Санкт-Петербургский государственный политехнический университет

Кафедра компьютерных систем и программных технологий

**Отчёт лабораторной работе №7**

**Дисциплина**: Базы данных

**Тема**: Разработка клиентского приложения

Выполнил студент гр. 43501/1 А.М. Зинченко

(подпись)

Руководитель А.В. Мяснов

(подпись)

“ ” 2014 г.

Санкт - Петербург

2014

1. **Цель работы**

Разработать консольное приложение на языке Java, выполняющее следующие функции:

* соединение с БД, выполнение фиксированного SQL-запроса и получение результатов запроса
* добавление данных в одну из таблиц БД
* выполнение хранимой процедуры

1. **Проведение работы**

В ходе лабораторной работы было создано java-приложение реализующее следующий алгоритм:

1. Инициализация драйвера
2. Создание подключения к базе данных
3. В зависимости от выбора пользователя:
   1. Выполнение запроса SELECT \*
   2. Добавление записи в vagons\_types
   3. Запуск процедуры

package database;

import java.awt.List;

import java.sql.\*;

import java.io.PrintStream;

import java.util.Scanner;

import java.util.Vector;

import javax.print.DocFlavor;

public class Database {

public static void main(String[] args) throws Exception{

// Путь

String strURL = "jdbc:firebirdsql://localhost/C:\\BD\\rgd.fdb";

// Инициализируемя Firebird JDBC driver.

// Эта строка действительна только для Firebird.

Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();

PrintStream printStream = new PrintStream(System.out, true, "cp866");

Scanner sc = new Scanner(System.in);

Connection conn=null;

//Создаём подключение к базе данных

conn = DriverManager.getConnection(strURL,"SYSDBA", "masterkey");

if (conn==null)

{

System.err.println("Could not connect to database");

}

// Получим список всех таблиц

DatabaseMetaData metaData = conn.getMetaData();

String temp2 = null;

String [] temp3 = null;

ResultSet temp=metaData.getTables(temp2, temp2, temp2, temp3);

Vector<String> vc;

vc = new Vector<String>();

while(temp.next())

{

temp2=temp.getString(3);

if(!temp2.contains("$"))

vc.add(temp2);

}

// Создаём класс, с помощью которого будут выполняться

// SQL запросы.

Statement stmt = conn.createStatement();

ResultSet rs;

int t=0;

while(t!=4)

{

printStream.println("1. Отобразить таблицу");

printStream.println("2. Добавить данные в типы вагонов");

printStream.println("3. Выполнить ХП cost");

printStream.println("4. Выход");

t=sc.nextInt();

if(t==1)

{

int t2=0;

for(int i=1;i<=vc.size();i++)

{

System.out.printf("%d. %s\n",i,vc.elementAt(i-1));

}

t2=sc.nextInt();

//Выполняем SQL запрос.

rs = stmt.executeQuery("SELECT \* from "+ vc.elementAt(t2-1));

// Выводим результат.

while(rs.next())

{

System.out.println();

printStream.println(rs.getString(1)+" "+rs.getString(2)+" "+rs.getString(3));

}

continue;

}

if(t==2)

{

printStream.println("Введите имя");

String tstr=sc.next();

printStream.println("Введите число мест");

int kol=sc.nextInt();

tstr="INSERT INTO vagons\_types(VAGONS\_TYPES\_NAMES,NUMB\_OF\_PLACES) VALUES ('"+tstr+"',"+String.valueOf(kol)+")";

stmt.executeUpdate(tstr);

continue;

}

if(t==3)

{

printStream.println("Введите start point");

int t1=sc.nextInt();

printStream.println("Введите finish point");

int t2=sc.nextInt();

printStream.println("Введите номер вагона");

int t3=sc.nextInt();

CallableStatement call\_stmt=conn.prepareCall("{call compute\_cost(?,?,?)}");

call\_stmt.setInt(1, t1);

call\_stmt.setInt(2, t2);

call\_stmt.setInt(3, t3);

call\_stmt.registerOutParameter(1, Types.INTEGER);

call\_stmt.execute();

int result = call\_stmt.getInt(1);

System.out.println("Cost: "+result);

continue;

}

}

// Освобождаем ресурсы.

stmt.close();

conn.close();

}

}

Тестирование:

1. Отобразить таблицу

2. Добавить данные в типы вагонов

3. Выполнить ХП cost

4. Выход

1

1. NAME\_ROUTES

2. ORDER

3. PLACES

4. ROUTES

5. TICKETS

6. TRAINS

7. TRIP

8. USERS

9. VAGONS

10. VAGONS\_TYPES

10

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

1. Отобразить таблицу

2. Добавить данные в типы вагонов

3. Выполнить ХП cost

4. Выход

2

Введите имя

new

Введите число мест

10

1. Отобразить таблицу

2. Добавить данные в типы вагонов

3. Выполнить ХП cost

4. Выход

1

1. NAME\_ROUTES

2. ORDER

3. PLACES

4. ROUTES

5. TICKETS

6. TRAINS

7. TRIP

8. USERS

9. VAGONS

10. VAGONS\_TYPES

10

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

7 new 10

1. Отобразить таблицу

2. Добавить данные в типы вагонов

3. Выполнить ХП cost

4. Выход

3

Введите start point

1

Введите finish point

4

Введите номер вагона

5

Cost: 339

1. **Вывод**

Трехзвенная архитектура— архитектурная модель программного комплекса, предполагающая наличие в нём трёх компонентов: клиента, сервера приложений (к которому подключено клиентское приложение) и сервера баз данных (с которым работает сервер приложений).

Клиент — это интерфейсный компонент комплекса, предоставляемый конечному пользователю. Этот уровень не должен иметь прямых связей с базой данных (по требованиям безопасности и масштабируемости), быть нагруженным основной бизнес-логикой (по требованиям масштабируемости) и хранить состояние приложения (по требованиям надёжности). На этот уровень обычно выносится только простейшая бизнес-логика.

Сервер приложений (средний слой, связующий слой) располагается на втором уровне, на нём сосредоточена большая часть бизнес-логики. Вне его остаются только фрагменты, экспортируемые на клиента (терминалы), а также элементы логики, погруженные в базу данных (хранимые процедуры и триггеры).

Сервер баз данных (слой данных) обеспечивает хранение данных

Данная архитектура позволяет получить следующие преимущества:

- Масштабируемость;

- Высокая безопасность;

- Высокая надежность;

- Балансировка нагрузки;

- Увеличение скорости работы;

- Простое обновление;

- Низкие требования к сети между клиентом и сервером приложений.

Отметим и минусы:

- Разработка приложений для трёхуровневых программных комплексов сложнее, чем для клиент-серверных приложений;

- Наличие дополнительного связующего программного обеспечения может налагать дополнительные издержки в администрировании таких комплексов.

Для создания сервера приложений производители СУБД предлагают разработчикам драйверы для работы с их базами данных. Освоение драйвера не отнимает много времени, что позволяет сразу начать разработку.