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. Выполнение запроса SELECT \* from vagons\_types и вывод результата
4. Добавление записи INSERT INTO vagons\_types VALUES (5,'New',200) и запрос пункта 3
5. Удаление последней записи delete from vagons\_types where vagons\_types\_id=5
6. Подготовка структуры к выполнению хранимой процедуры, её запуск и вывод результата

package database;

import java.sql.\*;

import java.io.PrintStream;

public class Database {

/\*\*

\* @param args the command line arguments

\*/

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");

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");

}

// Создаём класс, с помощью которого будут выполняться

// SQL запросы.

Statement stmt = conn.createStatement();

//Выполняем SQL запрос.

ResultSet rs = stmt.executeQuery("SELECT \* from vagons\_types");

// Выводим результат.

while(rs.next())

{

System.out.println();

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

}

stmt.executeUpdate("INSERT INTO vagons\_types VALUES (5,'New',200)");

rs = stmt.executeQuery("SELECT \* from vagons\_types");

// Выводим результат.

while(rs.next())

{

System.out.println();

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

}

stmt.executeUpdate("delete from vagons\_types where vagons\_types\_id=5");

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

call\_stmt.setInt(1, 1);

call\_stmt.setInt(2, 5);

call\_stmt.setInt(3, 3);

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

call\_stmt.execute();

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

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

// Освобождаем ресурсы.

stmt.close();

conn.close();

}

}

Запуск Jar-файла:

java -jar "database.jar"

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

4 Lux-2 10

1 Сидячий-3 100

2 Плацкарт 54

3 Люкс 30

4 Lux-2 10

5 New 200

Cost: 94

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.