**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Институт информационных технологий и управления**

**Кафедра компьютерных систем и программных технологий**

**ОТЧЕТ**

**по лабораторной работе №8**

**«Разработка простого клиентского приложения»**

**Информационное обеспечение систем управления**

Студент гр. 43501/1 Матлаш И.Г.

Преподаватель Мяснов А.В.

Санкт-Петербург

2015

# 1. Цель работы

Ознакомится с работой с БД с помощью API на примере СУБД Firebird 2.5 и языка Java.

# 2. Программа работы

Необходимо создать консольное приложение, выполняющее следующие функции:

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

# 3. Разработка клиентского приложения

Используется среда NetBeans 8.0.2. Работа с СУБД осуществляется с помощью драйвера JDBC из библиотеки jaybird версии 2.2.7.

Листинг приложения:

package db\_client;

import java.io.UnsupportedEncodingException;

import java.sql.\*;

import java.util.Scanner;

import java.util.Vector;

/\*\*

\*

\* @author Ivan

\*/

public class main {

public static void main(String[] args) throws ClassNotFoundException,

InstantiationException, IllegalAccessException, SQLException, UnsupportedEncodingException {

//System.setOut(new java.io.PrintStream(System.out, true, "Cp866")); //Cp866 UTF-8

//java -Dfile.encoding=Cp866 -jar "C:\Users\Иван\Documents\NetBeansProjects\DB\_Client\dist\DB\_Client.jar"

// объект класса для подключения к БД

Connection connect = null;

// объект класса, с помощью которого будут выполняться SQL запросы

java.sql.Statement stmt = null;

// создаем класс для ввода с консоли

Scanner sc = new Scanner(System.in);

int numb\_table = 0; // Номер введенной таблицы

Vector<String> vec\_tab = new Vector<String>();

String temp2 = null; // Временные строковые

String [] temp3 = null;

int var = 0; // переменная = выбранный пункт меню

String argz; // аргументы запроса insert (атрибуты таблицы)

int argz2;

int cnt\_col = 0;

// создаем класс для хранения результата запроса

ResultSet res;

// инициализируем Firebird JDBC driver

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

//путь к базе данных

String strPath = "jdbc:firebirdsql://localhost/c:/db/as.fdb";

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

// создаём подключение к базе данных

connect = DriverManager.getConnection(strPath, "SYSDBA", "masterkey");

if (connect == null) {

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

// создаём класс, с помощью которого будут выполняться SQL запросы

stmt = connect.createStatement();

System.out.println("Подключение к БД успешно выполнено.");

// Получаем список всех таблиц БД:

DatabaseMetaData metaData = connect.getMetaData();

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

while(temp.next())

{

temp2=temp.getString(3);

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

vec\_tab.add(temp2);

}

// цикл для вывода меню и выполнения операций с консоли

while (var != 4) {

// вывод меню

System.out.println("-------------------------------------------");

System.out.println("МЕНЮ:");

System.out.println("1.Вывести список таблиц.");

System.out.println("2.Добавить запись в таблицу person.");

System.out.println("3.Выполнить хранимую процедуру CHECK\_PAID.");

System.out.println("4.Выход");

// считывание номера пункта меню

System.out.println("Выберите пункт меню:");

//var = sc.nextInt();

try{

var = Integer.parseInt(sc.nextLine());

}catch(NumberFormatException e){

System.err.println("Ошибка! Вводите только цифры.");

continue;

}

if (var > 4)

System.err.println("Ошибка! Пункт меню с таким номером отсутствует.");

if (var == 1)

{

System.out.println("Список таблиц:");

for(int i=1;i<=vec\_tab.size();i++)

{

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

}

System.out.println("Введите номер таблицы для отображения ее содержимого или "

+ "\n0 для возврата в основное меню:");

try{

numb\_table=Integer.parseInt(sc.nextLine());

}catch(NumberFormatException e){

System.err.println("Ошибка! Номер должен быть числом!");

continue;

}

if((numb\_table > vec\_tab.size()) || (numb\_table < 0)){

System.err.println("Ошибка! Таблица с таким номером отсутсвует.");

continue;

}

if(numb\_table == 0){

continue;

}

System.out.println();

//Выполняем SQL запрос:

res = stmt.executeQuery("SELECT \* from "+ vec\_tab.elementAt(numb\_table-1));

// Выводим результат:

cnt\_col = res.getMetaData().getColumnCount();

// Выводим содержание таблицы:

// Сначала имена столбцов:

for(int i = 1; i < cnt\_col + 1; i++){

System.out.print(res.getMetaData().getColumnName(i)+

" | ");

}

// Затем сами записи в таблице:

while(res.next())

{

System.out.println();

for (int i = 1;i < cnt\_col + 1;i++)

{

Object obj = res.getObject(i);

if (obj!=null)

{

System.out.print(obj+" \t ");

}

}

}

System.out.println();

continue;

}

if (var == 2 )

{

if(connect == null) {

System.err.println("Соединение с БД не установлено.");

continue;

}

// вводим аргументы insert

System.out.println("Введите название имя пользователя.");

argz = sc.nextLine();

if (argz.length()>64)

{

System.err.println("Ошибка! Длина имени не должна превышать 30 символов.");

continue;

}

System.out.println("Введите номер телефона.");

try{

argz2=Integer.parseInt(sc.nextLine());

}catch(NumberFormatException e){

System.err.println("Ошибка! Номер не является числом или превышает 9 символов");

continue;

}

if (argz2 < 0)

{

System.err.println("Ошибка! Номер не может быть отрицательным.");

continue;

}

System.out.println("Введите адрес.");

String argz3 = sc.nextLine();

if (argz3.length()>64)

{

System.err.println("Ошибка! Длина адреса не должна превышать 30 символов.");

continue;

}

stmt.executeUpdate("insert into person (name,tel,adress) values ('" + argz + "',"+String.valueOf(argz2)+",'" + argz3 + "');");

System.out.println("Запись добавлена в таблицу.");

continue;

}

if (var == 3)

{

if(connect == null) {

System.err.println("Соединение с БД не установлено.");

continue;

}

System.out.println("Хранимая процедура меняет статус платных\n"

+ "объявлений, у которых истёк срок публикации.");

PreparedStatement pstmt = connect.prepareStatement("{call CHECK\_paid}");

pstmt.execute();

System.out.println("\nХранимая процедура CHECK\_PAID выполнена.");

pstmt.close();

continue;

}

if (var == 4)

{

System.out.println("Выход (exit)");

continue;

}

}

System.exit(0);

}

}

Тестирование приложения:

Подключение к БД успешно выполнено.

-------------------------------------------

МЕНЮ:

1.Вывести список таблиц.

2.Добавить запись в таблицу person.

3.Выполнить хранимую процедуру CHECK\_PAID.

4.Выход

Выберите пункт меню:

1

Список таблиц:

1. BODY

2. CAR

3. CONDITION

4. ENGINE

5. IS\_PAID

6. MARK

7. MODEL

8. MONEY

9. PERSON

10. STATUS

11. TRANSMISSION

12. VIEWS

Введите номер таблицы для отображения ее содержимого или

0 для возврата в основное меню:

6

ID | NAME |

1 Mercedes-Benz

2 Porsche

3 Lada

-------------------------------------------

МЕНЮ:

1.Вывести список таблиц.

2.Добавить запись в таблицу person.

3.Выполнить хранимую процедуру CHECK\_PAID.

4.Выход

Выберите пункт меню:

2

Введите название имя пользователя.

Ivan

Введите номер телефона.

921874532

Введите адрес.

SPB,Lenina-2/1

Запись добавлена в таблицу.

-------------------------------------------

МЕНЮ:

1.Вывести список таблиц.

2.Добавить запись в таблицу person.

3.Выполнить хранимую процедуру CHECK\_PAID.

4.Выход

Выберите пункт меню:

3

Хранимая процедура меняет статус платных

объявлений, у которых истёк срок публикации.

Хранимая процедура CHECK\_PAID выполнена.

-------------------------------------------

МЕНЮ:

1.Вывести список таблиц.

2.Добавить запись в таблицу person.

3.Выполнить хранимую процедуру CHECK\_PAID.

4.Выход

Выберите пункт меню:

4

Выход (exit)

Проверим реакцию приложения на некоторые некорректные запросы:

-------------------------------------------

МЕНЮ:

1.Вывести список таблиц.

2.Добавить запись в таблицу person.

3.Выполнить хранимую процедуру CHECK\_PAID.

4.Выход

Выберите пункт меню:

asd

**Ошибка! Вводите только цифры.**

**-------------------------------------------**

МЕНЮ:

1.Вывести список таблиц.

2.Добавить запись в таблицу person.

3.Выполнить хранимую процедуру CHECK\_PAID.

4.Выход

Выберите пункт меню:

1

Список таблиц:

1. BODY

2. CAR

3. CONDITION

4. ENGINE

5. IS\_PAID

6. MARK

7. MODEL

8. MONEY

9. PERSON

10. STATUS

11. TRANSMISSION

12. VIEWS

Введите номер таблицы для отображения ее содержимого или

0 для возврата в основное меню:

15

**Ошибка! Таблица с таким номером отсутствует.**

**-------------------------------------------**

МЕНЮ:

1.Вывести список таблиц.

2.Добавить запись в таблицу person.

3.Выполнить хранимую процедуру CHECK\_PAID.

4.Выход

Выберите пункт меню:

2

Введите название имя пользователя.

Ivan

Введите номер телефона.

345321345

Введите адрес.

asadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadadad

**Ошибка! Длина адреса не должна превышать 30 символов.**

Примечание: при работе в консоли Windows 8 приложение приходилось запускать с кодировкой Cp866. Пример запуска:

java -Dfile.encoding=Cp866 –jar "C:\DB\_Client.jar"

# 4. Вывод

В ходе работы было создано простое клиентское приложение, выполняющее различные запросы к базе данных, добавляющее записи и вызывающее хранимые процедуры. API, предоставляемые драйвером СУБД Firebird для языка Java, позволяют использовать все возможности SQL-запросов в приложении. Поскольку большая часть логики была реализована на стороне БД, клиентское приложение получилось простым.

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 и выносится на отдельный уровень, реализуется, как правило, средствами систем управления базами данных, подключение к этому компоненту обеспечивается только с уровня сервера приложений. К основным преимуществам такой архитектуры относится масштабируемость, надежность, высокая скорость работы и возможность балансировки нагрузки. Недостатки – сложность разработки приложений и дополнительные издержки в администрировании. Плюсом конкретной реализации является то, что JDBC, как и Java в целом, не привязан к конкретной аппаратной платформе, следовательно проблемы с переносимостью приложений сводятся к минимуму.