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 _____ Нгуен Тиен Ву
(подпись)

Руководитель _____ Мяснов А.В.
(подпись)

“__” _____2016 г.

Санкт-Петербург
2016

1. Цель работы.

Ознакомиться с разработкой клиентских приложений.

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
- реализовать импорт данных из JSON файлов
- реализовать экспорт данных в XML

3. Выполнение работы.

Для выполнения работы использовалась IDE NetBeans 8.0.2, в которой было создано консольное приложение Java.

В данном приложении Football использовались следующие классы:

1) Football.java

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

2) Base_Operations.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) ExportXML.java

Данный класс предназначен для экспорта данных таблицы в XML документы.

4) Import_JSON.java

Данный класс осуществляет импорт данных из файлов в формате JSON в таблицы GAME и LEAGUE.

Листинг:

Класс Football.java:

```
package Insurance;
import java.io.FileNotFoundException;
import java.sql.*;
import org.json.simple.parser.ParseException;



// Основной класс для работы с базой данных
public class Insur_company {

   /**
    * @param args the command line arguments
    * @throws java.lang.ClassNotFoundException
    * @throws java.sql.SQLException
    * @throws java.lang.InstantiationException
    * @throws java.lang.IllegalAccessException
    * @throws java.io.FileNotFoundException
    * @throws org.json.simple.parser.ParseException
    */
   public static void main(String[] args) throws ClassNotFoundException,
           SQLException, InstantiationException, IllegalAccessException,
           FileNotFoundException, ParseException
   {
      // Выбранный пункт меню
      int menu_item;
      // Инициализация
      Insurance.initialize();
      // Получение списка таблиц БД
      Insurance.show_table();
      // Процесс работы с меню
      while(true){
         print_menu();
         switch(menu_item = Integer.parseInt(Insurance.sc.nextLine())){
         case 1:
            Insurance.print_table();
            break;
```

```
        case 2:
            Insurance.client_add();
            break;
        case 3:
            Insurance.start_procedure();
            break;
        case 4:
            import_Json.input_JSON();
            break;
        case 5:
            exportXML.get_XML();
            break;
        case 6:
            System.out.println("Завершение работы...");
            System.exit(0);
        default:
            System.err.println("Ошибка! Пункт меню с таким номером"
                +"отсутствует!\n Повторите ввод!\n");
            break;
        }
    }
}

// Функция вывода меню
public static void print_menu(){
    System.out.println("\n\n*****************************************");
    System.out.println("|    INSURANCE COMPANY BASE CONSOLE APPLICATION    |");
    System.out.println("*****************************************");
    System.out.println("Выберите одну из следующих операций:");
    System.out.println("1. Вывод таблицы.");
    System.out.println("2. Добавление записи в таблицу CLIENT.");
    System.out.println("3. Выполнение хранимой процедуры SHO1111.");
    System.out.println("4. Импорт из формата JSON данных в таблицы SUM и CASE.");
    System.out.println("5. Экспорт содержимого таблицы в формат XML.");
    System.out.println("6. Выход из программы.\n");
}
}
```

Insurance.java:

```
package Insurance;

// Класс для выполнения операций с БД
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.util.Scanner;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Insurance {
    // Переменные
    public static String str_1 = null;
    public static String [] str_2 = null;
    public static int cnt_col = 0;
    public static Connection connect = null;
    // Объект для выполнения SQL запросов
    public static java.sql.Statement rqst = null;
    // Класс для работы с консолью
    public static Scanner sc = new Scanner(System.in);
    // Номер введенной таблицы
    public static int table_number = 0;
    public static Vector<String> vec_tab = new Vector<String>();
    //Объект для построения строки
    public static StringBuilder sb = new StringBuilder();
    // Класс для хранения результатов SQL запроса
    public static ResultSet res;
    public static DatabaseMetaData metaData;
    public static ResultSet reslt;
    private static int insrt1;
```

```
// Инициализация базы данных
public static void initialize(){
   try {
      try {
         // Инициализация драйвера
         Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();
      } catch (InstantiationException | IllegalAccessException ex) {
         Logger.getLogger(Insurance.class.getName()).log(Level.SEVERE, null, ex);
      }

      //Указание пути к БД
      String strPath = "jdbc:firebirdsql://localhost/"
            + "E://TIENVU.fdb";
      try {
         Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();
      } catch (InstantiationException | IllegalAccessException ex) {
         Logger.getLogger(Insurance.class.getName()).log(Level.SEVERE, null, ex);
      }

      try {
         //Подключение к БД
         connect = DriverManager.getConnection(strPath, "SYSDBA", "masterkey");
      } catch (SQLException ex) {
         Logger.getLogger(Insurance.class.getName()).log(Level.SEVERE, null, ex);
      }
      if (connect == null) {
         System.err.println("Ошибка при подключении к базе данных!");}

      try {
         //Создание класса для выполнения SQL запросов
         rqst = connect.createStatement();
      } catch (SQLException ex) {
         Logger.getLogger(Insurance.class.getName()).log(Level.SEVERE, null, ex);
      }
      System.out.println("Подключение к БД выполнено успешно!");
   } catch (ClassNotFoundException ex) {
      Logger.getLogger(Insurance.class.getName()).log(Level.SEVERE, null, ex);
   }
```

```
}
// Вывод таблиц
public static void show_table(){
   try {
      metaData = connect.getMetaData();
      reslt=metaData.getTables(str_1, str_1, str_1, str_2);
      while(reslt.next())
      {
         str_1=reslt.getString(3);
         if(!str_1.contains("$"))
            vec_tab.add(str_1);
      }
   } catch (SQLException ex) {
      Logger.getLogger(Insurance.class.getName()).log(Level.SEVERE, null, ex);
   }
}

// Печать таблиц
public static void print_table(){
   try
   {
      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!");
         return;
      }
```

```
if((table_number > vec_tab.size()) || (table_number < 0)){
   System.err.println("Ошибка! Несуществующий номер таблицы!");
   return;
}
if(table_number == 0)
{
   return;
}

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
// Вывод названия столбцов:
for(int i = 1; i < cnt_col + 1; i++){
   System.out.print(res.getMetaData().getColumnName(i)+
         "  |  ");
}
// Вывод записей 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 ex)
    {
        Logger.getLogger(Insurance.class.getName()).log(Level.SEVERE, null, ex);
    }
}

// Добавление в таблицу нового клуба
public static void client_add(){
    // Ввод аргументов для добавления нового клиент в таблицу
    // Ввод названия
    System.out.println("Введите CLIENT_ID:");
    String insrt1 = sc.nextLine();
    if (insrt1.length()>25 || insrt1.isEmpty())
    {
        System.err.println("Ошибка! Проверьте правильность ввода! "
                + "Название должно быть короче 25 символов");
        return;
    }

    // Ввод тренера
    System.out.println("Введите имя Клиента:");
    String insrt2 = sc.nextLine();
    if (insrt2.length()>25 || insrt2.isEmpty())
    {
        System.err.println("Ошибка! Проверьте правильность ввода! "
                + "Название должно быть короче 25 символов");
        return;
    }

    // Ввод владельца клуба
    System.out.println("Введите день рождения:");
    String insrt3 = sc.nextLine();
    if (insrt3.length()>25 || insrt3.isEmpty())
    {
        System.err.println("Ошибка! Проверьте правильность ввода! "
                + "Название должно быть короче 25 символов");
        return;
```

```
    }

    // Ввод сайта клуба
    System.out.println("Введите адресса клиента:");
    String insrt4 = sc.nextLine();
    if (insrt4.length()>25 || insrt4.isEmpty())
    {
       System.err.println("Ошибка! Проверьте правильность ввода! "
                    + "Название должно быть короче 25 символов");
       return;
    }

    // Ввод данных в таблицу club
    try
    {
       rqst.executeUpdate("INSERT INTO client VALUES("
                + "'" + insrt1+ "','" + insrt2 + "','" + insrt3
                + "','" + insrt4 + "'" + ");");
       System.out.println("Запись добавлена в таблицу!\n");
    }
    catch (SQLException se)
    {
       System.out.println(se.getMessage());
    }
}
// Функция вызова хранимой процедуры
public static void start_procedure(){
    try {
       System.out.println("Хранимая процедура сумму страховых вносов и сумму выплат");
       // Ввод аргументов хранимой процедуры
       // Ввод даты начала тура
       System.out.println("Введите клиета:");
       String client = sc.nextLine();
       // Ввод даты конца тура
       System.out.println("Введите сумма стпаховых вносов :");
       String price = sc.nextLine();
       // Ввод лиги для обновления
       System.out.println("Введите сумму выплат:");
```

```
            String payment = sc.nextLine();
            try (PreparedStatement pstmt = connect.prepareStatement("{call SHO1111('"+client
                +"','"+price+"','"+payment+"')}")) {
               pstmt.execute();
               System.out.println("Хранимая процедура SHO1111 успешно выполнена.");
            }
        } catch (SQLException ex) {
            Logger.getLogger(Insurance.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Класс exportXML.java:

```
package Insurance;

import static Insurance.Insurance.res;
import static Insurance.Insurance.rqst;
import static Insurance.Insurance.sc;
import static Insurance.Insurance.table_number;
import static Insurance.Insurance.vec_tab;
import de.jeckle.RS2DOM.RS2DOM;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Document;

// Класс для экспорта таблиц в XML
public class exportXML {
    //Класс для хранения XML
    public static Document doc = null;
    public static void get_XML(){
```

```
try
{
   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!");
         return;
      }
      if((table_number > vec_tab.size()) || (table_number < 0)){
         System.err.println("Ошибка! Таблица с таким номером отсутсвует.");
         return;
      }
      if(table_number == 0){
         return;
      }
      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 Schema.xml");
         myTransformer.transform(
               new DOMSource(xsd),
               new StreamResult(new FileOutputStream("C:"
```

```
                        + "\\Users\\Stud\\Desktop\\1_course4\\Information devision Systems and
base date\\lab8\\Schema.xml")));
            System.out.println(
                "\nСодержимое таблицы экспортировано в XML файл Data.xml");
            myTransformer.transform(
                new DOMSource(d),
                new StreamResult(new FileOutputStream("C:"
                    + "\\Users\\Stud\\Desktop\\1_course4\\Information devision Systems and
base date\\lab8\\Data.xml")));
        } catch (FileNotFoundException | TransformerException e) {
        }
    }
    catch(SQLException ex)
    {
      Logger.getLogger(Insurance.class.getName()).log(Level.SEVERE, null, ex);
    }
  }
}
```

Класс import_JSON.java:

```
package Insurance;

import static Insurance.Insurance.rqst;
import java.io.FileNotFoundException;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

// Класс для работы с JSON файлами
public class import_Json {
   // Объекты класса JSON
   public static JSONParser parser = new JSONParser();
   public static Object obj;
   public static String textjson;
```

```
    public static JSONObject jsonObj;
    public static JSONArray jo;
    // Адреса файлов JSON
    private static final String sum_dir = "C:\\Users\\Stud\\Documents\\MEGA\\MEGAsync update\\1_course4\\Information devision Systems and base date\\koncolnoe prilorenie\\Insurance\\src\\de\\jeckle\\RS2DOM\\SUM.json";
    //private static final String sum_dir = "C:\\Users\\Stud\\Desktop\\lab8\\SUM.json";
    private static final String case_dir = "C:\\Users\\Stud\\Documents\\MEGA\\MEGAsync update\\1_course4\\Information devision Systems and base date\\koncolnoe prilorenie\\Insurance\\src\\de\\jeckle\\RS2DOM\\CASE.json";

    // Функция импорта файлов JSON
    public static void input_JSON(){
        try {
            textjson = Running.read(sum_dir);
            obj = null;
            try {
                obj = parser.parse(textjson);
            } catch (ParseException ex) {
                Logger.getLogger(Insurance.class.getName()).log(Level.SEVERE, null, ex);
            }
            jsonObj = (JSONObject) obj;
            jo = (JSONArray) jsonObj.get("SUM");

            // Добавление данных в таблицу SUM
            for (int i = 0; i<jo.size();i++)
            {
                JSONObject element = (JSONObject) jo.get(i);
                try
                {
                    System.out.println("insert into \"SUM\" values (" + element.get("ID_SUM")
                            + "," + element.get("SUM_PAY")+");");
                    rqst.executeUpdate("insert into \"SUM\" values (" + element.get("ID_SUM")
                            + "," + element.get("SUM_PAY")+");");
//                  rqst.executeUpdate("insert into SUM values (" + element.get("ID_SUM"))
//                          + "'," + element.get("SUM_PAY") );
                }
                catch (SQLException se)
                {
```

```
                System.out.println(se.getMessage());
            }
        }
        System.out.println("Добавление данных в таблицу SUM из формата JSON успешно выполнено.");

        // Добавление данных в таблицу CASE
        textjson = Running.read(case_dir);
        try {
            obj = parser.parse(textjson);
        } catch (ParseException ex) {
            Logger.getLogger(Insurance.class.getName()).log(Level.SEVERE, null, ex);
        }
        jsonObj = (JSONObject) obj;
        jo = (JSONArray) jsonObj.get("CASE");
        for (int i = 0; i<jo.size();i++)
        {
            JSONObject element = (JSONObject) jo.get(i);
            try
            {
                System.out.println("insert into \"CASE\" values (" + element.get("ID_CASE")
                        + "," + element.get("CASE_IN")+ ","
                        + element.get("SUM_PAY")+");");
                rqst.executeUpdate("insert into \"CASE\" values (" + element.get("ID_CASE")
                        + "," + element.get("CASE_IN")+ ","
                        + element.get("SUM_PAY")+");");
            }
            catch (SQLException se)
            {
                System.out.println(se.getMessage());
            }
        }
        System.out.println("Добавление данных в таблицу CASE из формата JSON успешно выполнено.");
    } catch (FileNotFoundException ex) {
        Logger.getLogger(Insurance.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

```
}
```

Класс exportXML.java:

```
package Insurance;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

// Класс для работы с файлами
public class Running {
    // Объекты
    public static File file;
    public static StringBuilder sb;
    public static BufferedReader in;
    // Функция чтения файла
    public static String read(String fileName) throws FileNotFoundException {
    file = new File(fileName);
    sb = new StringBuilder();
    exists(fileName);
    try {
        //Объект для чтения файла в буфер
        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 {
            //Закрытие файла
            in.close();
        }
    } catch(IOException e) {
        throw new RuntimeException(e);
    }
```

```
    //Возвращаем полученный текст с файла
    return sb.toString();
    }
    // Проверка файла на существование
    private static void exists(String fileName) throws FileNotFoundException {
        file = new File(fileName);
        if (!file.exists()){
            throw new FileNotFoundException(file.getName());
        }
    }
}
```

Теперь рассмотрим взаимодействие приложения с пользователем:

При запуске программы выводится меню возможных действий с базой данных:

```
****************************************
|    FOOTBALL BASE CONSOLE APPLICATION    |
****************************************
Выберите одну из следующих операций:
Выберите одну из следующих операций:
1. Вывод таблицы.
2. Добавление записи в таблицу CLIENT.
3. Выполнение хранимой процедуры SHO1111.
4. Импорт из формата JSON данных в таблицы SUM и CASE.
5. Экспорт содержимого таблицы в формат XML.
6. Выход из программы.
Выберите команду:
```

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

Рассмотрим действия, которые может выполнять созданная программа:

1. Вывод списка всех таблиц базы данных:

```
1
Список таблиц:
1. CASE
2. CLIENT
3. INSURANCE_CASE_CON
4. INSURANCE_CASE_POLIC
5. MAXPRICE
6. POLIC
7. PRICE_INS
8. SIGN_CONTRACT
9. SUM
10. SUM_PRICE
11. TYPE
Введите номер таблицы для отображения ее содержимого или
0 для возврата в основное меню:
4

ID_CASE_POLIC  |  ID_POLIC  |  ID_CASE  |
1       1          1
2       1          2
3       1          3
4       1          4
5       1          5
6       2          1
7       2          2
8       3          2
9       3          3
10      4          4
11      4          5
12      5          1
13      6          2
14      7          3
15      8          4
```

2. Добавление записи в таблицу CLIENT:

```
2
Введите CLIENT_ID:
12675
Введите имя Клиента:
CALES PYOL
Введите день рождения:
08-12-1992
Введите адресса клиента:
BARCELONA
Запись добавлена в таблицу!
```

Воспользуемся приложением и посмотрим на таблицу CLIENT:

Как видно, запись была успешно добавлена в таблицу.

3. Выполнение хранимой процедуры SHO1111:

Процедура SHO1111 обновляет клиенты вывести сумму страховых вносов и сумму выплат

```
3
Хранимая процедура update_league обновляет таблицу очков команд в лиге после
очередного тура.

Хранимая процедура сумму страховых вносов и сумму выплат
Введите клиета:
1234
Введите сумма стпаховых вносов :
12000
Введите сумму выплат:
299990
Хранимая процедура SHO1111 успешно выполнена.
```

4. Импорт из JSON данных в таблицы SUM и CASE.

Файл данных SUM.json:

```
{
      "SUM" :
   [
       { "ID_SUM" : "11575" , "SUM_PAY" : "70000" }
   ]


}
```

Файл данных CASE.json:

```
   {
         "CASE":[
          { "ID_CASE" : "14" , "CASE_IN" : "'DDUILD'" , "SUM_PAY" :
   "100000" }
      ]


   }
```

Далее выполним соответствующий пункт меню программы:

```
   4
   insert into "SUM" values (11575,70000);
   Добавление данных в таблицу SUM из формата JSON успешно выполнено.
   insert into "CASE" values (14,'DDUILD',100000);
   Добавление данных в таблицу CASE из формата JSON успешно выполнено.
```

Для простоты проверки – посмотрим из консольного приложения на таблицу league:

5. Экспорт содержимого таблицы в XML

После выбора данного пункта меню на экран будет выведен список всех имеющихся таблиц. После ввода номера таблицы, содержимое таблицы и схема, описывающая XML будут экспортированы в соответствующие файлы.

```
5
Список таблиц:
1. CASE
2. CLIENT
3. INSURANCE_CASE_CON
4. INSURANCE_CASE_POLIC
5. MAXPRICE
6. POLIC
7. PRICE_INS
8. SIGN_CONTRACT
9. SUM
10. SUM_PRICE
11. TYPE
Введите номер таблицы для экспорта в XML:
6
Схема, описывающая XML, экспортирована в файл Schema.xml
Содержимое таблицы экспортировано в XML файл Data.xml
```

Содержимое файла Data.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><result
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><row><ID_POLIC>1</ID_POLIC><NAME_POLIC>ALL</NAME_POLIC><
PRICE>30000</PRICE><PERIOD>2015-03-
12</PERIOD></row><row><ID_POLIC>2</ID_POLIC><NAME_POLIC>CAR+PRO</N
AME_POLIC><PRICE>12000</PRICE><PERIOD>2014-04-
12</PERIOD></row><row><ID_POLIC>3</ID_POLIC><NAME_POLIC>PRO+MED</N
AME_POLIC><PRICE>13000</PRICE><PERIOD>2015-06-
15</PERIOD></row><row><ID_POLIC>4</ID_POLIC><NAME_POLIC>TRA+ACC</N
AME_POLIC><PRICE>20000</PRICE><PERIOD>2014-05-
14</PERIOD></row><row><ID_POLIC>5</ID_POLIC><NAME_POLIC>CAR</NAME_
POLIC><PRICE>5000</PRICE><PERIOD>2015-05-
14</PERIOD></row><row><ID_POLIC>6</ID_POLIC><NAME_POLIC>PROPERTY</
NAME_POLIC><PRICE>8000</PRICE><PERIOD>2014-03-
12</PERIOD></row><row><ID_POLIC>7</ID_POLIC><NAME_POLIC>MEDICINE</
NAME_POLIC><PRICE>6000</PRICE><PERIOD>2015-05-
12</PERIOD></row><row><ID_POLIC>8</ID_POLIC><NAME_POLIC>TRAVELS</N
AME_POLIC><PRICE>10000</PRICE><PERIOD>2015-03-
14</PERIOD></row><row><ID_POLIC>9</ID_POLIC><NAME_POLIC>ACCIDENT</
NAME_POLIC><PRICE>15000</PRICE><PERIOD>2014-05-
13</PERIOD></row><row><ID_POLIC>10</ID_POLIC><NAME_POLIC>ACCIDENT<
/NAME_POLIC><PRICE>15000</PRICE><PERIOD>2015-03-
12</PERIOD></row></result>
```

Содержимое файла Schema.xml:

```
 <?xml version="1.0" encoding="UTF-8" standalone="no"?><schema
attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns="http://www.w3.org/2001/XMLSchema"><element
name="result"><complexType><sequence><element maxOccurs="unbounded"
name="row"><complexType><sequence><element name="ID_POLIC"
type="integer"/><element name="NAME_POLIC" nillable="true"
type="string"/><element name="PRICE" type="integer"/><element name="PERIOD"
nillable="true"
type="date"/></sequence></complexType></element></sequence></complexType></ele
ment></schema>>
```

Из полученных файлов видно, что все данные были экспортированы корректно.

4. Вывод:

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

JDBC (Java DataBase Connectivity) – это платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета java.sql, входящего в состав Java SE.

Было очень интересно впервые создавать приложение, использующее базу данных. Я думаю, что в будущем практически каждое моё приложение будет содержать базу данных, и поэтому это очень важный опыт. К сожалению, у меня не было достаточно времени, чтобы создать WEB или GUI приложение. Однако я обязательно попробую сделать это, как только будет возможность.