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 Чан Ван Кует

(подпись)

Руководитель А.В. Мяснов

(подпись)

“ ” 2016 г.

Санкт-Петербург

2016

1. **Цель работы**

Ознакомиться с разработкой клиентских приложений.

1. **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
* Реализовать импорт данных не менее, чем из двух связанных таблиц из файлов в формате JSON. Вызов хранимой процедуры из приложения. Экспорт данных в формат XML.

1. **Выполнение работы**

**Листинг:**

**libraryfilm.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 libraryfilm;  import org.json.simple.JSONArray;  import org.json.simple.JSONObject;  import org.json.simple.parser.JSONParser;  import org.json.simple.parser.ParseException;  import org.w3c.dom.Document;  import java.io.FileNotFoundException;  import java.io.FileOutputStream;  import java.sql.Connection;  import java.sql.DatabaseMetaData;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  //import java.text.ParseException;  import de.jeckle.RS2DOM.RS2DOM;  import java.io.IOException;  import java.util.Scanner;  import java.util.Vector;  import javax.xml.transform.Transformer;  import javax.xml.transform.TransformerFactory;  import javax.xml.transform.dom.DOMSource;  import javax.xml.transform.stream.StreamResult;  import org.w3c.dom.Node;  /\*\*  \*  \* @author quyettran  \*/  public class Libraryfilm {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args)throws ClassNotFoundException, SQLException,  InstantiationException, IllegalAccessException, FileNotFoundException, ParseException, IOException {  // TODO code application logic here  Connection connect = null;  java.sql.Statement rqst = null; // объект для выполнения SQL запросов  Scanner sc = new Scanner(System.in); // класс для работы с консолью  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 argz3; // входной аргумент для операции INSERT  int argz4; // входной аргумент для операции INSERT  int cnt\_col = 0; //переменная для вывода содержимого таблицы  ResultSet res; // Класс для хранения результатов SQL запроса  Document doc = null; //Класс для хранения XML  // Иницализация драйвера  Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();  //Указание пути к БД  // String strPath = "jdbc:firebirdsql://localhost/D:/library\_films.fdb";  String strPath = "jdbc:firebirdsql://localhost/E:/FILMS.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("| LIBRARY\_FILMS |");  System.out.println("-------------------------------------------");  System.out.println("Функции:");  System.out.println("1.Вывод списка всех таблиц");  System.out.println("2.Добавление записи в таблицу STUDIO");  System.out.println("3.Выполнение хранимой процедуры DEL\_STU");  System.out.println("4.Импорт из JSON данных в GENRES и GENRES\_FILM");  System.out.println("5.Экспорт содержимого таблицы в XML");  System.out.println("6.Выход");  // 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 > 6)  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 В ТАБЛИЦУ ЖАНРОВ  System.out.println("Введите ID ЖАНРА:");  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 название ЖАНРА:");  String argz2 = sc.nextLine();  if (argz2.length()>25 || argz2.isEmpty())  {  System.err.println("Ошибка! Название типа не может быть пустым или больше 25 символов.");  continue;  }  try{  rqst.executeUpdate("insert into STUDIO values ('"+argz+"','"+argz2+"');");  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 DEL\_GENRE удаляет неиспользуемые жанры \n");  PreparedStatement pstmt = connect.prepareStatement("{call DEL\_STU}");  pstmt.execute();  System.out.println("\nХранимая процедура DEL\_GENRE выполнена.");  pstmt.close();  continue;  }  if (menu\_select == 4)  {  JSONParser parser = new JSONParser(); //создание объекта для парсинга  String textjson = filework.read("E:/libraryfilm/src/GENRES\_FILM.json");  Object obj = parser.parse(textjson);  JSONObject jsonObj = (JSONObject) obj;  JSONArray jo = (JSONArray) jsonObj.get("GENRES\_FILM");  //Добавление данных в таблицу cinema  for (int i = 0; i<jo.size();i++){  JSONObject element = (JSONObject) jo.get(i);  try{  rqst.executeUpdate("insert into genres\_film values ('"+element.get("ID")+"','"+element.get("ID\_FILM")+"','"+element.get("ID\_GENRES")+"');");  }catch (SQLException se){  System.out.println(se.getMessage());  }  }  //Добавление данных в таблицу bilet  textjson = filework.read("E:/libraryfilm/src/GENRES.json");  obj = parser.parse(textjson);  jsonObj = (JSONObject) obj;  jo = (JSONArray) jsonObj.get("GENRES");  for (int i=0; i<jo.size();i++){  JSONObject element = (JSONObject) jo.get(i);  try{  rqst.executeUpdate("insert into genres values ('"+element.get("ID")+"','"+element.get("GENRES")+"');");  }catch (SQLException se){  System.out.println(se.getMessage());  }  }  System.out.println("\nИмпорт данных из JSON файлов в таблицы GENRES и GENRES\_FILM выполнен.");  continue;  }  if (menu\_select ==5)  {  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 экспорта в XML:");  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));  Document xsd = RS2DOM.ResultSet2XSDDOM(res);  Document d = RS2DOM.ResultSet2DOM(res);  try {  Transformer myTransformer =  (TransformerFactory.newInstance()).newTransformer();  System.out.println(  "Схема, описывающая XML, экспортирована в файл Description.xml");  myTransformer.transform(  new DOMSource(xsd),  new StreamResult(new FileOutputStream("E:/Description.xml")));  System.out.println(  "\n\nСодержимое таблицы экспортировано в XML файл Data.xml");  myTransformer.transform(  new DOMSource(d),  new StreamResult(new FileOutputStream("E:/Data.xml")));  } catch (Exception e) {  e.printStackTrace();  }  continue;  }  if (menu\_select == 6)  {  System.out.println("ББ");  continue;  }  }  System.exit(0);  }    } |

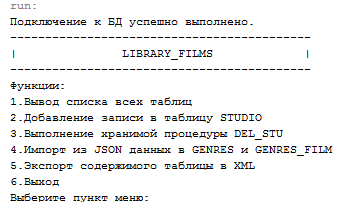
**filework.java**

|  |
| --- |
| **/**\*  \* 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 libraryfilm;  import java.io.BufferedReader;  import java.io.File;  import java.io.FileNotFoundException;  import java.io.FileReader;  import java.io.IOException;  /\*\*  \*  \* @author quyettran  \*/  public class filework {  public static String read(String fileName) throws FileNotFoundException, IOException {  //Этот спец. объект для построения строки  //Определяем файл  File file = new File(fileName);  StringBuilder sb = new StringBuilder();  exists(fileName);  try {  //Объект для чтения файла в буфер  BufferedReader in = new BufferedReader(new FileReader( file.getAbsoluteFile()));  try {  //В цикле построчно считываем файл  String s;  while ((s = in.readLine()) != null) {  sb.append(s);  sb.append("\n");  }  } finally {  //Также не забываем закрыть файл  in.close();  }  } catch(IOException e) {  throw new RuntimeException(e);  }  //Возвращаем полученный текст с файла  return sb.toString();  }  private static void exists(String fileName) throws FileNotFoundException {  File file = new File(fileName);  if (!file.exists()){  throw new FileNotFoundException(file.getName());  }  }  private static String fileName;  } |

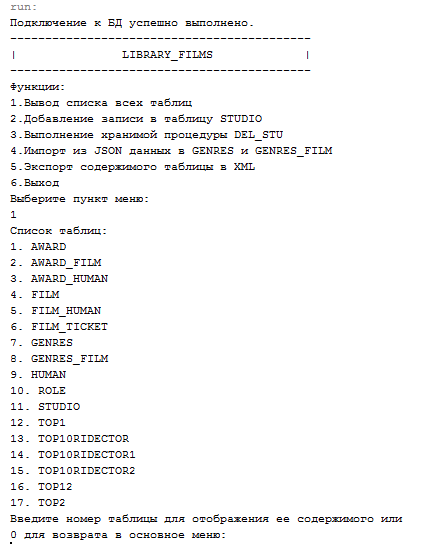
Для импорта данных из файлов JSON использовалась библиотека json-simple. Для экспорта данных в XML использовался класс RS2DOM.

**Примеры работы программы:**

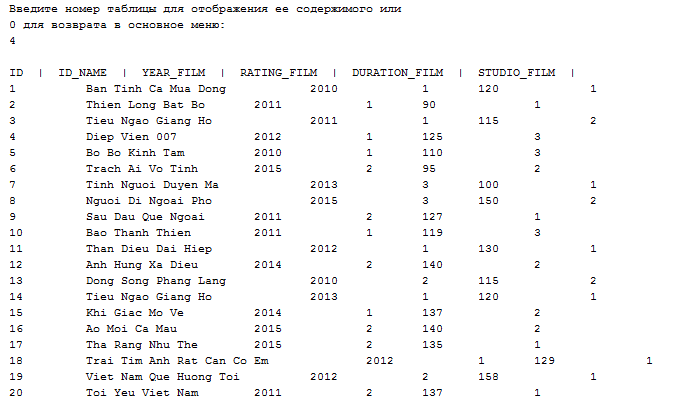
Запуск:



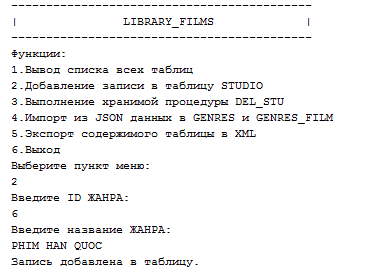
*1.Выполним первый пункт меню: Вывод списка всех таблиц.*



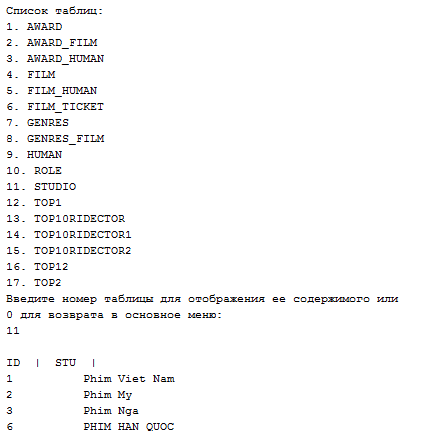
Выберем для вывода содержимого таблицу 4



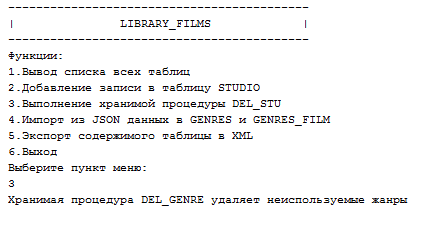
*2. Добавление записи в таблицу STUDIO:*



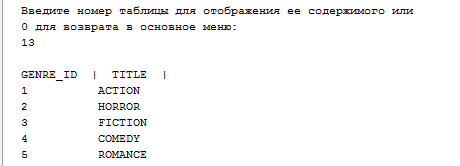
Отобразим содержимое таблицы STUDIO:



*3.Теперь выполним хранимую процедуру* – удаление неиспользуемых жанров в таблице фильмов:



Снова отобразим содержимое таблицы STUDIO:



Наша новая запись, которая не давно добавлена в той таблице, была удалена.

*4.Импорт из JSON данных в CINEMA1 и BILET*

File GENRES\_FILM.json:

|  |
| --- |
| {  "GENRES\_FILM":[  {  "ID" : 30,  "ID\_FILM" : 1 ,  "ID\_GENRES" : 1  }, {  "ID" : 35,  "ID\_FILM" : 2 ,  "ID\_GENRES" : 2  }      ]    } |

File GENRES.json:

|  |
| --- |
| {  "GENRES":[  {  "ID" : 15,  "GENRES" : "Phim hanh dong"  },  {  "ID" : 20,  "GENRES" : "Phim thieu nhi "  }    ]    } |

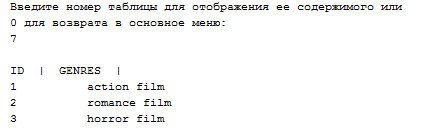
Выполнение функции:

Перед добавлениями:

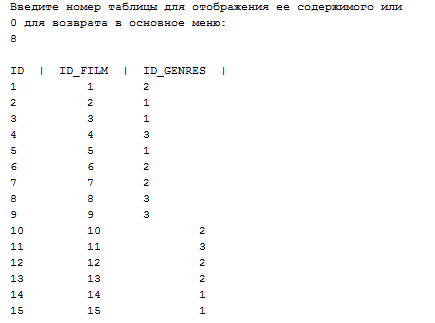
Выполнение функции:

Перед добавлениями:

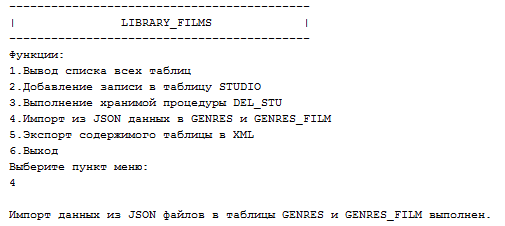
GENRES:



GENRES\_FILM:

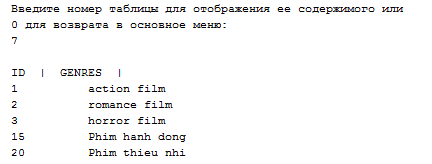


Добавления:

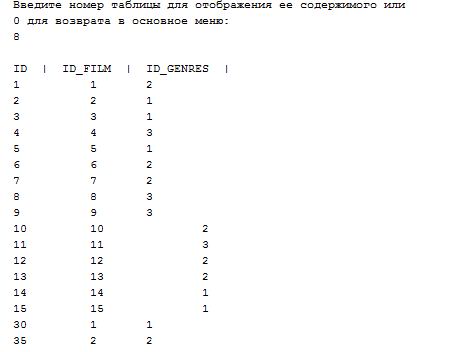


После добавления:

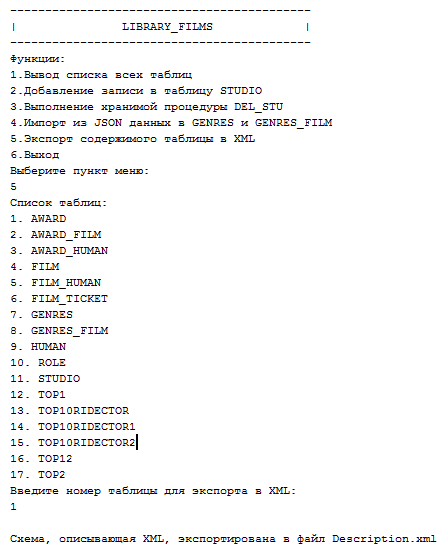
GENRES:



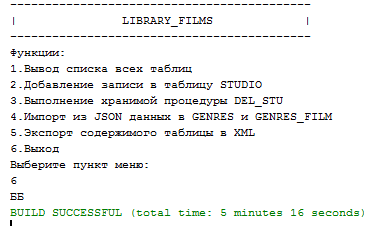
GENRES\_FILM:



*5. Экспорт содержимого таблицы в XML*



*6.Выход:*



1. **Вывод**

В результате выполнения работы было разработано клиентское приложение, осуществляющее некоторые функции для работы с нашей БД.

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.