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 3 (Аэропорт)

Студент: И.И. Божко

Преподаватель: Ю.Ю. Желтко

МИНСК 2024

1 ЦЕЛЬ РАБОТЫ

В ходе выполнения данной лабораторной работы необходимо разработать требования к схеме данных и пользовательскому интерфейсу прикладной программы, практически ознакомиться с интерфейсом PostgreSQL.

2 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1) Разработка схемы данных. Спроектировать схему базы данных, включающую основную таблицу, содержащую данные, достаточные для работы пользовательского интерфейса. Добавить 3-4 справочных таблиц (LUT) для заполнения и верификации полей основной таблицы. Не менее двух справочных таблиц должны быть связаны с основной таблицей отношением «многие ко многим».

2) Разработка пользовательского интерфейса. Описать, как пользователи будут взаимодействовать с приложением для работы с базой данных. 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 - это режим, в котором команды или запросы собираются в одном файле или пакете и выполняются все вместе, одним запуском, без участия пользователя в реальном времени.

3) Технические требования. 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.

4) Работа с PostgreSQL. Практически освоить команды работы с PostgreSQL для выполнения операций добавления, изменения и удаления данных в базе данных.

3 ВЫПОЛНЕНИЕ РАБОТЫ

3.1 Разработка схемы данных

Модель данных “Аэропорт” содержит сущности “Airline” – авиакомпания, “Destination” – направление, “Flight” – рейс, “Airplane” – самолёт, “Passenger” – пассажир.

В таблице 3.1 представлено описание сущности “Airline”.

Таблица 3.1 – Описание сущности “Airline”.

Название поля	Описание поля	Ключ
code	Код компании	Первичный ключ
name	Название компании	–
country	Страна	–

В таблице 3.2 представлено описание сущности “Destination”.

Таблица 3.2 – Описание сущности “Destination”.

Название поля	Описание поля	Ключ
code	Код аэропорта	Первичный ключ
name	Название аэропорта	–
country	Страна	–

В таблице 3.3 представлено описание сущности “Destination”.

Таблица 3.3 – Описание сущности “Flight”.

Название поля	Описание поля	Ключ
flight_number	Номер рейса	Первичный ключ
airline_code	Код авиакомпании	Внешний ключ 1
airplane_code	Код аэропорта	Внешний ключ 2
days	Дни недели	–
arrival_time	Время прибытия	–
departure_time	Время отправления	–

В таблице 3.4 представлено описание сущности “Airplane”.

Таблица 3.4 – Описание сущности “Airplane”.

Название поля	Описание поля	Ключ
code	Код самолёта	Первичный ключ
name	Название самолёта	–
manufacturer	Производитель	–
capacity	Вместимость	–
width	Фюзеляж (широкий/узкий)	–

В таблице 3.5 представлено описание сущности “Passenger”.

Таблица 3.5 – Описание сущности “Passenger”.

Название поля	Описание поля	Ключ
passport_number	Номер паспорта	Первичный ключ
first_name	Имя	–
last_name	Фамилия	–
email	E-mail адрес	–

В таблице 3.6 представлено описание связей в модели данных.

Таблица 3.6 – Описание связей.

Название связи	Связываемые таблицы	Промежуточная таблица
Авиакомпания, выполняющая рейс	Рейс, Авиакомпания	–
Самолёт, выполняющий рейс	Рейс, Самолёт	–
Направление (направления) рейса	Рейс, Направление	Flights-Destinations
Пассажир (пассажиры) рейса	Рейс, Пассажир	Flights-Passengers

По описанным сущностям и связям была разработана модель данных «Аэропорт», представленная на рисунке 3.1.

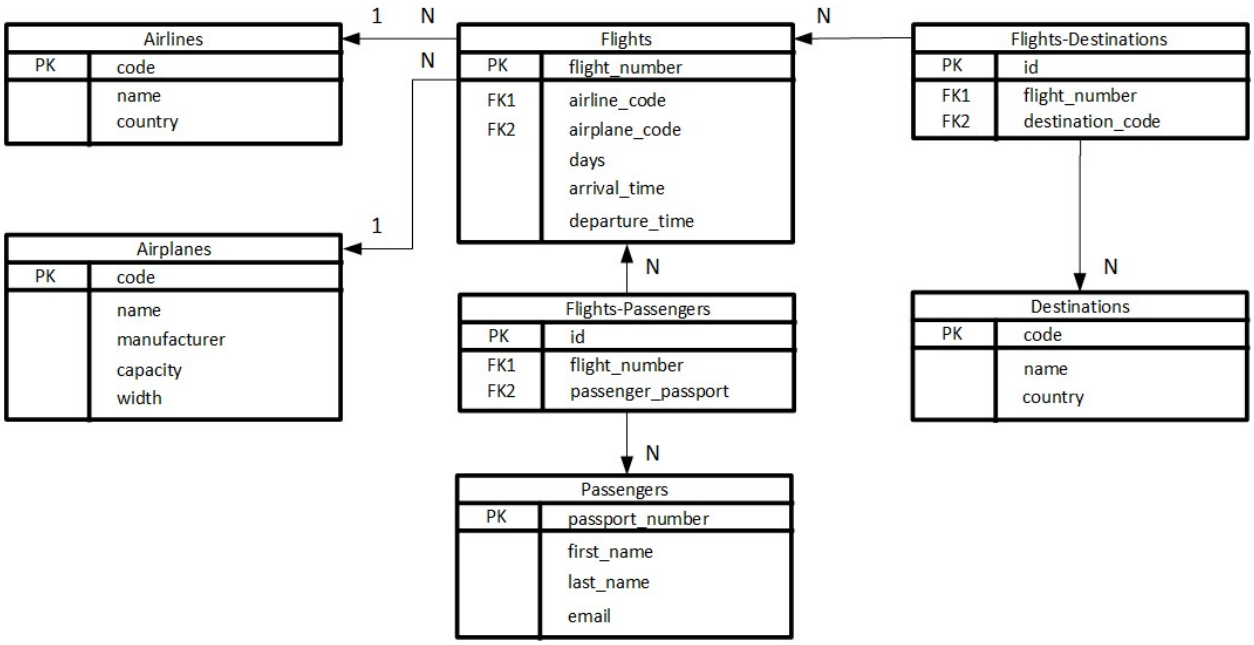

Рисунок 3.1 – Модель данных

3.2 Разработка интерфейса

Приложение предназначено для работы с базой данных “Аэропорт”. Приложение выполняет следующие функции: добавление записи, удаление записи, просмотр таблицы, удаление записи, обновление записи, выполнение SQL запросов в пакетном режиме. Для разработки приложения был выбран язык программирования Python, для создания графического интерфейса – библиотека Tkinter. Данные технологии были выбраны в связи с наличием большого количества методов и функций для работы с базой данных и текстовыми данными, а также с простотой создания графического интерфейса.

При создании записи будет создана новая строка, которая будет добавлена в базу данных после ввода всех полей. Удаление и обновление записей требует выбора записи в таблице. Для удаления после выбора необходимо нажать кнопку Delete, для обновления – ввести новые данные и нажать кнопку Update.

Для реализации взаимодействия с приложением исключительно с помощью клавиатуры для кнопок приложения назначаются горячие клавиши:

1. «`Tab`» – переключение между таблицами.
2. «`Ctrl + N`» – создать запись.
3. «`Ctrl + D`» – удалить запись.
4. «`Ctrl + P`» – выбор файла для исполнения в пакетном режиме.
5. «`Ctrl + U`» – обновить запись.
6. «`Enter`» – подтверждение операции.

На рисунке 3.2 представлен макет интерфейса приложения. На рисунке 3.3 представлен макет интерфейса при работе в пакетном режиме.

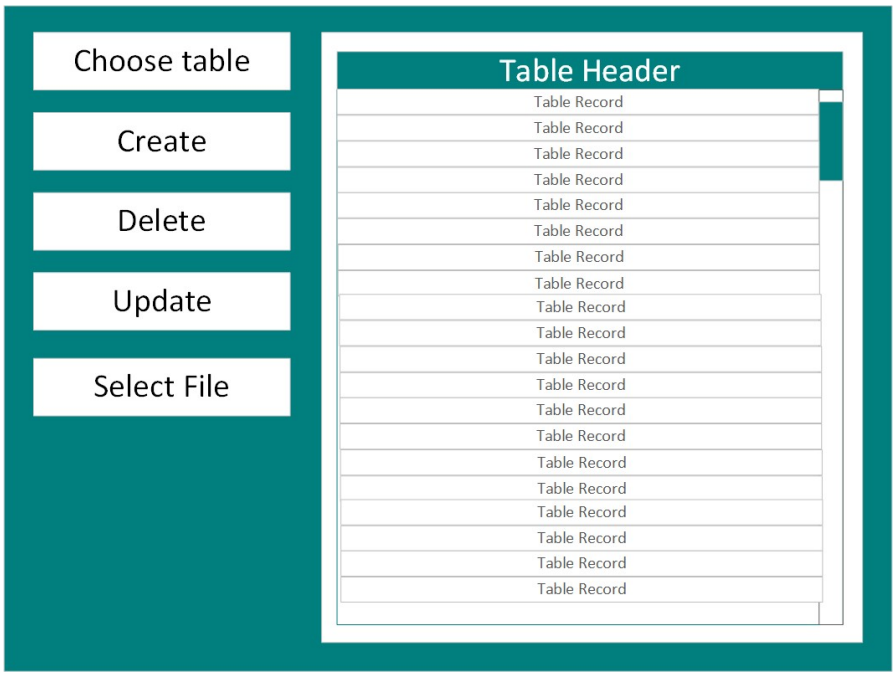

Рисунок 3.2 – Интерфейс приложения

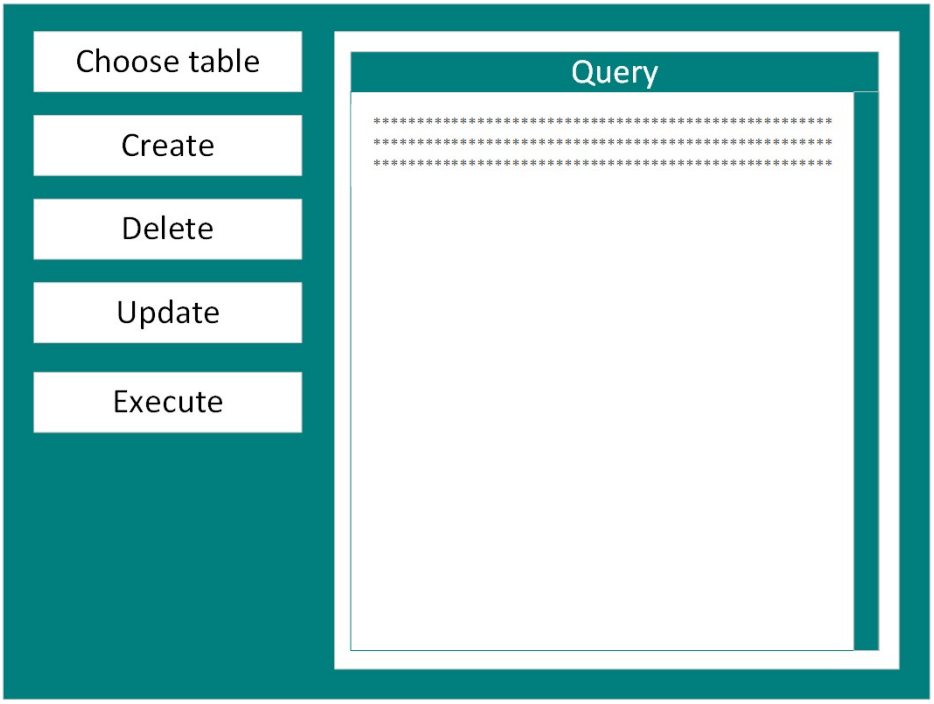

Рисунок 3.3 – Работа в пакетном режиме

3.3 SQL-команды для создания таблиц

Скрипт для реализации таблицы “Airlines”:

```
CREATE TABLE IF NOT EXISTS "Airlines"
(
    "code" varchar(2) PRIMARY KEY,
    "name" varchar(20) NOT NULL,
    "country" varchar(20) NOT NULL
)
```

Скрипт для реализации таблицы “Passengers”:

```
CREATE TABLE IF NOT EXISTS "Passengers"
(
    "passport_number" varchar(10) PRIMARY KEY,
    "first_name" varchar(20) NOT NULL,
    "last_name" varchar(20) NOT NULL,
    "email" varchar(20) NOT NULL
)
```

Скрипт для реализации таблицы “Airplanes”:

```
CREATE TABLE IF NOT EXISTS "Airplanes"
(
    "code" varchar(5) PRIMARY KEY,
    "name" varchar(20) NOT NULL,
```

```
    "manufacturer" varchar(10) NOT NULL,
    "capacity" integer NOT NULL,
    "width" integer NOT NULL
)
```

Скрипт для реализации таблицы “Destinations”:

```
CREATE TABLE IF NOT EXISTS "Destinations"
(
    "code" varchar(3) PRIMARY KEY,
    "name" varchar(20) NOT NULL,
    "country" varchar(20) NOT NULL,
)
```

Скрипт для реализации таблицы “Flights”:

```
CREATE TABLE IF NOT EXISTS "Flights"
(
    "flight_number" varchar(6) PRIMARY KEY,
    "airline_code" varchar(3) NOT NULL,
    "airplane_code" varchar(5) NOT NULL,
    "days" varchar(7) NOT NULL,
    "arrival_time" time without time zone,
    "departure_time" time without time zone,
    CONSTRAINT "Airline" FOREIGN KEY ("Airline")
        REFERENCES "Airlines" ("code") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT "Airplane" FOREIGN KEY ("Airline")
        REFERENCES "Airplanes" ("code") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
)
```

Скрипт для реализации таблицы “Flights-Destinations”:

```
CREATE TABLE IF NOT EXISTS "Flights-Destinations"
(
    "id" serial PRIMARY KEY,
    "flight_number" varchar(10) NOT NULL,
    "destination_code" varchar(3) NOT NULL,
     CONSTRAINT "Flight" FOREIGN KEY ("Flight")
        REFERENCES "Flights" ("flight_number") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT "Destination" FOREIGN KEY ("Destination")
        REFERENCES "Destinations" ("code") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
)
```

Скрипт для реализации таблицы “Flights-Passengers”:

```
CREATE TABLE IF NOT EXISTS "Flights- Passengers "
(
    "id" serial PRIMARY KEY,
    "flight_number" varchar(10) NOT NULL,
    "passenger_passport" varchar(10) NOT NULL,
     CONSTRAINT "Flight" FOREIGN KEY ("Flight")
        REFERENCES "Flights" ("flight_number") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT "Passenger" FOREIGN KEY ("Passenger")
        REFERENCES  "Passengers"  ("passport_number")  MATCH
SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
)
```

3.4 Примеры выполнения основных операций

Пример добавления данных в таблицу “Destinations”:

```
INSERT INTO destinations (code, name, country) VALUES
('MCO', 'Orlando', 'USA'),
('OXB', 'Bissau', 'Guinea-Bissau'),
('BQT', 'Brest', 'Belarus'),
('NRT', 'Tokyo Narita', 'Japan'),
('KEF', 'Reykjavik', 'Iceland').
```

Аналогично заполняются остальные таблицы, при этом id генерируется автоматически.

Пример условной выборки данных из таблицы “Flights”:

```
SELECT * FROM Flights WHERE airline_code = ‘B2’.
```

Пример изменения записи в таблице “Passengers”:

```
UPDATE Passengers
SET passport_number = “AB1234567”     WHERE last_name     =
“Johnson”;
```

Пример условного удаления записей из таблицы “Airplanes”:

```
DELETE FROM Airplanes WHERE airplane_code = “B738”.
```

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работ была разработана модель данных, требования к пользовательскому интерфейсу и макет интерфейса приложения.

В процессе разработки схемы были созданы сущности «Airline», «Airplane», «Destination», «Passenger», «Flight», а также реализованы связи между таблицами.

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.