# Базы данных

**Практическая работа № 16 «Распределенные БД.»**

## **Цель:**

## Возможности SQL для создания распределенных БД.

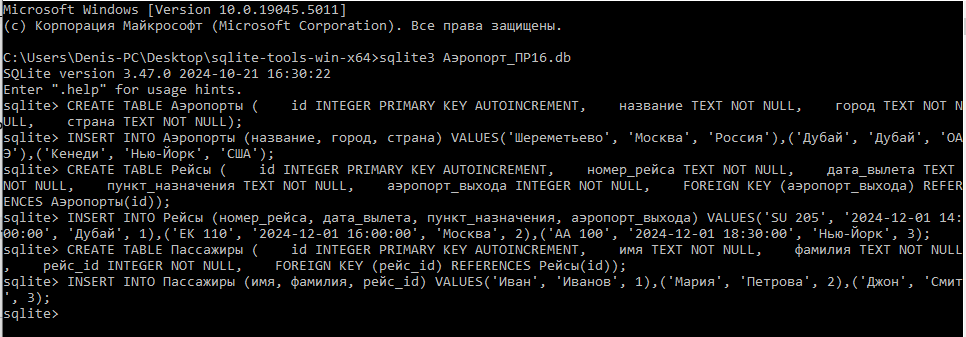
## **Задание.**

Согласно своему варианту разработайте клиентское приложение, соответствующее следующим предметным областям: (Вариант 13) - Клиентское приложение для учета деятельности аэропорта: учет выполнения рейсов.

## **Ход выполнения.**

Клиентское приложение для учета деятельности аэропорта с помощью NetBeans IDE 8.2 и базы данных SQLite3. Приложение может быть реализовано на языке Java с использованием JDBC для работы с базой данных.

**Создание базы данных и таблиц:**



* Таблица "Аэропорты": хранит информацию об аэропортах.
* Таблица "Рейсы": хранит информацию о рейсах, таких как номер рейса, пункт назначения, время вылета и т.д.
* Таблица "Пассажиры": информация о пассажирах, которые совершают рейсы.

**Создание клиентского приложения на Java:**

Клиентское приложение с графическим интерфейсом (GUI) на Swing. Приложение будет отображать информацию о рейсах, аэропортах и пассажирах, а также позволять выполнять операции с базой данных.

**Создание класса для работы с базой данных (JDBC):**

import java.sql.\*;

public class AirportDB {

private static final String URL = "jdbc:sqlite:Аэропорт\_ПР16.db";

public static Connection getConnection() throws SQLException {

return DriverManager.getConnection(URL);

}

// Получить список всех аэропортов

public static void getAirports() {

String sql = "SELECT \* FROM Аэропорты";

try (Connection conn = getConnection(); Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {

while (rs.next()) {

System.out.println("ID: " + rs.getInt("id") + ", Название: " + rs.getString("название") +

", Город: " + rs.getString("город") + ", Страна: " + rs.getString("страна"));

}

} catch (SQLException e) {

e.printStackTrace();

}

}

// Получить список всех рейсов

public static void getFlights() {

String sql = "SELECT \* FROM Рейсы";

try (Connection conn = getConnection(); Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {

while (rs.next()) {

System.out.println("Номер рейса: " + rs.getString("номер\_рейса") +

", Дата вылета: " + rs.getString("дата\_вылета") +

", Пункт назначения: " + rs.getString("пункт\_назначения"));

}

} catch (SQLException e) {

e.printStackTrace();

}

}

// Получить всех пассажиров на определённом рейсе

public static void getPassengers(int flightId) {

String sql = "SELECT \* FROM Пассажиры WHERE рейс\_id = ?";

try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setInt(1, flightId);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

System.out.println("Имя: " + rs.getString("имя") + ", Фамилия: " + rs.getString("фамилия"));

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

**Создание GUI с использованием Swing:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class AirportApp {

public static void main(String[] args) {

JFrame frame = new JFrame("Учет рейсов аэропорта");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(500, 400);

// Создание панели с кнопками

JPanel panel = new JPanel();

panel.setLayout(new FlowLayout());

JButton btnAirports = new JButton("Показать аэропорты");

JButton btnFlights = new JButton("Показать рейсы");

JButton btnPassengers = new JButton("Пассажиры рейса");

panel.add(btnAirports);

panel.add(btnFlights);

panel.add(btnPassengers);

// Добавление панели в окно

frame.add(panel);

// Действия при нажатии кнопок

btnAirports.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

AirportDB.getAirports();

}

});

btnFlights.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

AirportDB.getFlights();

}

});

btnPassengers.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String flightId = JOptionPane.showInputDialog(frame, "Введите ID рейса:");

if (flightId != null && !flightId.isEmpty()) {

AirportDB.getPassengers(Integer.parseInt(flightId));

}

}

});

frame.setVisible(true);

}

}

**Выполнение кода:**

