Санкт-Петербургский государственный политехнический университет
Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе**

**Дисциплина**: Базы данных

**Тема**: Разработка клиентского приложения

Выполнил студент гр. 43501/1 _____ А.О. Перешеин
(подпись)
Руководитель _____ А.В. Мяснов
(подпись)

“__” _____ 2015 г.

Санкт-Петербург
2015

## 1. Цель 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

## 2. Выполнение работы

Сначала в меню Services в NetBeans необходимо добавить в папку Drivers jar-файл драйвера Jaybird версии full (в нашем случае это jaybird-full-2.2.6.jar).

В самом приложении необходимо сначала инициализировать Firebird JDBC driver:
Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();

Затем выполнить подключение к БД, указав путь, пользователя и пароль:
Connect connect = DriverManager.getConnection(strPath, strUser, strPass);

Создадим класс для выполнения SQL запросов:
java.sql.Statement stmt = connect.createStatement();

Запросы будем выполнять используя следующую функцию, в аргументах указав сам запрос:
res = stmt.executeQuery("select * from <…>");

Возвращаемое значение res может использоваться для получения, например, имен атрибутов или их количество в таблице, получившейся в результате выполнения select:
res.getMetaData().getColumnCount();
res.getMetaData().getColumnName(i)

Для выполнения запросов insert, update, delete применяется следующая функция:
executeUpdate("insert into <…> values (<…>)");

Для выполнения хранимой процедуры при наличии у нее входных параметров используется класс PreparedStatement:

```
PreparedStatement pstmt = connect.prepareStatement("{call lab5_p1(?,?)}");
```

Задание входных параметров:

```
pstmt.setString(1, sc.next());
pstmt.setInt(2, sc.nextInt());
```

Исполнение ХП:

```
pstmt.execute();
```

Хранимая процедура, реализованная в приложении: все диски, которые были проданы более заданного количества раз, наградить заданной наградой.

В итоге созданное клиентское приложение обладает следующей функциональностью:

- подключение к БД
- выполнение запроса select для любой таблицы
- выполнение запроса insert для любой таблицы
- запуск определенной хранимой процедуры

## 3. Результаты экспериментов

### Подключение к БД:

МЕНЮ:
1) Подключение к БД (connect)
2) Вывести содержимое таблицы (select)
3) Добавить запись в таблицу (insert)
4) Выполнить ХП (execute)
5) Выход (exit)
Выберите пункт:
1
Введите путь к локальному серверу БД
шаблон для ввода: D:\folder\database.fdb:
D:\DBs\musicdb.fdb
Введите имя пользователя: SYSDBA

Введите пароль: masterkey
Подключение к БД успешно выполнено

### Вывод содержимого таблиц:

МЕНЮ:
1) Подключение к БД (connect)
2) Вывести содержимое таблицы (select)
3) Добавить запись в таблицу (insert)
4) Выполнить ХП (execute)
5) Выход (exit)
Выберите пункт:
2
Вывод содержимого таблицы (select)
Введите название таблицы:
CDs

| CODE | RELEASE_YEAR | SOLD | STATUS | AWARD |
|---|---|---|---|---|
| 679137227 | 1990 | 100000000 | 3 | mtv_award |
| 781676650 | 2000 | 10000000 | 3 | grammy |
| 329751487 | 1986 | 200000000 | 3 | mtv_award |
| 105520746 | 2013 | 1700000 | 3 | grammy |
| 714679272 | 1996 | 600000 | 1 | grammy |

МЕНЮ:
1) Подключение к БД (connect)
2) Вывести содержимое таблицы (select)
3) Добавить запись в таблицу (insert)
4) Выполнить ХП (execute)
5) Выход (exit)
Выберите пункт:
2
Вывод содержимого таблицы (select)
Введите название таблицы:
Artists

| ART_ID | ARTIST |
|---|---|
| 1 | Pantera |
| 2 | Amorphis |
| 3 | Slayer |
| 4 | Ice Cube |
| 5 | Edge of Sanity |

### Добавление записи:

МЕНЮ:

1) Подключение к БД (connect)
2) Вывести содержимое таблицы (select)
3) Добавить запись в таблицу (insert)
4) Выполнить ХП (execute)
5) Выход (exit)
Выберите пункт:
3
Добавление записи в таблицу (insert)
Введите название таблицы:
Artists
Введите значения аттрибутов через запятую без пробелов
ART_ID, ARTIST:
6,'Converge'
Запись успешно добавлена

МЕНЮ:
1) Подключение к БД (connect)
2) Вывести содержимое таблицы (select)
3) Добавить запись в таблицу (insert)
4) Выполнить ХП (execute)
5) Выход (exit)
Выберите пункт:
2
Вывод содержимого таблицы (select)
Введите название таблицы:
Artists

| ART_ID | ARTIST |
|---|---|
| 1 | Pantera |
| 2 | Amorphis |
| 3 | Slayer |
| 4 | Ice Cube |
| 5 | Edge of Sanity |
| 6 | Converge |

## Выполнение ХП:

МЕНЮ:
1) Подключение к БД (connect)
2) Вывести содержимое таблицы (select)
3) Добавить запись в таблицу (insert)
4) Выполнить ХП (execute)
5) Выход (exit)
Выберите пункт:
4
Выполнение ХП (execute)
[все диски, которые были проданы более заданного
количества раз, наградить заданной наградой]
Введите первый параметр (награда): award

Введите второй параметр (число продаж): 99000000

ХП успешно выполнена

МЕНЮ:
1) Подключение к БД (connect)
2) Вывести содержимое таблицы (select)
3) Добавить запись в таблицу (insert)
4) Выполнить ХП (execute)
5) Выход (exit)
Выберите пункт:

2
Вывод содержимого таблицы (select)
Введите название таблицы:
CDs

| CODE | RELEASE_YEAR | SOLD | STATUS | AWARD |
|---|---|---|---|---|
| 679137227 | 1990 | 100000000 | 3 | award |
| 781676650 | 2000 | 10000000 | 3 | grammy |
| 329751487 | 1986 | 200000000 | 3 | award |
| 105520746 | 2013 | 1700000 | 3 | grammy |
| 714679272 | 1996 | 600000 | 1 | grammy |

## 4. Листинг программы

```
package javadb;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) throws ClassNotFoundException, InstantiationException,
IllegalAccessException, SQLException {
        // 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 var = 0;        // переменная = выбранный пункт меню
        String table;       // имя таблицы для запроса
        int column;         // кол-во столбцов в текущей таблице
        String argz;        // аргументы запроса insert (атрибуты таблицы)
        // создаем класс для хранения результата запроса
        ResultSet res;
        // цикл для вывода меню и выполнения операций с консоли
        while (var != 5) {
            // вывод меню
            System.out.println("\n\n\tМЕНЮ:");
            System.out.println("1) Подключение к БД (connect)");
            System.out.println("2) Вывести содержимое таблицы (select)");
            System.out.println("3) Добавить запись в таблицу (insert)");
            System.out.println("4) Выполнить ХП (execute)");
            System.out.println("5) Выход (exit)");
            // считывание номера пункта меню
            System.out.println("Выберите пункт:");
            var = sc.nextInt();
            switch(var) {
                case 1:
                    // инициализируем Firebird JDBC driver
                    Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();
                    //путь к базе данных
                    String strPath = "jdbc:firebirdsql://localhost/";
                    System.out.println("Введите путь к локальному серверу БД");
                    System.out.println("шаблон для ввода: D:\\folder\\database.fdb:");
                    strPath = strPath + sc.next();      // "D:\\DBs\\musicdb.fdb"
                    // логин, с помощью которого подключаемся к базе данных
                    System.out.print("Введите имя пользователя: ");
                    String strUser = sc.next();         // "SYSDBA"
                    // пароль
                    System.out.print("\nВведите пароль: ");
                    String strPass = sc.next();         // "masterkey"
                    // инициализируем Firebird JDBC driver
                    Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();
```

```
            // создаём подключение к базе данных
            connect = DriverManager.getConnection(strPath, strUser, strPass);
            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");
            continue;
        case 2:
            if(connect == null) {
                System.err.println("Соединение с БД не установлено");
                continue;
            }
            System.out.println("Вывод содержимого таблицы (select)");
            System.out.println("Введите название таблицы:");
            table = sc.next();
            // выполнение запроса
            res = stmt.executeQuery("select * from " + table + ";");
            // смотрим количество колонок в результате SQL запроса
            column = res.getMetaData().getColumnCount();
            // выводим названия атрибутов
            for (int i = 1; i <= column; i++) {
                System.out.printf("%15s", res.getMetaData().getColumnName(i));
            }
            // выводим содержимое таблицы
            while(res.next()) {
                System.out.println();
                for (int i = 1; i <= column; i++) {
                    Object obj = res.getObject(i);
                    if (obj != null) {
                        System.out.printf("%15s", obj);
                    }
                }
            }
            continue;
        case 3:
            if(connect == null) {
                System.err.println("Соединение с БД не установлено");
                continue;
            }
            System.out.println("Добавление записи в таблицу (insert)");
            System.out.println("Введите название таблицы:");
            table = sc.next();
            // узнаем содержимое таблицы
            res = stmt.executeQuery("select * from " + table + ";");
            // узнаем кол-во атрибутов
            column = res.getMetaData().getColumnCount();
            // вводим аргументы insert
            System.out.println("Введите значения аттрибутов через запятую без пробелов");
            for(int i = 1; i <= column; i++) {
                if(i != column)
                    System.out.print(res.getMetaData().getColumnLabel(i) + ", ");
                else
                    System.out.println(res.getMetaData().getColumnLabel(i) + ":");
            }
            argz = sc.next();
            stmt.executeUpdate("insert into " + table + " values (" + argz + ");");
            System.out.println("Запись успешно добавлена");
            continue;
        case 4:
            if(connect == null) {
                System.err.println("Соединение с БД не установлено");
                continue;
            }
            System.out.println("Выполнение ХП (execute)");
            System.out.println("[все диски, которые были проданы более заданного");
            System.out.println("количества раз, наградить заданной наградой]");
            PreparedStatement pstmt = connect.prepareStatement("{call lab5_p1(?,?)}");
```

```
                    System.out.print("Введите первый параметр (награда): ");
                    pstmt.setString(1, sc.next());
                    System.out.print("\nВведите второй параметр (число продаж): ");
                    pstmt.setInt(2, sc.nextInt());
                    pstmt.execute();
                    System.out.println("\nХП успешно выполнена");
                    pstmt.close();
                    continue;
                case 5:
                    System.out.println("Выход (exit)");
                    continue;
                default:
                    System.err.println("Некорректный выбор пункта меню");
            }
        }
        // освобождаем ресурсы
        if(connect != null)
            connect.close();
        if(stmt != null)
            stmt.close();
    }
}
```

## 5. Выводы

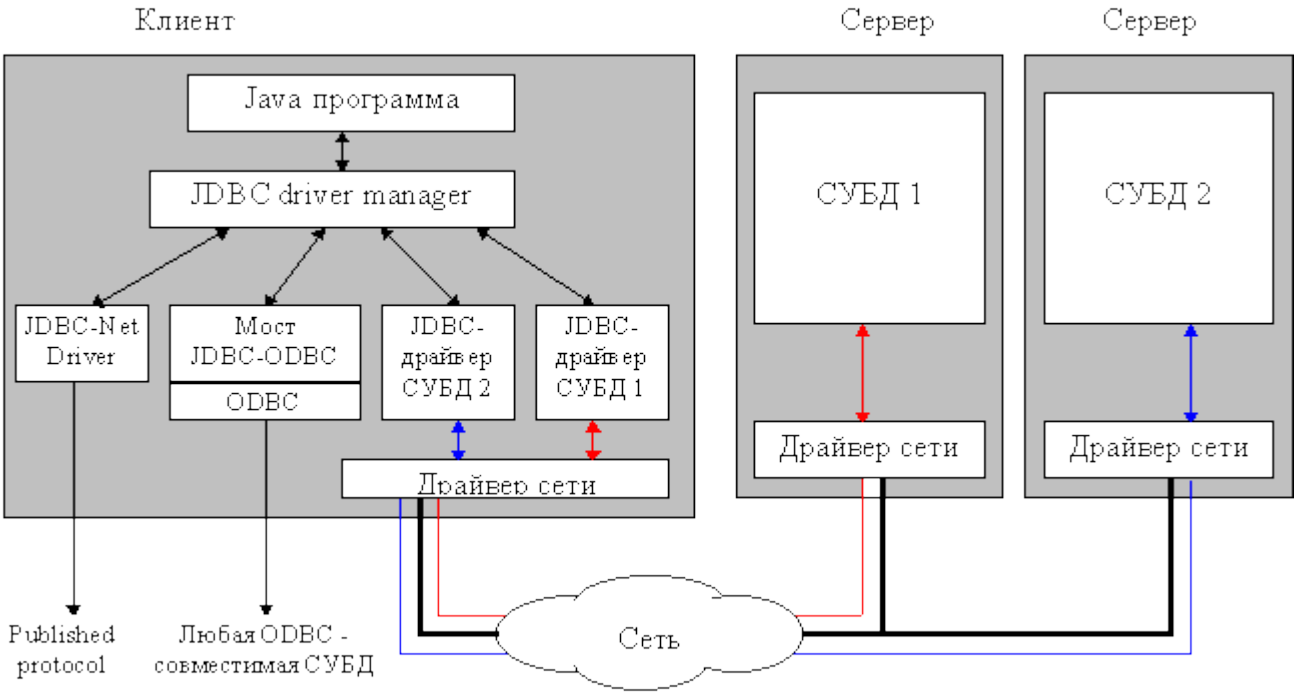
Трёхуровневая архитектура (трёхзвенная архитектура) - архитектурная модель программного комплекса, предполагающая наличие в нём трёх компонентов: клиента, сервера приложений (к которому подключено клиентское приложение) и сервера баз данных (с которым работает сервер приложений).

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.

Минусы такой архитектуры:
- 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.
JDBC (Java DataBase Connectivity) - это интерфейс прикладного программирования (API) для выполнения SQL-запросов к базам данных из программ, написанных на языке Java. Язык

Java, созданный компанией Sun, является платформенно - независимым и позволяет создавать как собственно приложения (standalone application), так и программы (апплеты), встраиваемые в web-страницы.




JDBC во многом подобен ODBC (см. рисунок), также построен на основе спецификации CLI, однако имеет ряд замечательных отличий. Во-первых, приложение загружает JDBC-драйвер динамически, следовательно администрирование клиентов упрощается, более того, появляется возможность переключаться на работу с другой СУБД без перенастройки клиентского рабочего места. Во-вторых, JDBC, как и Java в целом, не привязан к конкретной аппаратной платформе, следовательно проблемы с переносимостью приложений практически снимаются. В-третьих, использование Java-приложений и связанной с ними идеологии "тонких клиентов" обещает снизить требования к оборудованию клиентских рабочих мест.