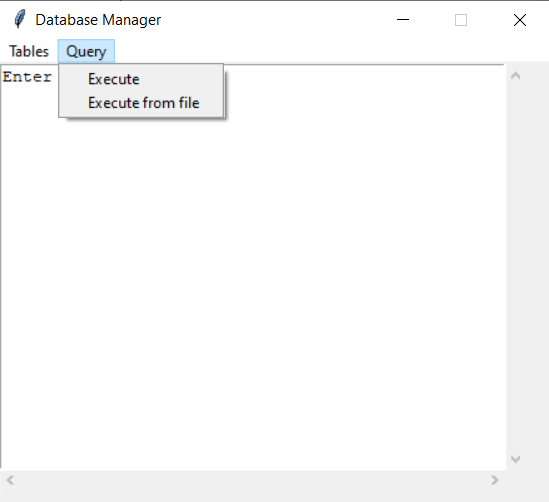


Рисунок 3.5 – Подменю «Query»

**3.2.1 Интерактивный режим**

В интерактивном режиме пользователь будет вводить команды вручную и немедленно получать отклик. Это позволяет администратору работать с базой данных в реальном времени. На рисунке 3.6 в красной рамке выделен текстовой редактор, где можно писать SQL-запросы.

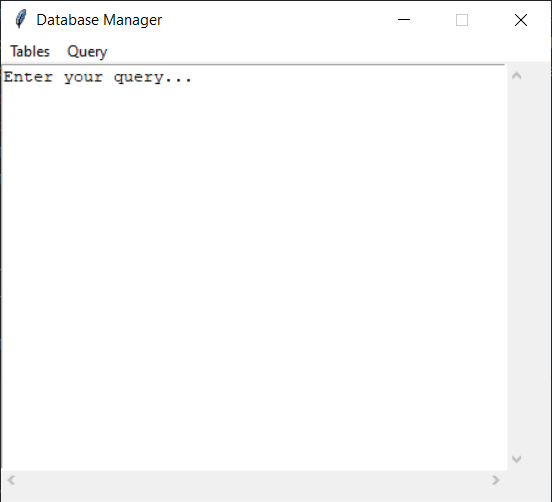


Рисунок 3.6 – Текстовый редактор для SQL-запросов для интерактивного режима

При выборе таблицы открывается новое окно для работы с выбранной таблицей. Например, на рисунке 3.7 показано окно, которое открывается после выбора таблицы «subject».

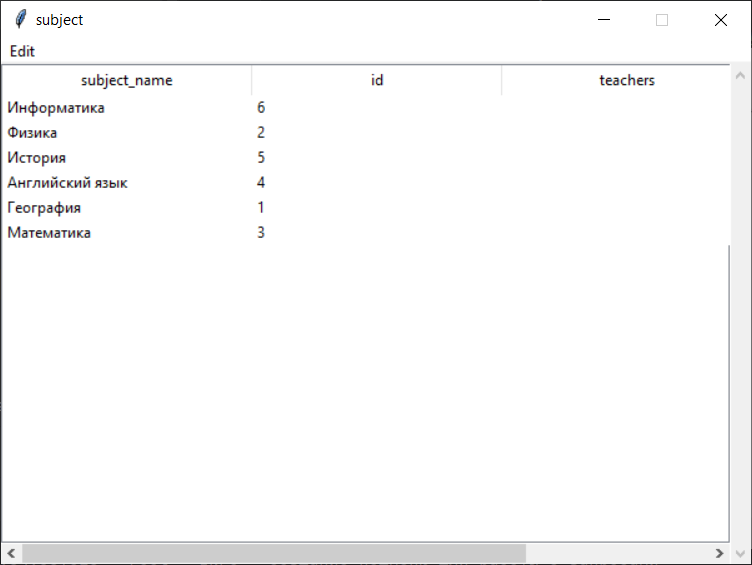


Рисунок 3.7 – Окно таблицы «subject»

В отрывшимся окне операции добавления, обновления и удаления можно выполнять через подменю «Edit» (рисунок 3.8).

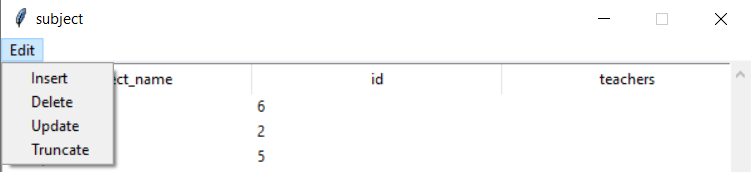


Рисунок 3.8 – Подменю «Edit»

При нажатии на «Insert» создается окно для заполнения полей сущности, которую нужно добавить в таблицу (рисунок 3.9).

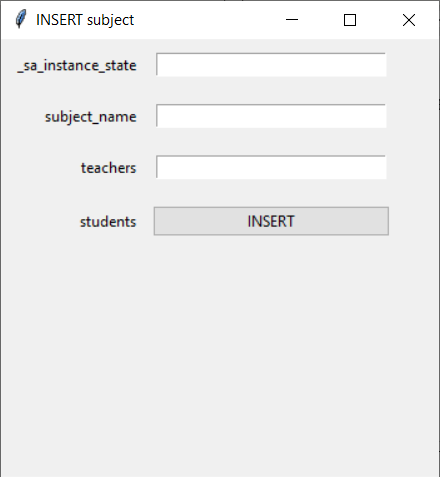


Рисунок 3.9 – Окно для добавления объекта в таблицу

При нажатии на «Delete» создается окно для заполнения полей сущности, которую нужно удалить из таблицы (рисунок 3.10).

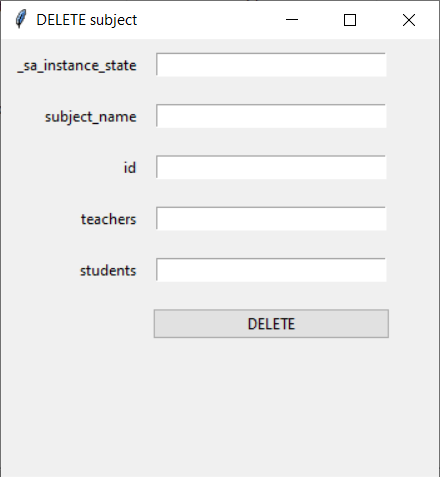


Рисунок 3.10 – Окно для удаления объекта из таблицы

При нажатии на «Update» создается окно для заполнения полей сущности, которую нужно обновить в таблице (рисунок 3.11).

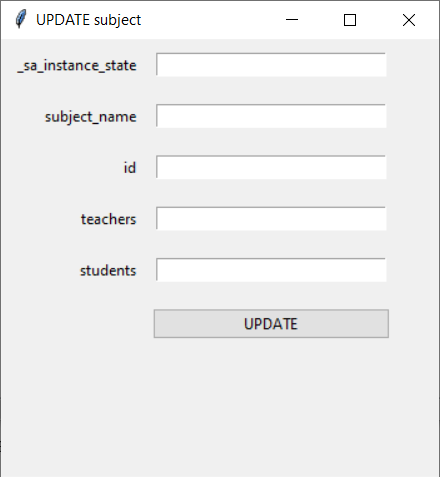


Рисунок 3.11 – Окно для обновления объекта в таблице

При нажатии на «Truncate» таблица полностью очищается.

**3.2.2 Пакетный режим**

Пакетный режим позволяет выполнять набор команд за один запуск программы.

В приложении этот режим осуществляется через подменю «Query» кнопкой «Execute from file» или с помощью комбинации клавиш (рисунок 3.5).

Пакетный режим может использоваться для массового добавления, обновления, чтения или изменения данных.

**3.3 Работа с PostgreSQL**

**3.3.1 Создание таблиц**

Далее приведен скрипт для создания таблиц. Связи между таблицами создаются с помощью оператора «ALTER». В самом начале скрипта таблицы предварительно удаляются, если были ранее созданы. Сами таблицы создаются с помощью оператора «CREATE». Также для ускорения работы с данными БД некоторые таблицы были проиндексированы.

-- Удаление таблиц, если они существуют

DROP TABLE IF EXISTS public.student\_subject CASCADE;

DROP TABLE IF EXISTS public.student CASCADE;

DROP TABLE IF EXISTS public.teacher CASCADE;

DROP TABLE IF EXISTS public.subject CASCADE;

DROP TABLE IF EXISTS public.class CASCADE;

-- Создание таблицы subject

CREATE TABLE IF NOT EXISTS public.subject (

id SERIAL PRIMARY KEY,

subject\_name VARCHAR(100) NOT NULL

);

-- Создание таблицы student

CREATE TABLE IF NOT EXISTS public.student (

id SERIAL PRIMARY KEY,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

gender\_type VARCHAR(20) NOT NULL CHECK (gender\_type IN ('MALE', 'FEMALE', 'OTHER')),

class\_id INTEGER NOT NULL

);

-- Создание таблицы student\_subject для отношения многие ко многим между student и subject

CREATE TABLE IF NOT EXISTS public.student\_subject (

student\_id INTEGER NOT NULL,

subject\_id INTEGER NOT NULL,

PRIMARY KEY (student\_id, subject\_id)

);

-- Создание таблицы class

CREATE TABLE IF NOT EXISTS public.class (

id SERIAL PRIMARY KEY,

class\_name VARCHAR(100) NOT NULL,

class\_teacher\_id INTEGER

);

-- Создание таблицы teacher

CREATE TABLE IF NOT EXISTS public.teacher (

id SERIAL PRIMARY KEY,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

age INTEGER NOT NULL CHECK (age >= 18),

phone\_no VARCHAR(15) NOT NULL,

gender\_type VARCHAR(20) NOT NULL CHECK (gender\_type IN ('MALE', 'FEMALE', 'OTHER')),

subject\_id INTEGER NOT NULL

);

-- Добавление индекса для оптимизации запросов

CREATE INDEX IF NOT EXISTS idx\_student\_class ON public.student(class\_id);

CREATE INDEX IF NOT EXISTS idx\_teacher\_subject ON public.teacher(subject\_id);

-- ALTER TABLE для добавления всех внешних ключей

-- Добавление внешнего ключа для связи студента с классом

ALTER TABLE public.student

ADD CONSTRAINT fk\_class FOREIGN KEY (class\_id)

REFERENCES public.class(id)

ON UPDATE CASCADE

ON DELETE CASCADE;

-- Добавление внешнего ключа для связи учителя с предметом

ALTER TABLE public.teacher

ADD CONSTRAINT fk\_subject FOREIGN KEY (subject\_id)

REFERENCES public.subject(id)

ON UPDATE CASCADE

ON DELETE CASCADE;

-- Добавление внешнего ключа для связи классного руководителя с классом

ALTER TABLE public.class

ADD CONSTRAINT fk\_teacher FOREIGN KEY (class\_teacher\_id)

REFERENCES public.teacher(id)

ON UPDATE CASCADE

ON DELETE SET NULL;

-- Добавление внешнего ключа для связи таблицы student\_subject (многие ко многим) с таблицей student

ALTER TABLE public.student\_subject

ADD CONSTRAINT fk\_student FOREIGN KEY (student\_id)

REFERENCES public.student(id)

ON DELETE CASCADE;

-- Добавление внешнего ключа для связи таблицы student\_subject (многие ко многим) с таблицей subject

ALTER TABLE public.student\_subject

ADD CONSTRAINT fk\_subject FOREIGN KEY (subject\_id)

REFERENCES public.subject(id)

ON DELETE CASCADE;

**3.3.2 Добавление данных**

Далее приведен скрипт для добавления данных в БД.

INSERT INTO public.subject (subject\_name) VALUES

('Mathematics'),

('Physics'),

('Chemistry'),

('Biology'),

('History'),

('Geography'),

('English'),

('French'),

('Spanish'),

('Computer Science'),

('Art'),

('Music'),

('Physical Education'),

('Economics'),

('Psychology'),

('Philosophy'),

('Political Science'),

('Sociology'),

('Literature'),

('Drama'),

('Business Studies'),

('Environmental Science'),

('Astronomy'),

('Law'),

('Media Studies'),

('Engineering'),

('Architecture'),

('Design'),

('Statistics'),

('Anthropology');

INSERT INTO public.class (class\_name, class\_teacher\_id) VALUES

('1A', NULL),

('1B', NULL),

('2A', NULL),

('2B', NULL),

('3A', NULL),

('3B', NULL),

('4A', NULL),

('4B', NULL),

('5A', NULL),

('5B', NULL),

('6A', NULL),

('6B', NULL),

('7A', NULL),

('7B', NULL),

('8A', NULL),

('8B', NULL),

('9A', NULL),

('9B', NULL),

('10A', NULL),

('10B', NULL),

('11A', NULL),

('11B', NULL),

('12A', NULL),

('12B', NULL),

('13A', NULL),

('13B', NULL),

('14A', NULL),

('14B', NULL),

('15A', NULL),

('15B', NULL);

INSERT INTO public.teacher (first\_name, last\_name, age, phone\_no, gender\_type, subject\_id) VALUES

('John', 'Doe', 40, '123-456-7890', 'MALE', 1),

('Jane', 'Smith', 35, '123-456-7891', 'FEMALE', 2),

('Alice', 'Johnson', 29, '123-456-7892', 'FEMALE', 3),

('Bob', 'Brown', 50, '123-456-7893', 'MALE', 4),

('Charlie', 'Davis', 45, '123-456-7894', 'MALE', 5),

('Emily', 'Miller', 33, '123-456-7895', 'FEMALE', 6),

('Grace', 'Wilson', 38, '123-456-7896', 'FEMALE', 7),

('Henry', 'Moore', 42, '123-456-7897', 'MALE', 8),

('Ivy', 'Taylor', 30, '123-456-7898', 'FEMALE', 9),

('James', 'Anderson', 36, '123-456-7899', 'MALE', 10),

('Katherine', 'Thomas', 41, '123-456-7800', 'FEMALE', 11),

('Liam', 'Jackson', 39, '123-456-7801', 'MALE', 12),

('Mia', 'White', 31, '123-456-7802', 'FEMALE', 13),

('Noah', 'Harris', 43, '123-456-7803', 'MALE', 14),

('Olivia', 'Martin', 37, '123-456-7804', 'FEMALE', 15),

('Paul', 'Garcia', 44, '123-456-7805', 'MALE', 16),

('Quincy', 'Martinez', 46, '123-456-7806', 'MALE', 17),

('Rachel', 'Robinson', 28, '123-456-7807', 'FEMALE', 18),

('Sam', 'Clark', 49, '123-456-7808', 'MALE', 19),

('Tina', 'Rodriguez', 34, '123-456-7809', 'FEMALE', 20),

('Ulysses', 'Lewis', 32, '123-456-7810', 'MALE', 21),

('Victoria', 'Lee', 47, '123-456-7811', 'FEMALE', 22),

('Will', 'Walker', 35, '123-456-7812', 'MALE', 23),

('Xander', 'Hall', 29, '123-456-7813', 'MALE', 24),

('Yvonne', 'Allen', 48, '123-456-7814', 'FEMALE', 25),

('Zoe', 'Young', 27, '123-456-7815', 'FEMALE', 26),

('Arthur', 'King', 52, '123-456-7816', 'MALE', 27),

('Betty', 'Wright', 53, '123-456-7817', 'FEMALE', 28),

('Carl', 'Scott', 54, '123-456-7818', 'MALE', 29),

('Diana', 'Green', 55, '123-456-7819', 'FEMALE', 30);

INSERT INTO public.student (first\_name, last\_name, gender\_type, class\_id) VALUES

('Ethan', 'Reed', 'MALE', 1),

('Sophia', 'Bailey', 'FEMALE', 2),

('Jacob', 'Cook', 'MALE', 3),

('Ava', 'Bell', 'FEMALE', 4),

('Michael', 'Parker', 'MALE', 5),

('Isabella', 'Murphy', 'FEMALE', 6),

('William', 'Price', 'MALE', 7),

('Mia', 'Gray', 'FEMALE', 8),

('James', 'Ramirez', 'MALE', 9),

('Olivia', 'Torres', 'FEMALE', 10),

('Benjamin', 'Peterson', 'MALE', 11),

('Amelia', 'Cooper', 'FEMALE', 12),

('Lucas', 'Brooks', 'MALE', 13),

('Emma', 'Foster', 'FEMALE', 14),

('Henry', 'Sanders', 'MALE', 15),

('Emily', 'Jenkins', 'FEMALE', 16),

('Alexander', 'Perez', 'MALE', 17),

('Ella', 'Sullivan', 'FEMALE', 18),

('Daniel', 'James', 'MALE', 19),

('Avery', 'Barnes', 'FEMALE', 20),

('Jackson', 'Ross', 'MALE', 21),

('Harper', 'Ward', 'FEMALE', 22),

('Sebastian', 'Butler', 'MALE', 23),

('Grace', 'Russell', 'FEMALE', 24),

('David', 'Stewart', 'MALE', 25),

('Chloe', 'Young', 'FEMALE', 26),

('Joseph', 'Hughes', 'MALE', 27),

('Scarlett', 'Bryant', 'FEMALE', 28),

('Matthew', 'Williams', 'MALE', 29),

('Abigail', 'Hernandez', 'FEMALE', 30);

INSERT INTO public.student\_subject (student\_id, subject\_id) VALUES

(1, 1), (1, 2), (1, 3),

(2, 4), (2, 5), (2, 6),

(3, 7), (3, 8), (3, 9),

(4, 10), (4, 11), (4, 12),

(5, 13), (5, 14), (5, 15),

(6, 16), (6, 17), (6, 18),

(7, 19), (7, 20), (7, 21),

(8, 22), (8, 23), (8, 24),

(9, 25), (9, 26), (9, 27),

(10, 28), (10, 29), (10, 30);

**3.3.3 Выполнение запросов на выборку**

Пример 1. Получить всех студентов и их классы.

SELECT

student.first\_name AS student\_first\_name,

student.last\_name AS student\_last\_name,

public.class.class\_name

FROM

public.student

JOIN

public.class ON public.student.class\_id = public.class.id;

Результат выполнения скрипта из примера 1 показан на рисунке 3.12.

Пример 2. Получить всех учителей и их предметы

SELECT

teacher.first\_name AS teacher\_first\_name,

teacher.last\_name AS teacher\_last\_name,

subject.subject\_name

FROM

public.teacher

JOIN

public.subject ON teacher.subject\_id = subject.id;

Результат выполнения скрипта из примера 2 показан на рисунке 3.13.

Пример 3. Выбрать всех студентов, которые изучают определенный предмет (например, «Mathematics»)

SELECT

student.first\_name AS student\_first\_name,

student.last\_name AS student\_last\_name

FROM

public.student

JOIN

public.student\_subject ON student.student\_id = student\_subject.student\_id

JOIN

public.subject ON student\_subject.subject\_id = subject.subject\_id

WHERE

subject.subject\_name = 'Mathematics';

Результат выполнения скрипта из примера 3 показан на рисунке 3.14.

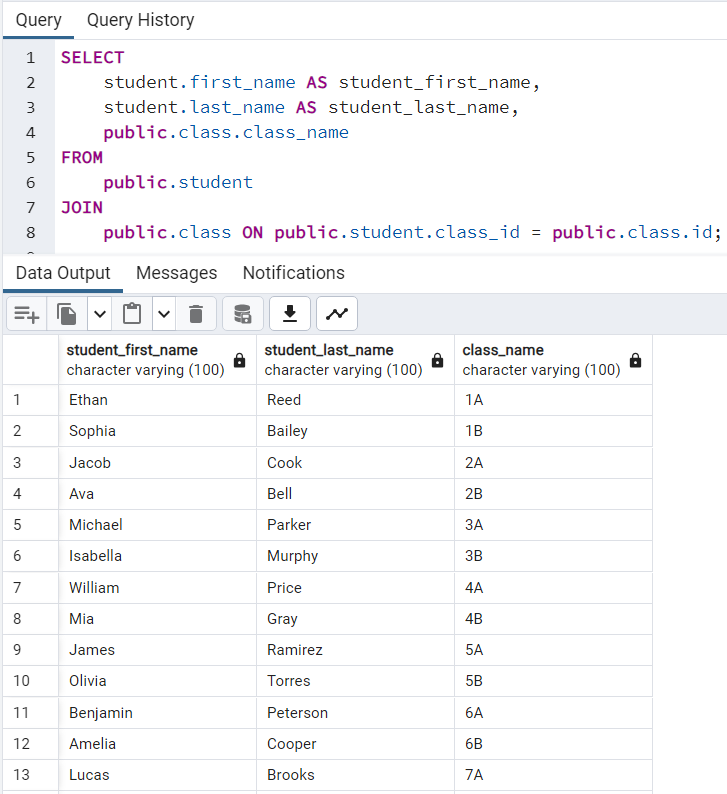


Рисунок 3.12 – Результата выполнения SQL-скрипта из примера 1

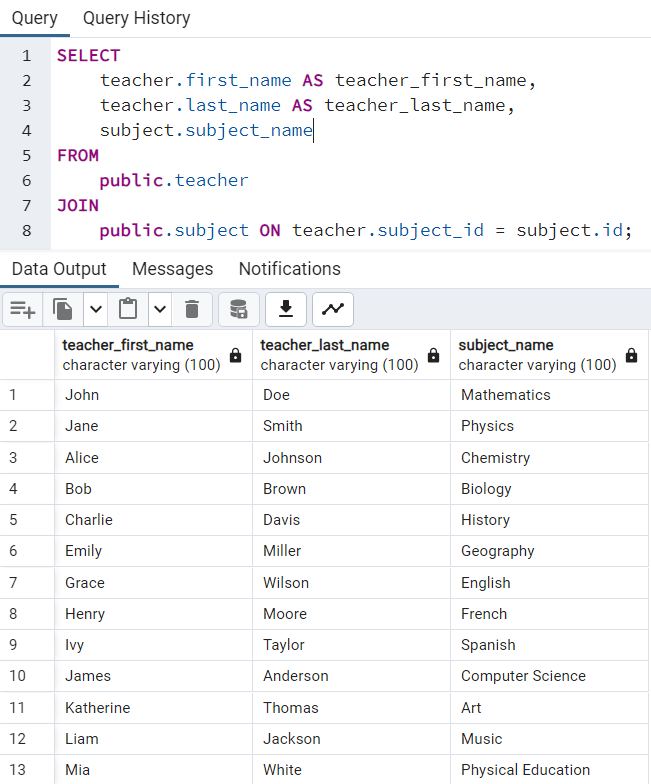


Рисунок 3.13 – Результата выполнения SQL-скрипта из примера 2

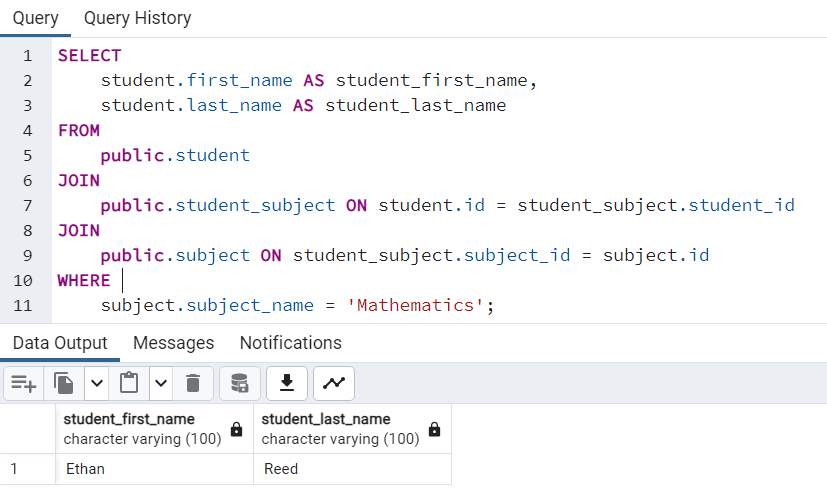


Рисунок 3.14 – Результат выполнения скрипта из примера 3

**4 ВЫВОД**

В рамках данной лабораторной работы была спроектирована схема данных и определены требования к пользовательскому интерфейсу.

В процессе разработки схемы были созданы сущности, такие как «subject», «student», «class» и «teacher», а также реализованы связи между таблицами, включая взаимосвязи типа «многие ко многим».

Кроме того, было составлено техническое задание на создание интерфейса, поддерживающего как интерактивный, так и пакетный режимы работы. Пользователи могут добавлять, редактировать и удалять данные, взаимодействуя с программой как с помощью клавиатуры, так и мыши. Это обеспечивает более быструю и удобную работу для опытных пользователей. Также была предложена система горячих клавиш для быстрого выполнения ключевых операций и навигации по разделам программы.

Во время работы были изучены основные принципы использования «PostgreSQL», что помогло закрепить навыки создания, изменения таблиц и выполнения основных операций с данными в реальной базе данных. Практическое применение команд «PostgreSQL» способствовало углублению знаний о взаимодействии с реляционными базами данных и их структурированием. Также были определены связи между различными классами и их мощности.