Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчёт о лабораторной работе №8**

**Дисциплина**: Базы данных

**Тема**: Клиентское приложение

Выполнил студент гр. 43501/1 Сякаев В.В.

(подпись)

Руководитель Мяснов А. В.

(подпись)

“ ” 2016 г.

Санкт-Петербург

2016

1. **Цель работы.**

Ознакомиться с разработкой клиентских приложений.

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

Задание: в приложении реализовать возможность выполнения любого запроса на выборку, запросы на изменение, добавление, удаление не должны выполняться.

1. **Выполнение работы.**

Для выполнения работы использовалась IDE NetBeans 8.1, в которой было создано консольное приложение Java.

В данном приложении Java\_musical\_Library использовались следующие классы:

* 1. **Java\_musical\_Library.java**

Это Main-класс программы. Здесь реализована основная работа с консолью и вызовом необходимых функций других классов.

* 1. **File\_Work.java**

Это основной класс программы, где реализовано всё взаимодействие с базой данных. Здесь находятся методы подключения к базе, а также взаимодействия с ней: просмотра её таблиц, изменения таблицы game, вызова хранимой процедуры update\_league.

3) RS2DOM.java

Дерево проекта:

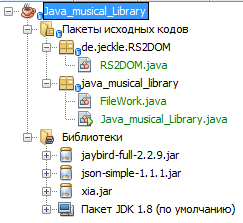
****

Рис.1. Дерево проекта

**Листинг:**

**Java\_musical\_Library.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package java\_musical\_library;  /\*\*  \*  \* @author Vitaly  \*/  import java.io.\*;  import java.io.FileNotFoundException;  import java.sql.\*;  import java.util.Scanner;  import java.util.Vector;  import org.json.simple.parser.ParseException;  import org.w3c.dom.Document;  import de.jeckle.RS2DOM.RS2DOM;  import java.sql.Connection;  public class Java\_musical\_Library {  /\*\*  \*  \* @param args  \*/  public static Scanner sc = new Scanner(System.in); // класс для работы с консолью  public static Connection connect = null;  //public static CallableStatement callableStatement = null;  public static void main(String[] args) throws ClassNotFoundException,  SQLException, InstantiationException, IllegalAccessException,  FileNotFoundException, ParseException  {    java.sql.Statement rqst = null; // объект для выполнения SQL запросов    int table\_number = 0; // номер введенной таблицы  Vector<String> vec\_tab = new Vector<String>();  String temp2 = null; // временные строковые переменные  String [] temp3 = null;  StringBuilder sb = new StringBuilder(); //объект для построения строки  int menu\_select = 0; // переменная = выбранный пункт меню  int argz; // входной аргумент для операции INSERT  int cnt\_col = 0; //переменная для вывода содержимого таблицы  ResultSet res; // Класс для хранения результатов SQL запроса  // Иницализация драйвера  Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();  //Указание пути к БД  String strPath;  strPath = "jdbc:firebirdsql://localhost/C:/Users/Vitaly/Desktop/BaseDann/my/HP/D\_5.fdb";  Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();  //Подключение к БД  connect = DriverManager.getConnection(strPath, "SYSDBA", "masterkey");  if (connect == null) {  System.err.println("Невозможно подключиться к БД.");}  //Создание класса для выполнения SQL запросов  rqst = connect.createStatement();  System.out.println("Подключение к БД успешно выполнено.");  // Получение списка 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);  }  // Меню  while (menu\_select != 6) {  // вывод меню  System.out.println("-------------------------------------------");  System.out.println("| MUSIC\_LIBRARY |");  System.out.println("-------------------------------------------");  System.out.println("Функции:");  System.out.println("1.Вывод списка всех таблиц");  System.out.println("2.Добавление записи в таблицу TYPE");  System.out.println("3.Выполнение хранимой процедуры FIVE\_ARTS");  System.out.println("4.Выполнение любых sql запросов на выборку");  System.out.println("5.Выход из программы");  // считывание номера пункта меню  System.out.println("Выберите пункт меню:");  try{  menu\_select = Integer.parseInt(sc.nextLine());  }catch(NumberFormatException e){  System.err.println("Ошибка! Вводите только цифры.");  continue;  }  if (menu\_select > 5)  System.err.println("Ошибка! Пункт меню с таким номером отсутствует.");  if (menu\_select == 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{  table\_number=Integer.parseInt(sc.nextLine());  }catch(NumberFormatException e){  System.err.println("Ошибка! Номер должен быть числом!");  continue;  }  if((table\_number > vec\_tab.size()) || (table\_number < 0)){  System.err.println("Ошибка! Таблица с таким номером отсутсвует.");  continue;  }  if(table\_number == 0){  continue;  }  System.out.println();  //Выполнение SQL запроса  res = rqst.executeQuery("SELECT \* from "+ vec\_tab.elementAt(table\_number-1));  // Вывод результата  cnt\_col = res.getMetaData().getColumnCount();    // Вывод содержимого 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 (menu\_select == 2 )  {  if(connect == null) {  System.err.println("Соединение с БД не установлено.");  continue;  }  // ВВОД АРГУМЕНТОВ ДЛЯ ОПЕРАЦИИ INSERT  System.out.println("Введите ID:");  try{  argz=Integer.parseInt(sc.nextLine());  }catch(NumberFormatException e){  System.err.println("Ошибка! ID не является числом или превышает 9 символов.");  continue;  }  if (argz <= 0)  {  System.err.println("Ошибка! ID не может быть отрицательным или равным нулю.");  continue;  }    System.out.println("Введите тип альбома:");  String argz2 = sc.nextLine();  if (argz2.length()>1 || argz2.isEmpty())  {  System.err.println("Ошибка! Название типа не может быть пустым или больше 1 символа.");  continue;  }    //////////////////  System.out.println("Введите название альбома:");  String argz22 = sc.nextLine();  if (argz22.length()>50 || argz22.isEmpty())  {  System.err.println("Ошибка! Название типа не может быть пустым или больше 50 символов.");  continue;  }      /////////////////    try{  rqst.executeUpdate("insert into TYPE values ('"+argz+"','"+argz2+"','"+argz22+"');");  System.out.println("Запись добавлена в таблицу.");  }catch (SQLException se){  System.out.println(se.getMessage());  }    continue;  }  if (menu\_select == 3)  {    if(connect == null) {  System.err.println("Соединение с БД не установлено.");  continue;  }  System.out.println("Хранимая процедура FIVE\_ARTS"  + " Вывести 5 исполнителей с наилучшим увеличением производительности за заданный период \n");    System.out.println("Введите начало периода:");  String min\_date = sc.nextLine();  System.out.println("Введите конец периода:");  String max\_date = sc.nextLine();  System.out.println();  //Вызываем функцию (которая хранится в БД)    CallableStatement callableStatement = connect.prepareCall(  "{call FIVE\_ARTS('"+min\_date+"','"+max\_date+"')} ");  ResultSet result3 = callableStatement.executeQuery();  System.out.println("\tART\_ID |\tPERFOMANCE");  System.out.println("----------------------------");  while (result3.next()) {  System.out.println("\t"+result3.getInt("ART\_ID")+"\t\t"+result3.getFloat("PERFOMANCE"));  }  System.out.println();  callableStatement.execute();  System.out.println("\nХранимая процедура FIVE\_ARTS выполнена.");  callableStatement.close();  continue;    }    if (menu\_select == 4)  {  System.out.println("\nВведите запрос: \n");  String new\_req = sc.nextLine();  if (!(new\_req.contains("UPDATE")) && !(new\_req.contains("update"))  && !(new\_req.contains("DELETE")) && !(new\_req.contains("delete"))  && !(new\_req.contains("INSERT")) && !(new\_req.contains("insert")) )  {  try{  res = rqst.executeQuery(new\_req);  cnt\_col = res.getMetaData().getColumnCount();  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();  }catch (SQLException se){  System.out.println(se.getMessage());  }    }  else {  System.out.println("Внимание! Введен неверный запрос. "  + "Изменение, добавление и удаление записей невозможно.\n");  continue;  }  }  if (menu\_select == 5)  {  System.out.println("Goodbye");  System.exit(0);  }  }  System.exit(0);  }  } |

Теперь рассмотрим взаимодействие приложения с пользователем:

При запуске программы выводится меню возможных действий с базой данных:

run:

Подключение к БД успешно выполнено.

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

| MUSIC\_LIBRARY |

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

Функции:

1.Вывод списка всех таблиц

2.Добавление записи в таблицу TYPE

3.Выполнение хранимой процедуры FIVE\_ARTS

4.Выполнение любых sql запросов на выборку

5.Выход из программы

Выберите пункт меню:

Далее пользователь должен ввести номер команды, которую он хочет выполнить.

**Рассмотрим действия, которые может выполнять созданная программа:**

* + 1. Вывод списка всех таблиц:

Выберите пункт меню:

1

Список таблиц:

1. ALBUM

2. ALBUMS\_FEST\_S

3. ALBUMS\_FEST\_SS

4. ALBUMS\_FEST\_SSS

5. ALBUM\_DATE\_S

6. ALBUM\_S

7. ALBUM\_SONG

8. ALBUM\_SONG\_S

9. ALBUM\_SORT\_S

10. ARTIST

11. ARTIST\_CNT\_SS

12. ARTIST\_NAME\_S

13. ARTIST\_NAME\_SS

14. ARTIST\_NAME\_SSS

15. ARTIST\_S

16. DISC

17. DISC\_S

18. FEST

19. FEST\_ALBUMS

20. FEST\_ALBUMS\_S

21. FEST\_CITY\_SS

22. FEST\_S

23. FEST\_SONG

24. FEST\_SONG\_S

25. GROUPS

26. GROUPS\_S

27. SONG

28. SONG\_S

29. SR\_COMPOSE\_IN\_ALBUM

30. TOP\_ARTIST\_S

31. TYPE

32. TYPE\_S

Посмотрим для примера содержимое таблицы ARTIST (10-ый номер):

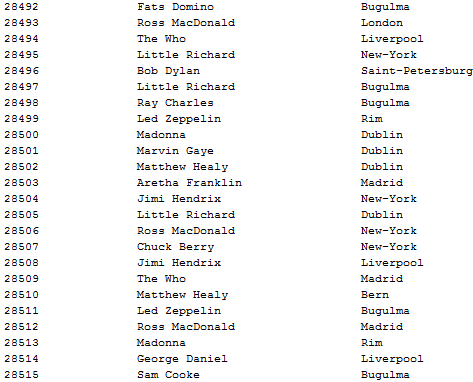


Рис.2. Фрагмент из выведенных данных

* + 1. Добавление записи в таблицу TYPE:

Выберите пункт меню:

2

Введите ID:

100003

Введите тип альбома:

v

Введите название альбома:

Video\_album

Запись добавлена в таблицу.

Воспользуемся приложением и посмотрим на таблицу TYPE:

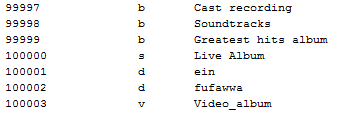


Рис.3. Фрагмент таблицы TYPE после добавления записи.

Как видно, запись была успешно добавлена в таблицу.

* + 1. Выполнение хранимой процедуры FIVE\_ARTS:

Процедура update\_league:

Вывести 5 исполнителей с наилучшим увеличением производительности за заданный период времени. Под производительностью понимать количество новых композиций за единицу времени.

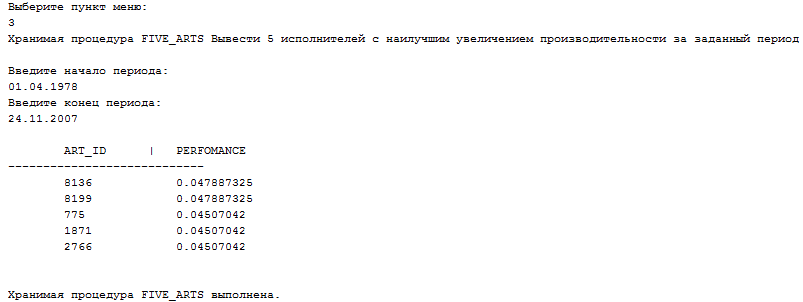


Рис.4. Выполнение хранимой процедуры FIVE\_ARTS.

* + 1. Выполнение любых sql запросов на выборку:

Введем произвольные sql запросы на выборку:



Рис.5. Результат выполнения SELECT (1)

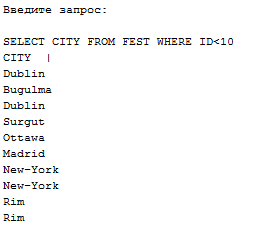


Рис.6. Результат выполнения SELECT (2)

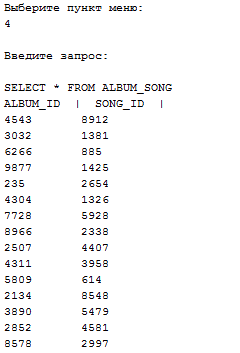
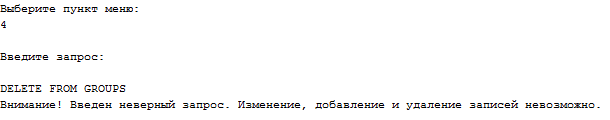
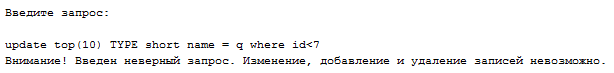


Рис.7. Результат выполнения SELECT (3)

Попробуем ввести запрос на удаление, изменение и добавление:





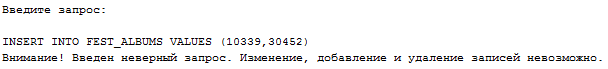


Рис.8. Попытка ввести запросы модификации данных

1. **Вывод:**

В результате выполнения лабораторной работы было разработано клиентское приложение, осуществляющее работу с базой данных, созданной ранее (музыкальная библиотека).

JDBC (Java DataBase Connectivity) – это платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными [СУБД](https://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%91%D0%94), реализованный в виде пакета java.sql, входящего в состав [Java SE](https://ru.wikipedia.org/wiki/Java_SE" \o "Java SE).

Преимуществами JDBC являются:

1) Сочетание JAVA API и JDBC API делает создание приложений лёгким и эффективным.

2) Код приложения подвержен наименьшим изменениям в случае, если происходит смена базы данных.

3) Лёгкость подсоединения к базе через легко описываемый URL.

4) JDBC API полностью предоставляет доступ к метаданным, что позволяет писать сложные приложения

5) Нет необходимости установки специального программного обеспечения, как на стороне клиента, так и на стороне сервера.

6) Драйвера JDBC могут загружаться динамически.

Выполнение данного задания как завершающего в данном цикле лабораторных работ логично, т.к. оно объединяет в себе две изученных платформы для программирования: пройденный ранее язык Java и язык SQL. По сути, была создана простейшая реализация клиент-серверной модели взаимодействия: клиент с помощью программы подключается к базе данных (серверу), изменяет или вводит новые данные, на сервере происходят соответствующие изменения.