# Базы данных

**Практическая работа № 15 «Создание пользователей для доступа к БД.»**

## **Цель:**

## Понятие и виды логинов (имен входа). Понятие серверных ролей.

## **Задание.**

## Определить 3 должностных лица, которые смогут работать с таблицами БД. Для каждого должностного лица определить набор привилегий, которыми он может пользоваться.

## **Ход выполнения.**

В NetBeans IDE 8.2 в связке с SQLite нельзя напрямую создавать учетные записи с различными правами доступа, так как SQLite не поддерживает встроенную многопользовательскую систему. Однако, можно реализовать подобный функционал через Java-программу, контролируя доступ к различным операциям на уровне кода.

Создаем в проекте файл (например, users.txt или users.json) для хранения учетных записей и ролей:

admin:admin123:admin

editor:editor123:editor

viewer:viewer123:viewer

**Реализовываем авторизацию в Java-программе**

В начале программы попросим пользователя ввести логин и пароль. В зависимости от роли (например, admin, editor, viewer), предоставляем доступ к определенным операциям.

**Реализовываем контроль прав доступа**

В зависимости от роли, разрешаем выполнение определенных операций:

* Администратор: вызов методов для создания и удаления таблиц.
* Редактор: доступ к методам UPDATE и INSERT.
* Просмотрщик: только запросы SELECT.

**Код Java:**

import java.sql.\*;

import java.util.Scanner;

public class RoleBasedAccess {

private static final String[][] USERS = {

{"admin", "admin123", "admin"},

{"editor", "editor123", "editor"},

{"viewer", "viewer123", "viewer"}

};

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Введите логин:");

String username = scanner.nextLine();

System.out.println("Введите пароль:");

String password = scanner.nextLine();

String role = authenticate(username, password);

if (role == null) {

System.out.println("Неверные учетные данные. Доступ запрещен.");

return;

}

System.out.println("Успешный вход. Ваша роль: " + role);

String url = "jdbc:sqlite:Прокат\_ПР14.db";

try (Connection conn = DriverManager.getConnection(url)) {

if (conn != null) {

switch (role) {

case "admin":

adminOperations(conn, scanner);

break;

case "editor":

editorOperations(conn, scanner);

break;

case "viewer":

viewerOperations(conn);

break;

}

}

} catch (SQLException e) {

System.out.println("Ошибка подключения: " + e.getMessage());

}

}

private static String authenticate(String username, String password) {

for (String[] user : USERS) {

if (user[0].equals(username) && user[1].equals(password)) {

return user[2]; // Возвращает роль пользователя

}

}

return null; // Если учетные данные не найдены

}

private static void adminOperations(Connection conn, Scanner scanner) throws SQLException {

System.out.println("Выберите операцию:\n1. Создать таблицу\n2. Удалить таблицу");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

String createTableSQL = "CREATE TABLE IF NOT EXISTS TestTable (id INTEGER PRIMARY KEY, name TEXT)";

conn.createStatement().execute(createTableSQL);

System.out.println("Таблица создана.");

break;

case 2:

System.out.println("Введите название таблицы для удаления:");

String tableName = scanner.nextLine();

String dropTableSQL = "DROP TABLE IF EXISTS " + tableName;

conn.createStatement().execute(dropTableSQL);

System.out.println("Таблица удалена.");

break;

default:

System.out.println("Неверный выбор.");

}

}

private static void editorOperations(Connection conn, Scanner scanner) throws SQLException {

System.out.println("Выберите операцию:\n1. Вставить данные\n2. Обновить данные");

int choice = scanner.nextInt();

scanner.nextLine(); // Считываем оставшуюся строку

switch (choice) {

case 1:

System.out.println("Введите данные для вставки (пример: 'Название, 2023'):");

String insertData = scanner.nextLine();

String insertSQL = "INSERT INTO Автомобили (марка, модель, год\_выпуска, пробег, цена\_в\_день) VALUES (" + insertData + ")";

conn.createStatement().execute(insertSQL);

System.out.println("Данные вставлены.");

break;

case 2:

System.out.println("Введите SQL-запрос для обновления:");

String updateSQL = scanner.nextLine();

conn.createStatement().execute(updateSQL);

System.out.println("Данные обновлены.");

break;

default:

System.out.println("Неверный выбор.");

}

}

private static void viewerOperations(Connection conn) throws SQLException {

String selectSQL = "SELECT \* FROM Автомобили";

ResultSet rs = conn.createStatement().executeQuery(selectSQL);

System.out.println("Данные из таблицы 'Автомобили':");

while (rs.next()) {

System.out.println(

"ID: " + rs.getInt("id") +

", Марка: " + rs.getString("марка") +

", Модель: " + rs.getString("модель") +

", Год выпуска: " + rs.getInt("год\_выпуска") +

", Пробег: " + rs.getInt("пробег") +

", Цена в день: " + rs.getDouble("цена\_в\_день")

);

}

}

}

**Результат работы:**

