**ЛАБОРАТОРНАЯ РАБОТА №4**

# Работа с базами данных в Java

Цель:Научиться создавать приложения трехзвенной архитектуры, взаимодействующие с базой данных.

## **Основные понятия баз данных**

База данных - набор сведений, хранящихся некоторым упорядоченным способом.

Можно сравнить базу данных со шкафом, в котором хранятся документы. Иными словами, база данных - это хранилище данных. Сами по себе базы данных не представляли бы интереса, если бы не было систем управления базами данных (СУБД).

Выделяют следующие типы структур баз данных: реляционная, иерархическая, сетевая.

|  |  |
| --- | --- |
| Реляционная |  |
| Иерархическая |  |
| Сетевая |  |

Рис.1. Типы структур БД

Для **иерархических структур** характерна подчиненность объектов нижнего уровня объектам верхнего уровня. В дереве, между верхними и нижними объектами, задано отношение **«один ко многим».** Исходные элементы порождают подчиненные.

**Сети** имеют много уровней взаимосвязанных объектов, между которыми задано отношение **«многие ко многим».** Сетевая организация обладает большей гибкостью и облегчает процесс поиска требуемых данных.

**Реляционные базы данных** получили наибольшее распространение, т.к. они обладают преимуществом - наглядность и понятность для пользователя табличной структуры. К реляционной структуре можно свести любой тип структуры данных (деревья и сети). Название “реляционная” (от *relational* - отношение) связано с тем, что каждая запись в таблице содержит информацию, относящуюся только к одному конкретному объекту.

Чаще всего база данных строится на основе нескольких таблиц, связанных между собой.

В любой БД есть таблицы. ***Таблицы*** непосредственно хранят информацию, относящуюся к конкретной предметной области.

|  |  |
| --- | --- |
| Реляционная алгебра (логическая модель) | Реляционная база данных (физическая модель) |
| Сущность  Кортеж  Атрибут  Домен | Таблица  Строка (Запись)  Столбец (Поле)  Тип данных |

Сущность – это объект предметной области, которая исследуется и моделируется.

Кортеж – это экземпляр объекта предметной области (экземпляр сущности).

Атрибут отражает определенное свойство, качество, признак сущности.

Домен задает множество допустимых значений атрибута.

В любой таблице должен быть ключ. Ключ или потенциальный ключ – это минимальный набор атрибутов, по значениям которых можно однозначно выбрать требуемый экземпляр сущности.

По ограничению целостности данных ключи делятся на:

Первичный ключ – служит как ограничение целостности в рамках одной таблицы для однозначной идентификации, конкретно поле первичного ключа не может повторятся или быть пустым.

Внешний ключ(вторичный) – служит как ограничение целостности связей нескольких таблиц, конкретно подчиненная таблица не может ссылаться на несуществующие записи главной таблицы (что позволяет строить целостные модели данных).

**Виды отношений:**

а) идентифицирующее отношение

Сущность А1 однозначно определяет сущность А2. Ее первичный ключ наследуется в качестве первичного ключа сущностью А2 (внешний ключ)



Рис.2. Идентифицирующее отношение

б) неидентифицирующее отношение

Сущность А1 связана с сущностью А2, но однозначно не определяет ее. Первичный ключ сущности А1 наследуется в качестве неключевого атрибута сущности А2



Рис.3. Неидентифицирующее отношение

в) отношение «многие-ко-многим»

Сущности А1 и А2 имеют формальную связь, но наследования атрибутов не происходит.



Рис.4. Отношение «многие-ко-многим»

**Примеры основных SQL команд**

SQL (Structured Query Language) - это язык, являющийся стандартом ANSI, предназначенный для работы с базами данных.

SQL работает с такими базами данных, как Access, DB2, MySQL, Microsoft SQL Server, Oracle, Sybase и многими другими (к сожалению, большинство из этих систем имеют свои собственные расширения этого языка).

Как уже говорилось ранее, базы данных содержат объекты, которые называются таблицами. Именно в этих таблицах содержатся записи, в которых хранятся данные. Таблицы идентифицируются по именам (например, "Persons (персоны)", "Orders (заказы)", "Users (пользователи)").

Таблицы состоят из колонок и рядов с данными. Ряды содержат записи (например, одна запись для каждого человека). Колонки содержат данные (например, Category (категория), Name (имя), Age (воздраст).

Пример таблицы, имеющей имя "Users(пользователи)" приведен ниже.

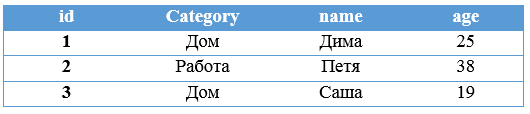


Рис.5. Таблица БД

LastName, FirstName, Address, и City - это колонки таблицы. Ряды содержат 3 записи с информацией о 3 персонах.

Рассмотрим примеры основных SQL команд, таких как CREATE TABLE, SELECT, INSERT, UPDATE, DELETE

1. CREATE TABLE

CREATE TABLE

`users` (

`category` ENUM('дом', 'работа', 'учеба') NOT NULL,

`id` MEDIUMINT NOT NULL AUTO\_INCREMENT,

`name` CHAR(30) NOT NULL,

`age` SMALLINT(6) NOT NULL,

PRIMARY KEY (`id`, `category`)

)

1. SELECT

SELECT \* FROM users

1. INSERT INTO

INSERT INTO

`users` (`category`, `name`, `age`)

VALUES

('дом', 'Оля', 26),

('дом', 'Настя', 20),

('работа', 'Артем', 26),

('учеба', 'Дима', 25),

('работа', 'Саша', 27),

('учеба', 'Миша', 25),

('работа', 'Лена', 35)

1. UPDATE

UPDATE users

SET name= 'Иван'

WHERE age=20

1. DELETE

DELETE FROM users

WHERE name = 'Иван'

## **Интерфейс JDBC**

*JDBC* (*Java Database Connectivity*) является не протоколом, а интерфейсом и основан на спецификациях *SAG CLI* (*SQL Access Group Call Level Interface* – интерфейс уровня вызова группы доступа *SQL*). *JDBC* – это стандартный прикладной интерфейс (*API*) языка *Java* для организации взаимодействия между приложением и СУБД. Обобщенная модель работы представлена на рис 6.

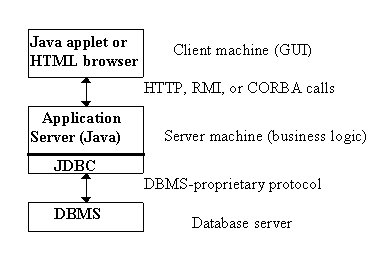


Рис.6.Принцип работы.

## **Пакеты, классы и интерфейсы Java для работы с БД**

Основные пакеты в Java для работы с БД это:

* java.sql.\* - основные классы для работы с данными

Обеспечивает API для доступа и обработки данных, сохраненных в источнике данных (обычно реляционная база данных) .

* sun.jdbc.odbc.\* - классы и интерфейсы моста JDBC – ODBC

## **Основные классы и интерфейсы**

* java.sql.Connection – соединение с драйвером БД;

Данный интерфейс определяет характеристики и состояние соединения с БД; кроме того, он предоставляет средства для контроля транзакций и уровня их изолированности;

* java.sql.DriverManager – загрузка и манипулирование драйверами БД;

Т.е. данный интерфейс обеспечивает загрузку драйверов и создание новых соединений (connection) с базой данных; это стержневой интерфейс JDBC, определяющий корректный выбор и инициализацию драйвера для данной СУБД в данных условиях;

* java.sql.Statement – объекты для исполнения SQL-предложений;

Т.е. данный класс используется для отправки SQL-операторов

* java.sql.ResultSet – объекты для обработки результатов Select-запросов;

Данный интерфейс предоставляет доступ к набору строк, полученному в результате выполнения данного SQL-выражения.

* java.sql.SQLException – прерывания при работе с БД;

Проще говоря обработчик исключений при проблемах с получением доступа к данным БД.

* sun.jdbc.odbc.JdbcOdbcDriver – драйвер моста JDBC – ODBC.

## **Схема работы с БД из программ на Java**

Ниже приведена последовательность действий для работы с БД приложений, написанных на языке Java.

1. Загрузить класс(ы), реализующие необходимые драйверы

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver")

1. Установить соединение с БД, используя загруженный драйвер

DriverManager.getConnection("jdbc:odbc:dsn")

1. Создать объект(ы) для исполнения SQL-команд

connection.createStatement();

1. Исполнять необходимые SQL-команды

stmt.executeUpdate("Delete From MyTable Where Id=1");

stmt.executeQuery("Select \* From MyTable");

1. Обрабатывать полученные таблицы

result.getString("fieldName");

1. Закрыть открытые соединения

connection.close();

## **Описание последовательности действий разработчика по организации взаимодействия приложений Java с базой данных MySQL**

В данной лабораторной работе будет использована СУБД MySQL и интегрированная среда разработки NetBeans.

Создадим приложение на языке программирования Java, реализующее двухзвенную архитектуру (приложение – база данных). Приложение позволяет добавлять информацию в базу данных. Структура таблицы базы данных:

Таблица Assortiment

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| Naimenovanie | Текстовый |
| Nazvanie | Текстовый |
| Cena | Денежный |
| Massa | Числовой |

1) Для начала нам потребуется СУБД MySQL, которую мы можем загрузить с официального сайта https://www.mysql.com/

2) После установки запустим приложение mysqld, которое располагается по пути ……\MySQL Server 5.5\bin\mysqld.exe.

3) Запустим NetBeans и с его помощью создадим базу данных для СУБД MySQL и таблицу с требующимися нам столбцами.

4) Переходим во вкладку службы, выбираем базы данных, далее MySQL. Далее нажимаем создать базу данных. (Рис 7)

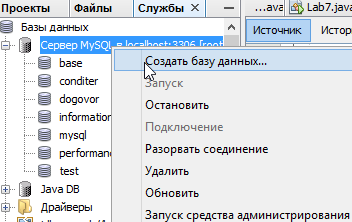


Рис.7.Вкладка службы.

5) Устанавливаем соединение с БД. (Рис 8)

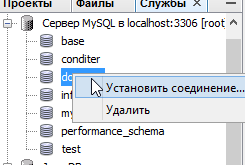


Рис.8.Установка соединения.

6) Создаем таблицу (Рис 9)

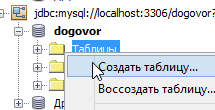


Рис.9.Вкладка службы.

7) Добавляем столбцы (Рис 10)

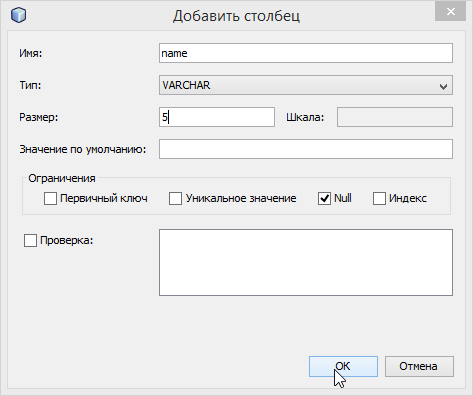


Рис.10.Добавление столбца

8) Прежде чем подключаться к серверу БД необходимо выяснить, к какому именно серверу вы собираетесь подключаться и скачать соответствующий драйвер. Драйвер представляет собой \*.jar файл. В нашем случае мы выбираем коннектор MySQL. (Рис 11)

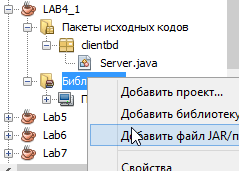


Рис.11.Добавление коннектора

Для лучшего представления работы запустим пример 4.1, предварительно создав таблицу *positions* со столбом *name*, заполнив его данными.

Пример 4.1 Организация взаимодействия приложения двухзвенной архитектуры.

В индивидуальном задании студента вместо swing необходимо использовать JavaFX

import java.awt.\*;

import java.net.\*;

import java.sql.\*;

import java.io.\*;

import javax.swing.\*;

import java.awt.event.\*;

import javax.swing.table.\*;

import javax.swing.event.\*;

public class DBClass extends JFrame{

JTable table1;JScrollPane sp;

static String nazv, cena, naim;

static int massa;

Statement sq;

ListSelectionModel lsm;

DefaultTableModel dtm;

String vibor\_naim;

Connection db;

String colheads[]={"Наименование", "Название", "Цена","Масса"};

static Object dataConditer[][];

JTextField tf1,tf2,tf3,tf4;

void showData(){

try{

String sq\_str="SELECT \* FROM Assortiment";

ResultSet rs= sq.executeQuery(sq\_str);

while(rs.next()){

nazv=rs.getString("Nazvanie");

cena=rs.getString("Cena");

naim=rs.getString("Naimenovanie");

massa=rs.getShort("Massa");

Object addingData[]= {naim,nazv,cena,massa};

dtm.addRow(addingData);

System.out.println(naim+nazv+massa+cena);

}

}

catch(Exception e){}

}

public DBClass(String Title){

super(Title);

setLayout(null);

dtm =new DefaultTableModel(dataConditer,colheads);

table1=new JTable(dtm);

JScrollPane sp=new JScrollPane(table1,ScrollPaneConstants.VERTICAL\_SCROLLBAR\_AS\_NEEDED,ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

sp.setBounds(50,5,350,250);

add(sp);

JButton btn1=new JButton("Добавить");

btn1.setBounds(100,370,100,30);

add(btn1);

btn1.addActionListener(new ActionListenerClass1());

tf1=new JTextField("0");

tf1.setBounds(55,280,80,30);

add(tf1);

tf2=new JTextField("0");

tf2.setBounds(140,280,80,30);

add(tf2);

tf3=new JTextField("0");

tf3.setBounds(225,280,80,30 );

add(tf3);

tf4=new JTextField("0");

tf4.setBounds(310,280,80,30 );

add(tf4);

lsm=table1.getSelectionModel();

lsm.addListSelectionListener(new SelectionListenerClass());

this.setSize(500,450);

this.setVisible(true);

try{

Class.forName("com.mysql.jdbc.Driver");//получение класса драйвера

db=DriverManager.getConnection("jdbc:mysql://localhost:3306/conditer", "root", "");

sq=db.createStatement();

String sq\_str="SELECT \* FROM Assortiment";

ResultSet rs= sq.executeQuery(sq\_str);

while(rs.next()){

nazv=rs.getString("Nazvanie");

cena=rs.getString("Cena");

naim=rs.getString("Naimenovanie");

massa=rs.getShort("Massa");

Object addingData[]= {naim,nazv,cena,massa};

dtm.addRow(addingData);

}

}

catch(Exception e){}

}

class SelectionListenerClass implements ListSelectionListener{

public void valueChanged(ListSelectionEvent lse){

Object obj=table1.getValueAt(table1.getSelectedRow(),table1.getSelectedColumn());

tf1.setText(table1.getValueAt(table1.getSelectedRow(),0).toString());

tf2.setText(table1.getValueAt(table1.getSelectedRow(),1).toString());

tf3.setText(table1.getValueAt(table1.getSelectedRow(),2).toString());

tf4.setText(table1.getValueAt(table1.getSelectedRow(),3).toString());

vibor\_naim=tf1.getText();

}

}

class ActionListenerClass1 implements ActionListener{//кнопка Добавить

public void actionPerformed(ActionEvent ae)

{

try{

naim=tf1.getText();

nazv=tf2.getText();

cena=tf3.getText();

massa=Integer.parseInt(tf4.getText());

String sq\_str\_insert="INSERT INTO Assortiment VALUES ('"+naim+"','"+nazv+"','"+

cena+"',"+massa+")";

int rs\_update2= sq.executeUpdate(sq\_str\_insert);

Object addingData[]={naim,nazv,massa,cena};

dtm.addRow(addingData);

}

catch(Exception e){}

}

}

public static void main(String args[]){

// объявление переменных

DBClass classObj=new DBClass("Frame");

}

}

## **Организация взаимодействия приложений трехзвенной архитектуры**

Следующий пример демонстрирует работу с БД в трехзвенной архитектуре (рис. 11). Вся информация хранится в таблице базы данных. Клиент взаимодействует с сервером по протоколу TCP. Клиент посылает запрос на сервер на добавление данных в таблицу. Приложение сервера подключается к базе, реализует SQL запрос на добавление. Обновленные данные таблицы посылаются сервером клиенту. Клиент видит результат добавления данных. Структура таблицы базы данных приведена ниже в таблице.

Таблица договор

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| Name | Varchar(10) |
| Type | Varchar(5) |
| Firma | Varchar(10) |
| Price | Integer |
| Date | Date |

Пример 4.2 Реализации трёхзвенной архитектуры.

**Клиент:**

import java.awt.Color;

import java.net.\*;

import java.sql.\*;

В индивидуальном задании студента вместо swing необходимо использовать JavaFX

import java.io.\*;

import javax.swing.\*;

import java.awt.event.\*;

import javax.swing.table.\*;

public class Client extends JFrame{

Socket sock;

InputStream is;

OutputStream os;

String message;

int a=0;

JTable table1;JScrollPane sp;

static String name, type, firma,price,date;

Statement sq;

DefaultTableModel DTM;

String vibor\_naim;

Connection db;

String colheads[]={"Наименование", "Тип", "Фирма","Цена","Дата"};

static Object dataConditer[][];

byte clientMessage[]=new byte[5000];

byte bytemessage[] = new byte[5000];

byte bytemessage2[] = new byte[100];

JTextField tf1,tf2,tf3,tf4,tf5;

public Client(String Title){

super(Title);

setLayout(null);

DTM=new DefaultTableModel(dataConditer,colheads);

table1=new JTable(DTM);

RowSorter<TableModel> sorter=new TableRowSorter<TableModel>(DTM);

table1.setRowSorter(sorter);

table1.setBackground(Color.getHSBColor(159, 216, 234));

JScrollPane sp=new JScrollPane(table1,ScrollPaneConstants.VERTICAL\_SCROLLBAR\_AS\_NEEDED,ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

sp.setBounds(50,5,350,250);

add(sp);

JButton btn1=new JButton("Добавить");

JButton btn2=new JButton("Отключить");

btn1.setBounds(84,320,125,30);

add(btn1);

btn1.addActionListener(new ActionListenerClass1());

btn2.setBounds(239,320,125,30);

btn1.setBackground(Color.getHSBColor(72,150,180));

btn2.setBackground(Color.getHSBColor(72,150,180));

add(btn2);

btn2.addActionListener(new ActionListenerClass2());

tf1=new JTextField("0");

tf1.setBounds(50,280,68,25);

add(tf1);

tf2=new JTextField("0");

tf2.setBounds(120,280,68,25);

add(tf2);

tf3=new JTextField("0");

tf3.setBounds(190,280,68,25 );

add(tf3);

tf4=new JTextField("0");

tf4.setBounds(260,280,68,25 );

add(tf4);

tf5=new JTextField("0");

tf5.setBounds(330,280,68,25 );

add(tf5);

this.setSize(500,450);

this.setVisible(true);

addWindowListener(new WindowClose());

try{

sock=new Socket(InetAddress.getByName("localhost"),1024);

is=sock.getInputStream();

os=sock.getOutputStream();

is.read(clientMessage);

String tempString=new String(clientMessage,0,clientMessage.length);

for (int i=0; i<tempString.length(); i++)

{

if (tempString.charAt(i) == '/')

a++;

}

String arrStr[] = tempString.split("/");

for (int r=0; r<=a;r=r+5)

{

Object addingData[]= {arrStr[r],arrStr[r+1],arrStr[r+2],arrStr[r+3],arrStr[r+4]};

DTM.addRow(addingData);

}

}

catch(Exception e){}

}

class ActionListenerClass1 implements ActionListener{

public void actionPerformed(ActionEvent ae)

{

try{

name=tf1.getText();

type=tf2.getText();

firma=tf3.getText();

price=tf4.getText();

date=tf5.getText();

Object addingData[]={name,type,firma,price,date};

DTM.addRow(addingData);

message = (name+ "/" + type +"/" +firma+"/"+ price + "/" +date+ "/");

bytemessage=message.getBytes();

os.write(bytemessage);

}

catch(Exception e){}

}

}

class ActionListenerClass2 implements ActionListener{

public void actionPerformed(ActionEvent ae)

{

try{

String h ="End";

bytemessage2=h.getBytes();

os.write(bytemessage2);

}

catch(Exception e){}

}

}

public static void main(String args[]){

Client classObj=new Client("Client");

}

class WindowClose extends WindowAdapter {

public void windowClosing(WindowEvent we) {

try{

String h ="End";

bytemessage2=h.getBytes();

os.write(bytemessage2);

}

catch(Exception e){}

System.exit(0);

}

}

}

**Сервер:**

import java.awt.Color;

import java.net.\*;

import java.sql.\*;

import java.io.\*;

import javax.swing.\*;

import java.awt.event.\*;

import javax.swing.table.\*;

public class Server extends JFrame{

ServerSocket sock;

InputStream is;

OutputStream os;

String st;

String stroka;

JTable table1;JScrollPane sp;

static String name, newname,type ,newtype,firma , newfirma ,price,newprice,date,newdate ;

Statement sq;

DefaultTableModel DTM;

String vibor\_naim;

Connection db;

String colheads[]={"Наименование", "Тип", "Фирма","Цена","Дата"};

static Object dataConditer[][];

byte bytemessage[] = new byte[10000];

public Server(String Title){

super(Title);

setLayout(null);

DTM=new DefaultTableModel(dataConditer,colheads);

table1=new JTable(DTM);

table1.setBackground(Color.getHSBColor(159, 216, 234));

JScrollPane sp=new JScrollPane(table1,ScrollPaneConstants.VERTICAL\_SCROLLBAR\_AS\_NEEDED,ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

sp.setBounds(50,5,350,250);

add(sp);

this.setSize(500,450);

this.setVisible(true);

addWindowListener(new WindowClose());

try{

Class.forName("com.mysql.jdbc.Driver");//получение класса драйвера

db=DriverManager.getConnection("jdbc:mysql://localhost:3306/dogovor", "root", "");

sq=db.createStatement();

StringBuffer x = new StringBuffer();

String sq\_str="SELECT \* FROM dogovor";

ResultSet rs= sq.executeQuery(sq\_str);

while(rs.next()){

name=rs.getString("Name");

type=rs.getString("Type");

firma=rs.getString("Firma");

price=rs.getString("Price");

date=rs.getString("Date");

Object addingData[]= {name,type,firma,price,date};

DTM.addRow(addingData);

st = (name+ "/" + type +"/" +firma+"/"+ price +"/" +date+"/");

x.append(st);

}

stroka=x.toString();

sock=new ServerSocket(1024);

while(true)

{

Socket client=sock.accept();

is=client.getInputStream();

os=client.getOutputStream();

bytemessage=stroka.getBytes();

os.write(bytemessage);

boolean flag = true;

while(flag==true)

{

byte readmessage[] = new byte[10000];

is.read(readmessage);

String tempString=new String(readmessage,0,readmessage.length);

if(tempString.trim().compareTo("End")==0) {

flag = false;

}

else

{

stroka= stroka+tempString;

String arrStr[] = tempString.split("/");

newname=arrStr[0];

newtype=arrStr[1];

newfirma=arrStr[2];

newprice=arrStr[3];

newdate=arrStr[4];

Object addingData[]={newname,newtype,newfirma,newprice,newdate};

DTM.addRow(addingData);

String sq\_str\_insert="INSERT INTO dogovor VALUES ('"+newname+"','"+newtype+"','"+

newfirma+"','"+newprice+"','"+newdate+"')";

int rs\_update2= sq.executeUpdate(sq\_str\_insert);

}}

is.close();

os.close();

client.close();

}

}

catch(Exception e){}

}

public static void main(String args[]){

Server classObj=new Server("Server");

}

class WindowClose extends WindowAdapter {

public void windowClosing(WindowEvent we) {

System.exit(0);

}

}

}

## **Варианты индивидуальных заданий**

Задание: разработать приложения, взаимодействующие с базой данных, по схеме, показанной на рисунке 12.

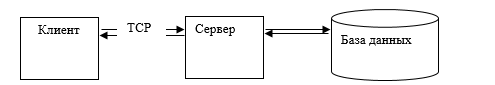


Рис.12 Схема

**Требования к базе данных:**

СУБД MySQL, состоит из одной таблицы.

Таблица содержит 6 полей. Название и содержание таблицы для каждого варианта свои (определены ниже).

**Требования к клиенту:**

GUI интерфейс, реализованный на JavaFX.

Взаимодействует с сервером по протоколу TCP. Может добавлять, удалять, редактировать информацию в базе данных (кнопки «Добавить», «Удалить», «Редактировать»).

**Требования к серверу:**

интерфейс любой (может быть GUI или в виде консоли).

Перенаправляет запросы на добавление от клиента к базе данных и возвращает назад результат.

Названия таблицы базы данных по вариантам:

1. Автомобили.
2. Книги.
3. Страны.
4. Студенты.
5. Товары.
6. Адресная книга.
7. Договора.
8. Военная техника.
9. Животные.
10. Растения.
11. Профессии.
12. Мебель.
13. Магазины.
14. Одежда.
15. Факультет

## **Контрольные вопросы**

1. Понятие базы данных (БД). Требования, предъявляемые к БД.
2. Основные типы структур (моделей) БД.
3. Понятия сущность, атрибут, отношение, типы отношений между сущностями.
4. Таблица как объект БД: поля, записи, ключ. Описание структуры таблицы.
5. Операторы SQL: СREATE TABLE, SELECT, INSERT, UPDATE, DELETE.
6. Понятия драйвера, соединения и запроса к базам данных.

**Источники:**

Web-ресурсы:

1. <https://netbeans.org/kb/docs/ide/mysql_ru.html>
2. <https://netbeans.org/kb/docs/php/wish-list-lesson1_ru.html>
3. <http://www.mysql.ru/docs/mysql-man-4.0-ru/tutorial.html#connecting-disconnecting>
4. http://www.javaprobooks.ru/java-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5/%D1%87%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-jdbc.html
5. http://www.lib.csu.ru/dl/bases/prg/kompress/articles/2000\_10\_DBMS6/index.htm

Печатные издания:

1. Книга «Изучаем Java» Берт Бейтс, «ЭКСМО»,2012г.
2. Книга «Язык запросов SQL. Учебный курс» Филипп Андон, Валерий Резниченко, «Питер, BHV - Киев», 2006 г.