Лабораторная работа № 1

Введение

В данной работе используется двухуровневая архитектура модели доступа к базе данных - приложение ***Java*** взаимодействует напрямую с источником данных, в отличие от трехуровневой (*Java Tutorials)*, с использованием бизнес-логики на клиентском компьютере, Рисунок 1.

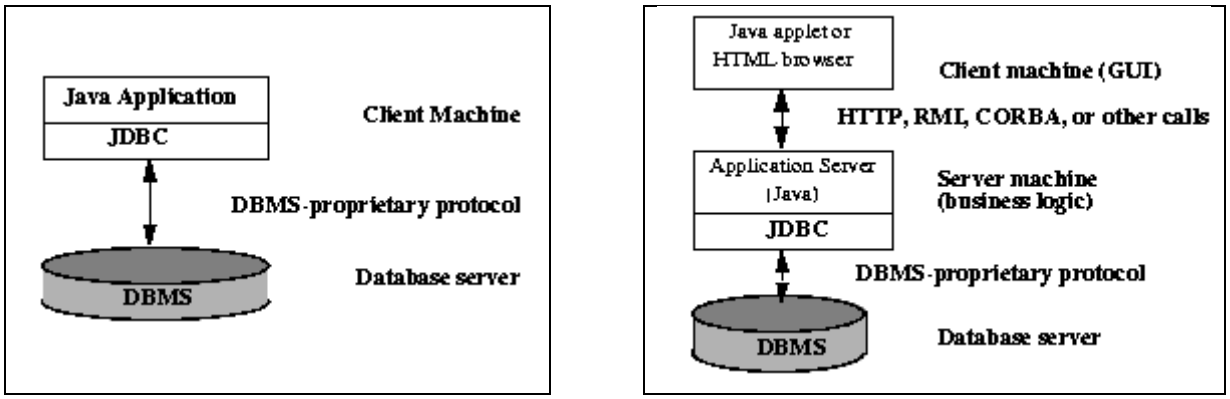

Рисунок 1

Для разработки приложения двухуровневой архитектуры с использованием базы данных, например, для связи с сервером ***MySQL***, необходимо установить на компьютер следующие компоненты:

- платформа Java Standard Edition (Java SE), включающая Java Development Kit (JDK) и Java Runtime Environment (JRE),
- среда IDE NetBeans,
- система управления базами данных (СУБД) ***MySQL,***
- драйвер ***JDBC*** от поставщика базы данных ***MySQL*** (***MySQL Connector / J*** - является драйвером ***JDBC***, обеспечивающим связь для клиентских приложений, разработанных на языке программирования ***Java*** с СУБД ***MySQL***). Приложение ***JDBC*** подключается к целевому источнику данных, используя один из двух классов: ***DriverManager***, либо ***DataSource***. В данной работе используется класс ***DriverManager***.

Соединение из приложения Java с базой данных

Задание № 1

1. Скачайте с официальных сайтов и установите ***Java SE***, ***IDE NetBeans***, и ***MySQL8*** (https://dev.mysql.com) **.**
2. Откройте MySQL Workbench (запустится локальный SQL Server) и создайте новую базу данных ***mydb***,
3. Откройте среду ***IDE NetBeans,*** выполните команду меню ***Tools, Libraries***. В окне ***Library Manager*** нажмите ***New Library,*** создайте новую библиотеку и добавьте путь к драйверу, Рисунок 2.
4. Создайте новый проект, выберите Категорию ***Java with Ant***, Проекты –***Java Application.***
5. На втором шаге укажите имя проекта ***MyDB1***, расположение проекта, например, ***d:\DataBase***, имя пакета ***mydb*** и главного класса ***TestMydb0***.
6. Чтобы указать путь к драйверу ***JDBC*** в среде ***IDE NetBeans*** в контекстном меню папки ***Libraries*** выберите ***Add Library,*** затем укажите библиотеку**,** Рисунок 3.
7. Наберите код класса ***TestMydb0***, согласно Рисунку 4.
8. Добавьте в файл ***TestMydb0*** новый класс ***Mydb0.java*** и наберите в нем исходный код, Рисунок 5.
9. Создайте новый пакет ***myConnect*** с классом ***Connect*** (через контекстное меню папки ***src***).
10. Наберите в файле ***Connect.java*** исходный код, Рисунок 6.

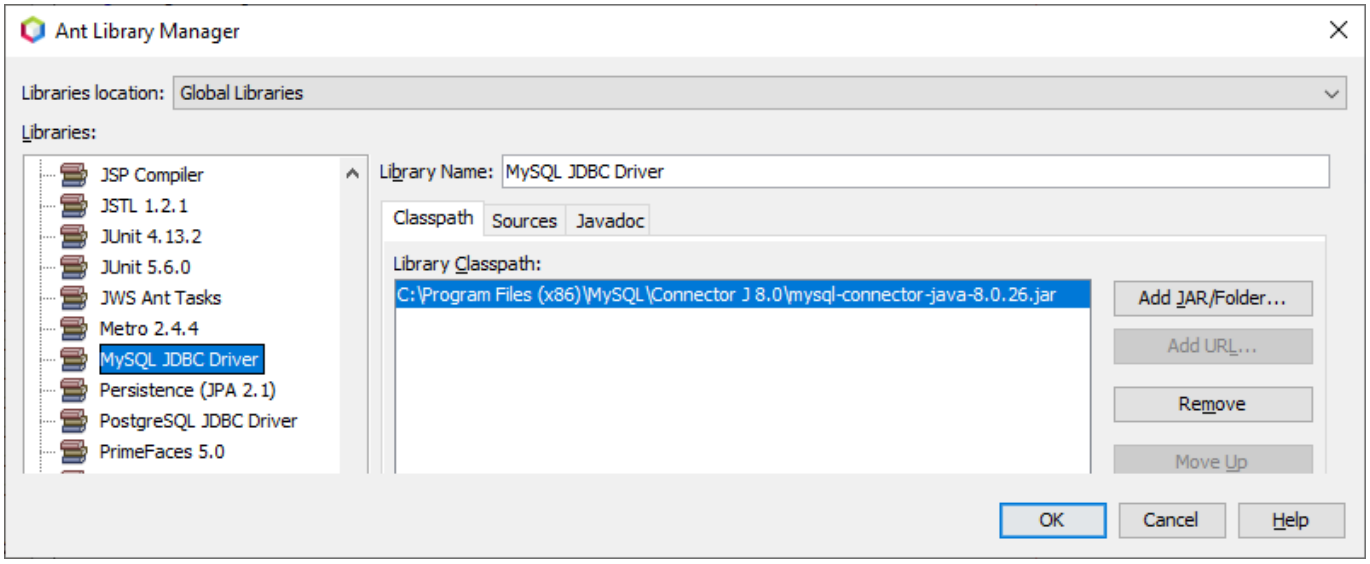

Рисунок 2

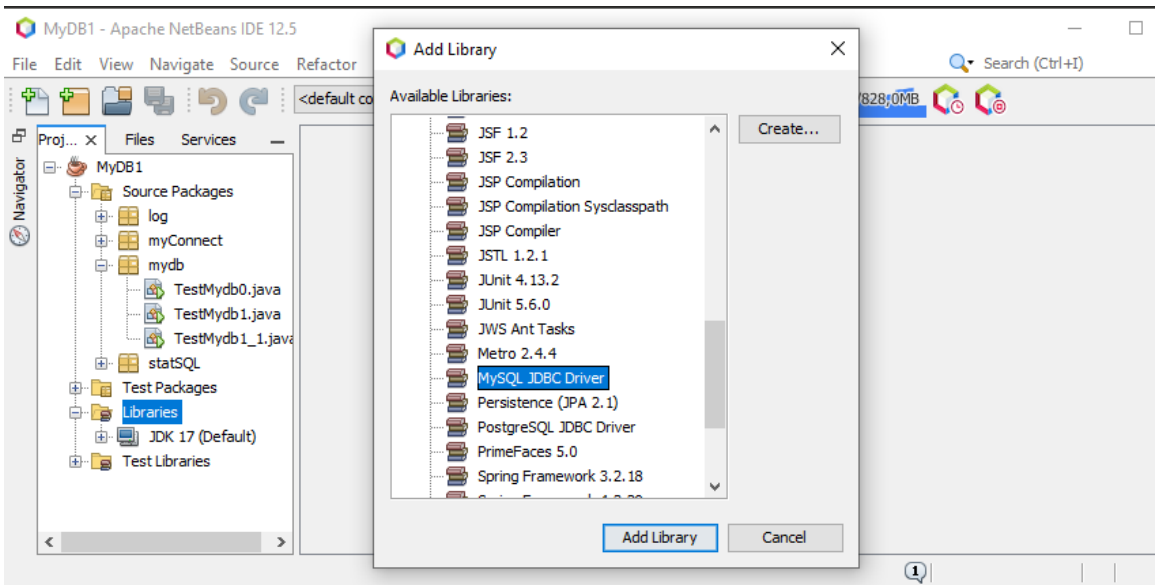

Рисунок 3

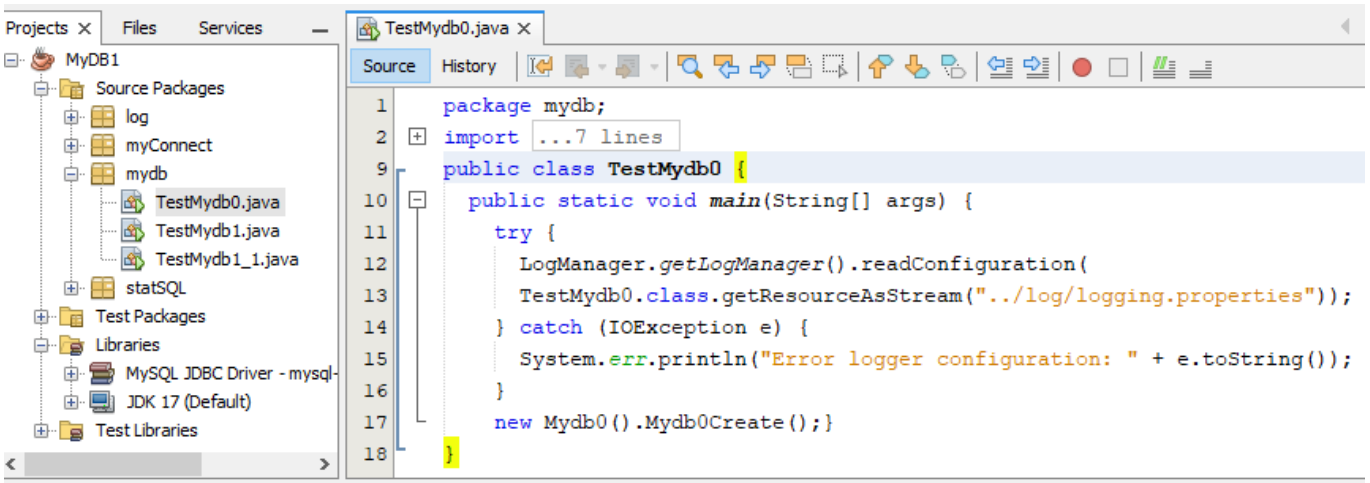

Рисунок 4

TestMydb0.java

Source | History

```
19  class Mydb0 {
    private static Logger log = Logger.getLogger(Mydb0.class.getName());
21  public void Mydb0Create() {
22      try {
        Connection conn = new Connect().getConnection();
24      Statement stat = conn.createStatement();
25      stat.execute("CREATE TABLE MyTable1(Message VarChar(50))");
26      stat.execute("INSERT INTO MyTable1 VALUES ('It\"s MyTable1!')");
27
28      ResultSet result = stat.executeQuery("SELECT * FROM MyTable1");
29      result.next();
30      JOptionPane.showMessageDialog(null, result.getString(1));
31      JOptionPane.showMessageDialog(null, result.getString("Message"));
32      result.close();
33      stat.execute("DROP TABLE MyTable1");
34      JOptionPane.showMessageDialog(null, "DROP TABLE MyTable1");
35      stat.close();
36      conn.close();
37    }
38    catch (SQLException ex) {
39      while (ex != null) {
40      log.log(Level.SEVERE, "Exception: ", ex);
41        ex = ex.getNextException();
42      }
43    }
44    catch (IOException ex) {
45      log.log(Level.SEVERE, "Exception: ", ex);}
46  }
47  }
```

Рисунок 5

logging.properties | Connect.java | TestMydb1.java | TestMydb1_1.java

```
package myConnect;
import ...3 строк
public class Connect {
 public Connection getConnection() throws  IOException ,SQLException {
    Properties props = new Properties();
    FileInputStream in = new FileInputStream("prop/mydb.properties");
    props.load(in);
    in.close();
    String url = props.getProperty("jdbc.url");
    String username = props.getProperty("jdbc.username");
    String password = props.getProperty("jdbc.password");
    return  DriverManager.getConnection(url, username, password);
} }
```

Рисунок 6

Задание № 2

1. Создайте новую папку ***prop*** и в ней новый файл свойств ***mydb.properties*** (На вкладке ***Files*** через контекстное меню папок ***MyDB1*** и ***prop***).
2. Наберите в файле ***mydb.properties*** названия свойств и их значения, Рисунок 7.

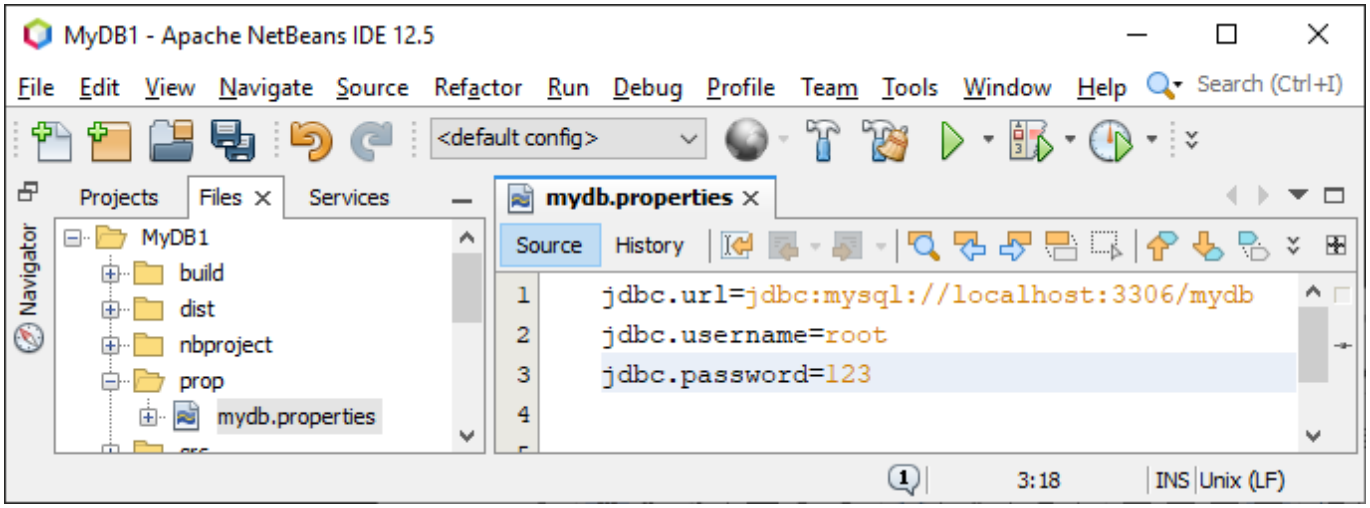

Рисунок 7

Логирование используется для записи различных сообщений об ошибках в специальный файл, чтобы не использовать для этого консоль или окно вывода в IDE.

Логгер это статическое поле класса инициализируемое при загрузке класса, использующее метод log для записи сообщений об ошибках. Класс LogManager имеет метод для указания файла конфигурирования логов.

Задание № 3

1. Создайте новую папку ***log*** и в ней новый файл свойств ***logging.properties*** (через контекстное меню папок ***src*** и ***log***.
2. Наберите в файле ***logging.properties*** названия свойств и их значения, Рисунок 8.

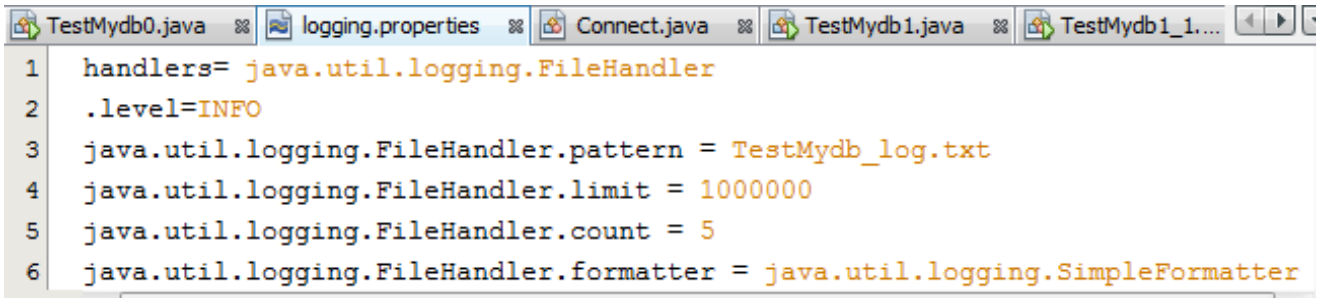

Рисунок 8

3. Запустите файл ***Mydb0.java*** на выполнение (контекстное меню, ***Выполнить файл***) и протестируйте его. Если в процессе выполнения вы не увидите окон диалога с сообщениями, например, It\"s MyTable1!, DROP TABLE MyTable1, то в папке ***MyDB1*** откройте файл вида ***TestMydb_log.txt.0***, прочитайте сообщения об ошибках и выполните отладку проекта.

Лабораторная работа № 1 (продолжение)

Работа с записями таблиц базы данных из приложения Java

1. метод ***execute*** объекта ***Statement*** возвращает ***true*** для запроса ***SELECT***, либо ***false*** для запросов ***UPDATE***, ***INSERT*** или ***DELETE***, например,
   ```
   if (stmt.execute("SELECT * FROM myTable")) { rs = stmt.getResultSet(); }
   ```
2. метод ***executeQuery*** возвращает объект ***ResultSet***, например,
   ```
   rs = stmt.executeQuery("SELECT * FROM myTable");
   ```
3. метод ***executeUpdate*** возвращает количество строк, затронутых оператором ***SQL***. Используйте этот метод, если вы используете ***INSERT***, ***DELETE*** или ***UPDATE***.

Получить ***ResultSet*** объекты, возвращенные из запроса, несколько раз можно вызывая ***Statement.getResultSet***. Вы получаете доступ к данным в объекте ***ResultSet*** через курсор. Этот курсор не является курсором базы данных, это указатель на одну строку данных в объекте ***ResultSet***. Первоначально курсор располагается перед первой строкой. Вы вызываете различные методы, определенные в ***ResultSet*** объекте, для перемещения курсора. Например, метод ***ResultSet.next*** вызывается для перемещения курсора вперед на одну строку.

Когда вы закончите использовать а ***Statement***, вызовите метод, ***Statement.close*** чтобы освободить ресурсы, которые он использует. Когда вы вызываете этот метод объекты ***ResultSet*** тоже закрываются.

Если запрос ***UPDATE***, ***INSERT*** или ***DELETE***, вы можете подсчитывать измененные строки путем вызова метода ***getUpdateCount()*** объекта ***Statement.***

В ***JDBC 4.1***, которая доступна в ***Java SE*** версии 7 и более поздних версиях можно использовать конструкцию ***try-with-resources*** для автоматического освобождения ресурсов ***Connection***, ***Statement*** и ***ResultSet*** объектов, независимо от того, было ли выброшено исключение, например,

```
try (Statement stmt = con.createStatement ()) {...}
```

Задание № 4

1. В проекте ***MyDB1*** в пакете ***mydb*** создайте главный класс ***TestMydb1***. Наберите код класса, согласно Рисунку 9.
2. Добавьте в файл ***TestMydb1*** новый класс ***Mydb1.java*** (через контекстное меню папки ***mydb***) и наберите в нем исходный код, Рисунок 10.

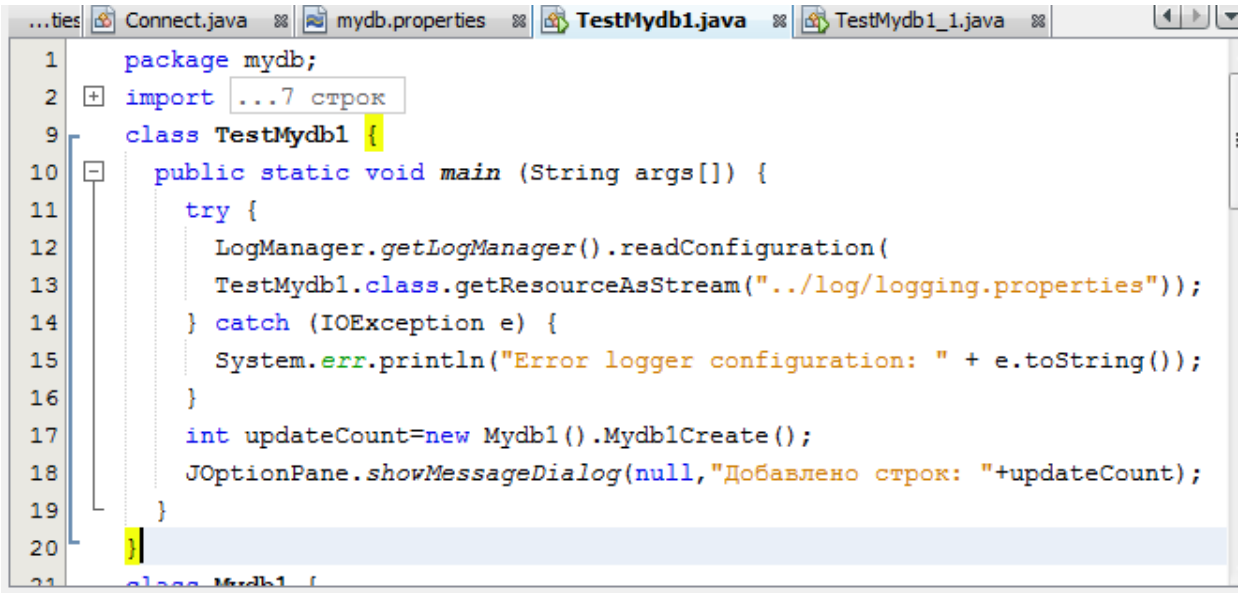

Рисунок 9

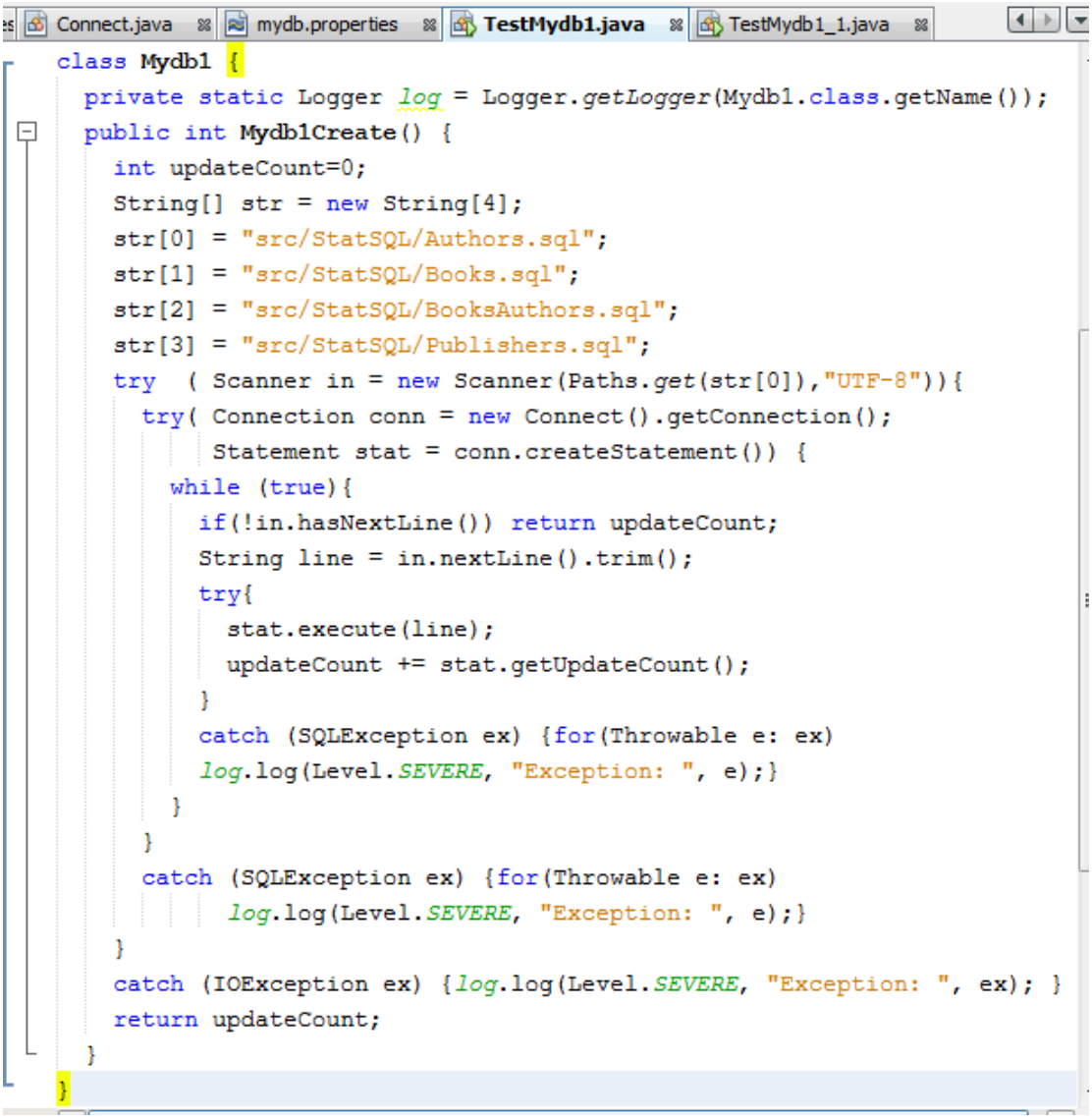

Рисунок 10

3. Создайте новую папку ***statSQL*** и в ней новые файлы: Authors.sql, Books.sql , BooksAuthors.sql, Publishers.sql (через контекстное меню папок ***src*** и ***statSQL***), Рисунок 11.

4. Наберите в файлах запросы, создания таблиц и вставки данных в таблицы, Рисунок 11. Примеры исходного кода, приведенные в этой работе, доступны по адресу: http: //horstmann.com/corejava [1].

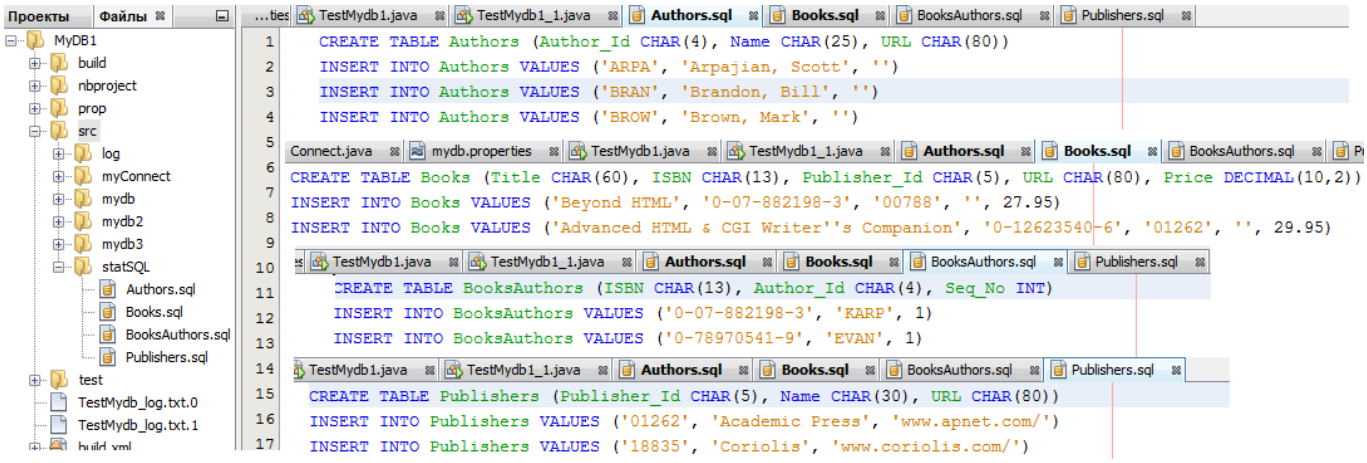

Рисунок 11

5. Запустите файл ***Mydb1.java*** на выполнение (контекстное меню, ***Выполнить файл***) и протестируйте его. Если в процессе выполнения вы не увидите окно диалога с сообщением, «Добавлено строк XX», то в папке ***MyDB1*** откройте файл вида ***TestMydb_log.txt.0***, прочитайте сообщения об ошибках и выполните отладку проекта.
6. Изменяя путь к файлу с ***SQL*** запросами Scanner(Paths.get(str[X]) запустите файл ***Mydb1.java*** на выполнение для всех файлов. У вас должно получиться в базе данных ***mydb*** четыре таблицы с данными.

Задание № 5

1. В проекте ***MyDB1*** в пакете ***mydb*** создайте главный класс ***TestMydb1_1***. Наберите код класса, согласно Рисунку 12.
2. Добавьте в файл ***TestMydb1_1*** новый класс ***Mydb1_1.java*** (через контекстное меню папки ***mydb***) и наберите в нем исходный код, Рисунки 12 и 13.
3. Запустите файл ***Mydb1_1.java*** на выполнение (контекстное меню, ***Выполнить файл***) и протестируйте его. Если в результате выполнения файла вы не увидите окно таблицей, как показано, например, на Рисунке 14, то в папке ***MyDB1*** откройте файл вида ***TestMydb_log.txt.0***, прочитайте сообщения об ошибках и выполните отладку проекта.
4. Изменяя строку с ***SQL*** запросами line = str[X] запустите файл ***Mydb1_1.java*** на выполнение для всех запросов. У вас должны получаться соответствующие запросам таблицы с данными, Рисунок 15.
5. Создайте модель из базы данных mydb, Рисунок 14, и установите связи между таблицами, Рисунок 16.

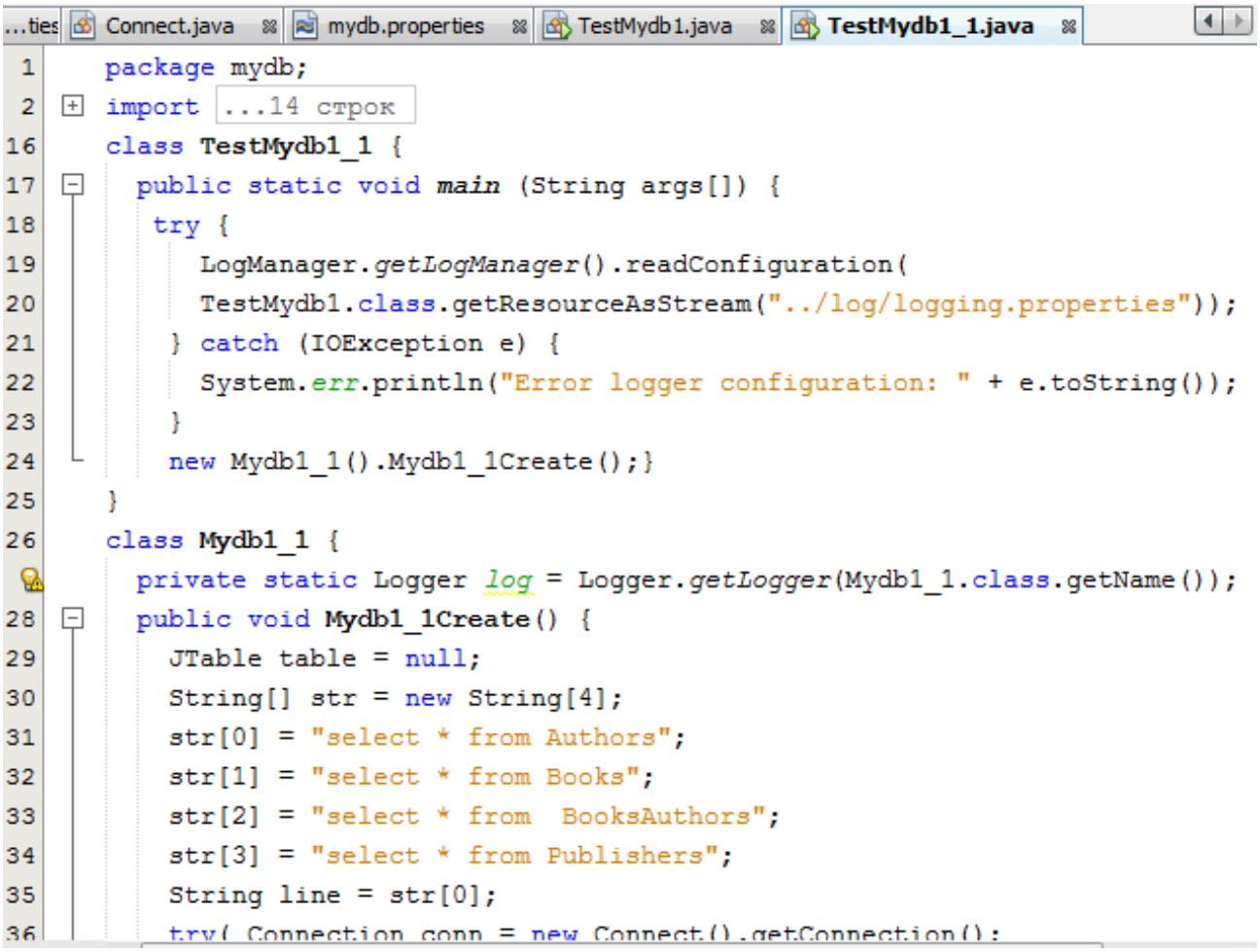

Рисунок 12

```
...ava  logging.properties  Connect.java  mydb.properties  TestMydb1.java  TestMydb1_1.java
36        try( Connection conn = new Connect().getConnection();
37             Statement stat = conn.createStatement()) {
38            try{
39              boolean isResult = stat.execute(line);
40              if (isResult){
41                try (ResultSet rs = stat.getResultSet()){
42                table = new JTable(buildTableModel(rs));}
43              }
44            }
45            catch (SQLException ex) {for(Throwable e: ex)
46              log.log(Level.SEVERE, "Exception: ", e);}
47          }
48          catch (SQLException ex) {for(Throwable e: ex)
49             log.log(Level.SEVERE, "Exception: ", e);}
50           catch (IOException ex) {log.log(Level.SEVERE, "Exception: ", ex); }
51          JFrame frame=new FrameTable(table, line);
52          frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
53           frame.setVisible(true);
54      }
55      public static DefaultTableModel buildTableModel(ResultSet rs)throws SQLException{
56        ResultSetMetaData metaData = rs.getMetaData();
57        Vector<String> columnNames = new Vector<String>();
58        int columnCount = metaData.getColumnCount();
59        for (int column = 1; column <= columnCount; column++) {
60          columnNames.add(metaData.getColumnName(column));
61        }
62        Vector<Vector<Object>> data = new Vector<Vector<Object>>();
63        while (rs.next()) {
64          Vector<Object> vector = new Vector<Object>();
65          for (int columnIndex = 1; columnIndex <= columnCount; columnIndex++) {
66              vector.add(rs.getObject(columnIndex));
67          }
68          data.add(vector);
69        }
70        return new DefaultTableModel(data, columnNames);
71      }
72      class FrameTable extends JFrame{
73        public FrameTable(JTable tbl,String line){
74          setTitle(line);
75          setSize(800, 600);
76          getContentPane().add(new JScrollPane(tbl),BorderLayout.CENTER);
77      } }
78    }
```

Рисунок 13

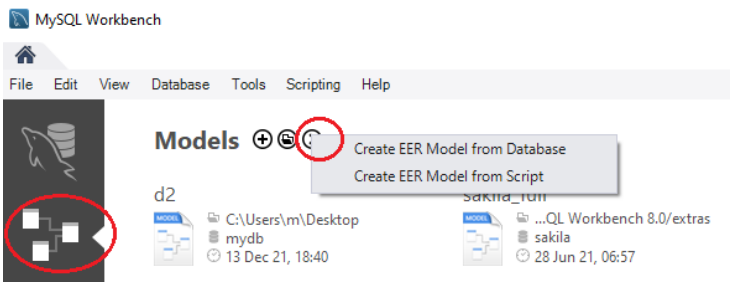

Рисунок 14

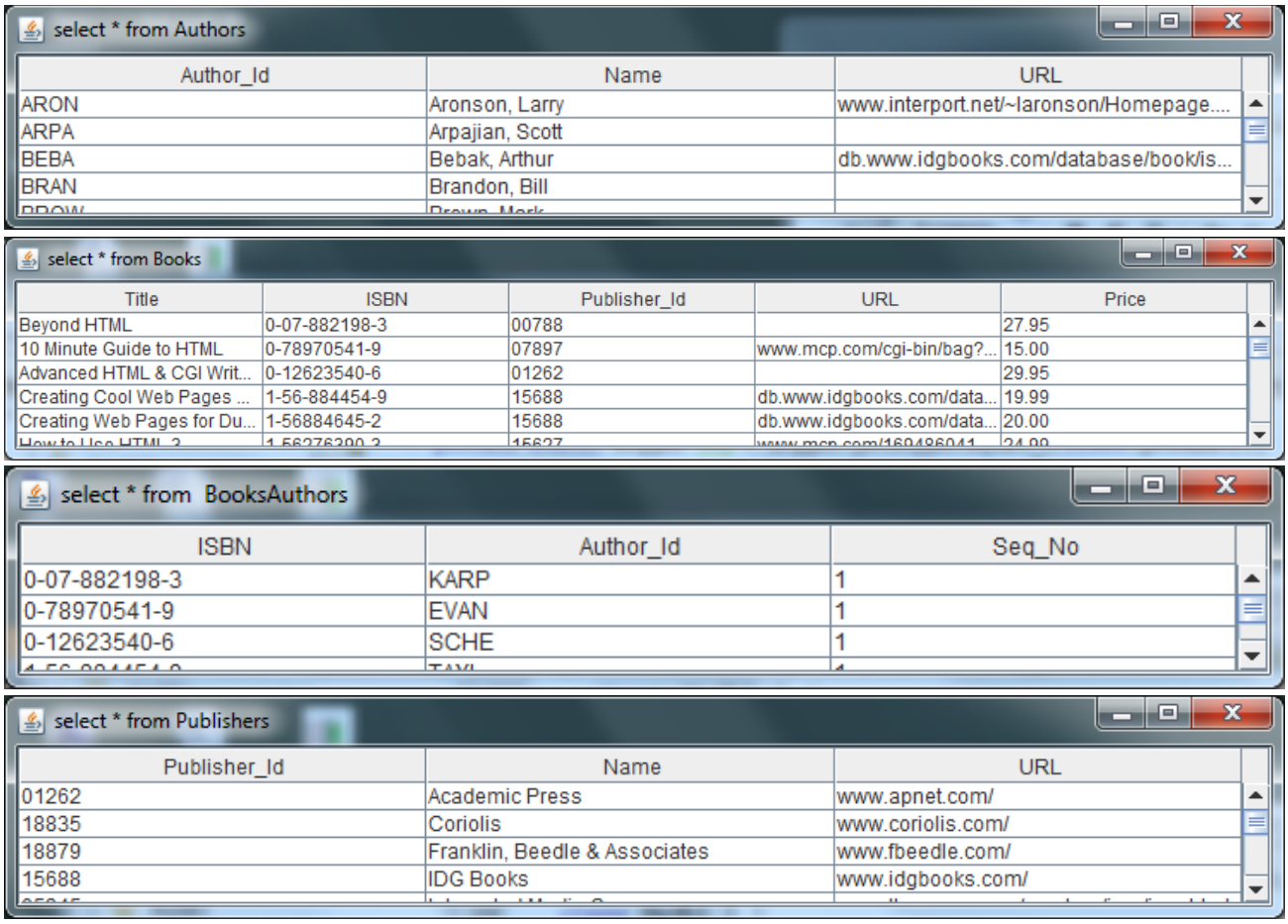

Рисунок 15

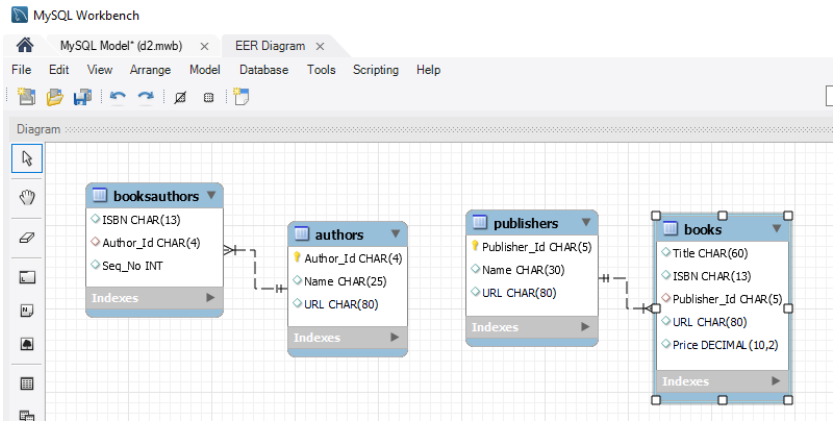

Рисунок 16

Список литературы

1. Хорстманн, Кей С. X82 Java. Библиотека профессионала, том 2. Расширенные средства программирования, 10-е изд. : Пер. с англ. — СпБ. : ООО "Альфа-книга", 2017. — 976 с.
2. https://www.oracle.com/technetwork/java/javase/documentation/index.html
3. https://docs.oracle.com/netbeans/nb82/netbeans/NBDAG/working_nbeans.htm
4. http://qaru.site/questions/180554/most-simple-code-to-populate-jtable-from-resultset
5. https://habr.com/ru/search/?q=логирование#h