Государственное образовательное учреждение высшего профессионального образования
ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ
Кафедра «Информационные и вычислительные системы»

Курсовой проект по дисциплине «Базы данных»
Пояснительная записка

Разработал:
студент гр. ПВТ-711
Круглов В.А.

Руководитель:
Тырва А.В.

Санкт-Петербург
2010

Текст задания

Оглавление

1. Описание предметной области

В нашем современно мире, мире новых технологий, до сих пор остаются изобретенные уже давно вещи, которые не теряют своей популярности. Одной из такой вещей является книга, будь то бумажное издание или же её электронная версия. Книги до сих пор остаются основным источником информации.

Многие предприятия занимающиеся книгами, зачастую обладают большими базами таковых, и порой найти нужную среди всех не так уж легко. Однако благодаря введению группировки книг по смысловому содержанию поиск оной становится намного проще. Для библиотек это каталоги, для магазинов это разделы. В итоге поиск и учет книг сводится к выбору раздела, в котором содержатся нужные книги. А благодаря ведению правильной иерархии это может стать еще более простой задачей.

Выбор данной предметной области обусловлен тем, что предприятиям, будь то библиотека или магазин, обладающим большими базами книг, важно учитывать наличие у себя тех или иных книг, а также оперативно организовывать их поиск.

В эту предметную область входят такие сущности как: Книга, Автор, Каталог, Свойство книги, а также связи между книгами и каталогами.

2. Выбор и описание БД

2.1. СУБД.

В курсовом проекте в качестве системы управления базой данных был выбран продукт компании Oracle — Oracle Database 10*g* XE.

Oracle Database 10*g* — первая в мире база данных, разработанная специально для работы в сетях распределенных вычислений. Oracle Database 10*g* предназначена для эффективного развертывания на базе различных типов оборудования, от небольших серверов до Oracle Enterprise Grid мощных многопроцессорных серверных систем, от отдельных кластеров до корпоративных распределенных вычислительных систем.

Oracle Database 10*g* позволяет пользователям виртуализировать использование аппаратного обеспечения — серверов и систем хранения данных. Oracle Database 10*g* обладает технологиями, которые позволяют администраторам надежно хранить и быстро распределять и извлекать данные для пользователей и приложений, работающих в сетях.

Oracle Database 10*g* предоставляет возможность автоматической настройки и управления, которая делает ее использование простым и экономически выгодным. Ее уникальные возможности осуществлять управление всеми данными предприятия — от обычных операций с бизнес-информацией до динамического многомерного анализа данных (OLAP), операций с документами формата XML, управления распределенной/локальной информацией — делает ее идеальным выбором для выполнения приложений, обеспечивающих обработку оперативных транзакций, интеллектуальный анализ информации, хранение данных и управление информационным наполнением.

СУБД Oracle Database 10*g* поставляется в четырех различных редакциях, ориентированных на различные сценарии разработки и развертывания приложений:

- ✓ Oracle Database 10*g* Standard Edition One
- ✓ Oracle Database 10*g* Standard Edition (SE)
- ✓ Oracle Database 10*g* Enterprise Edition (EE)
- ✓ Oracle Database 10*g* eXpress Edition (XE)

Кратко о редакции базы данных использованной в курсовом проекте.

Oracle Database XE — это СУБД начального уровня, основанная на базе кода Oracle Database 10*g* Release 2. Предоставляет возможность разработчикам ПО, администраторам баз данных и всем желающим получить бесплатную базовую версию СУБД, позволяющую начать разработку и развертывание собственных приложений. Кроме того, эта версия предлагается бесплатно независимым разработчикам ПО и поставщикам оборудования для свободной дистрибуции или встраивания в собственные приложения.

Oracle Database XE — это отличная СУБД начального уровня для:

- разработчиков, работающих с PHP, Java, .NET и Open Source приложениями (альтернатива MySQL);
- администраторов БД, которым нужна бесплатная СУБД для обучения и инсталляций;
- независимых производителей ПО и аппаратных средств, которым нужна бесплатная СУБД для свободного распространения;
- образовательных учреждений и студентов, которым необходима бесплатная СУБД для обучения.

Oracle Database XE имеет следующие преимущества:

- абсолютно свободная СУБД;
- можно тестировать, разрабатывать и распространять с нулевыми инвестициями в ПО и без риска;
- легко мигрировать на промышленные редакции;
- не требуется переписывание приложений при переходе на промышленные редакции;

В данном курсовом проекте создается база данных, предоставляющая информацию о книгах находящихся на предприятии, будь то склад, магазин или библиотека. Администратор данной базы сможет просматривать, а также изменять и добавлять, информацию о книгах, их авторах и каталогах, в которых они состоят. Для обычного пользователя, например продавец в магазине или библиотекарь, некоторые функции по использованию базы данных ограничены, дабы не повредить целостность. Для гостя же также определены права, но на минимальном уровне — просмотр и поиск.

Хранимой в базе информацией являются: данные об авторах книг, такие как имя, фамилия и дата рождения; данные о книгах, такие как название, автор, издательство, год издания, также есть возможность создания новых свойств для книг; данные о каталогах, причем каталоги имеют жесткую иерархичную структуру с отсутствием множественного наследования.

2.2. Сервер приложений.

В курсовом проекте в качестве сервера приложений был выбран продукт компании Sun Microsystems — GlassFish v3.

GlassFish v3 — первый сертифицированный сервер приложений, поддерживающий стандарт Java EE 6, включающий в себя:

- enterprise-технологии: EJB 3.1, JPA 2.0, JDBC 4.0, CORBA 3.0;
- Web-технологии: Servlet 3.0, JSP 2.2, JSTL 1.2, EL 2.2, JSF 2.0 (Facelets), RESTful web services;

2.3. Среда разработки.

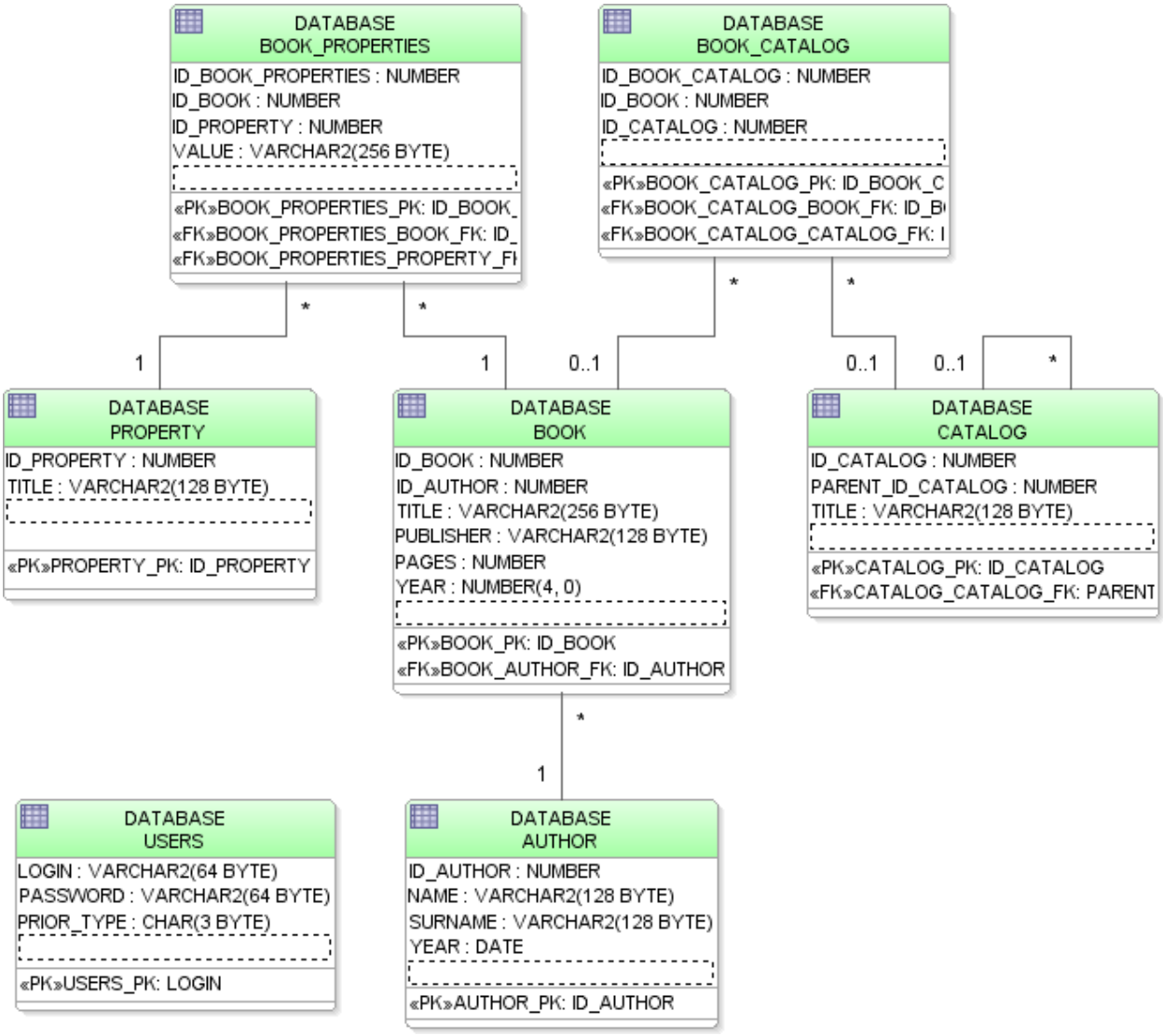
В курсовом проекте в качестве среды разработки был выбран продукт компании NetBeans Community — NetBeans 6.9.1.

NetBeans IDE — свободная интегрированная среда разработки приложений (IDE) на языках программирования Java, JavaFX, Ruby, Python, PHP, JavaScript, C++, Ада и ряде других. Данная среда поддерживает разработку Java Web-приложений, создание JSP, HTML страниц, XML документов, работу с базами данных.

Проект NetBeans IDE поддерживается и спонсируется компанией Oracle, однако разработка NetBeans ведется независимо сообществом разработчиков-энтузиастов (NetBeans Community) и компанией NetBeans Org.

3. Описание и схема БД

3.1. Схема БД



3.2. Описание схемы БД

Таблица AUTHOR — содержит информацию об авторах книг.

Атрибуты таблицы:

- ✓ ID_AUTHOR — идентификационный номер автора
- ✓ NAME — имя автора
- ✓ SURNAME — фамилия автора
- ✓ YEAR — дата рождения автора

Ограничения таблицы:

- AUTHOR_PK: ID_AUTHOR — первичный ключ таблицы

Таблица BOOK — содержит информацию о книгах.

Атрибуты таблицы:

- ✓ ID_BOOK — идентификационный номер книги
- ✓ ID_AUTHOR — идентификационный номер автора книги
- ✓ TITLE — название книги
- ✓ PUBLISHER — название издательства книги
- ✓ PAGES — кол-во страниц книги
- ✓ YEAR — год издания книги

Ограничения таблицы:

- BOOK_PK: ID_BOOK — первичный ключ таблицы
- BOOK_AUTHOR_FK: ID_AUTHOR — внешний ключ на Author

Таблица CATALOG — содержит информацию о каталогах.

Атрибуты таблицы:

- ✓ ID_CATALOG — идентификационный номер каталога
- ✓ PARENT_ID_PROPERTY — идентификационный номер родительского каталога
- ✓ TITLE — название каталога

Ограничения таблицы:

- CATALOG_PK: ID_CATALOG — первичный ключ таблицы
- CATALOG_CATALOG_FK: PARENT_ID_CATALOG — внешний ключ на Catalog

Таблица PROPERTY — содержит информацию о свойствах.

Атрибуты таблицы:

- ✓ ID_PROPERTY — идентификационный номер свойства
- ✓ TITLE — название свойства

Ограничения таблицы:

- PROPERTY_PK: ID_PROPERTY — первичный ключ таблицы

Таблица BOOK_CATALOG — содержит информацию о наличии книги в каталоге.

Атрибуты таблицы:

- ✓ ID_BOOK_CATALOG — идентификационный номер записи(связи)
- ✓ ID_BOOK — идентификационный номер книги
- ✓ ID_CATALOG — идентификационный номер каталога

Ограничения таблицы:

- BOOK_CATALOG_PK — первичный ключ таблицы
- BOOK_CATALOG_BOOK_FK: ID_BOOK — внешний ключ на Book
- BOOK_CATALOG_CATALOG_FK: ID_CATALOG — внешний ключ на Catalog

Таблица BOOK_PROPERTIES — содержит информацию о свойствах книг.

Атрибуты таблицы:

- ✓ ID_BOOK_PROPERTIES — идентификационный номер записи(связи)
- ✓ ID_BOOK — идентификационный номер книги
- ✓ ID_PROPERTY — идентификационный номер свойства
- ✓ VALUE — значение свойства

Ограничения таблицы:

- BOOK_PROPERTIES_PK — первичный ключ таблицы
- BOOK_PROPERTIES_BOOK_FK: ID_BOOK — внешний ключ на Book
- BOOK_PROPERTIES_PROPERTY_FK: ID_PROPERTY — внешний ключ на Property

Таблица USERS — содержит информацию о пользователях БД.

Атрибуты таблицы:

- ✓ LOGIN — логин пользователя
- ✓ PASSWORD — пароль пользователя
- ✓ PRIOR_TYPE — права пользователя

Ограничения таблицы:

- USERS_PK: LOGIN — первичный ключ таблицы

4. Разработка приложения

Разработка web-приложения базируется на концепции **MVC** (Model-View-Controller).

Model-View-Controller (MVC, «Модель-представление-поведение», «Модель-представление-контроллер») — архитектура программного обеспечения, в которой модель данных приложения, пользовательский интерфейс и управляющая логика разделены на три отдельных компонента, так, что модификация одного из компонентов оказывает минимальное воздействие на другие компоненты. Шаблон *MVC* позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента:

- Модель (Model). Модель предоставляет данные (обычно для View), а также реагирует на запросы (обычно от контроллера), изменяя своё состояние.
- Представление (View). Отвечает за отображение информации (пользовательский интерфейс).
- Поведение (Controller). Интерпретирует данные, введённые пользователем, и информирует модель и представление о необходимости соответствующей реакции.

В соответствии с данной моделью были созданы 3 пакета: **entity**, **database**, **servlet**.

- ✓ **entity** — классы, представляющие собой модели таблиц БД (часть Model).
- ✓ **database** — классы, отвечающие за взаимодействие с БД (часть Model).
- ✓ **servlet** — классы-сервлеты (Controller), получающие данные, пришедшие на сервер после отправки формы из браузера и взаимодействующие с классами пакетов database и entity.

Servlet — является Java-программой, выполняющейся на стороне сервера и расширяющей функциональные возможности сервера. **Servlet** взаимодействует с клиентами посредством принципа запрос-ответ.

Роль View в трёхзвенной архитектуре занимают **jsp**-страницы.

JSP (**JavaServer Pages**) — технология, создания страниц, которые имеют как статические, так и динамические компоненты. Страница **JSP** является текстовым документом, который содержит:

- статическую часть (в формате HTML,XML).
- динамическую часть (использование Java-кода).

JSP — одна из высокопроизводительных технологий, так как весь код страницы транслируется в java-код сервлета, и затем компилируется в байт-код виртуальной машины java (JVM).

Для поддержания связи в проекте существует дескриптор развертывания, в виде XML-документа, в котором указывается адрес, при обращении к которому будет вызываться тот или иной сервлет. При вызове сервлет выполняет порученные ему функции и направляет результат на jsp страницу, которую просматривает пользователь.

5. Описание пользовательского интерфейса (инструкция пользователя)

1. При запуске приложения пользователь видит окно:

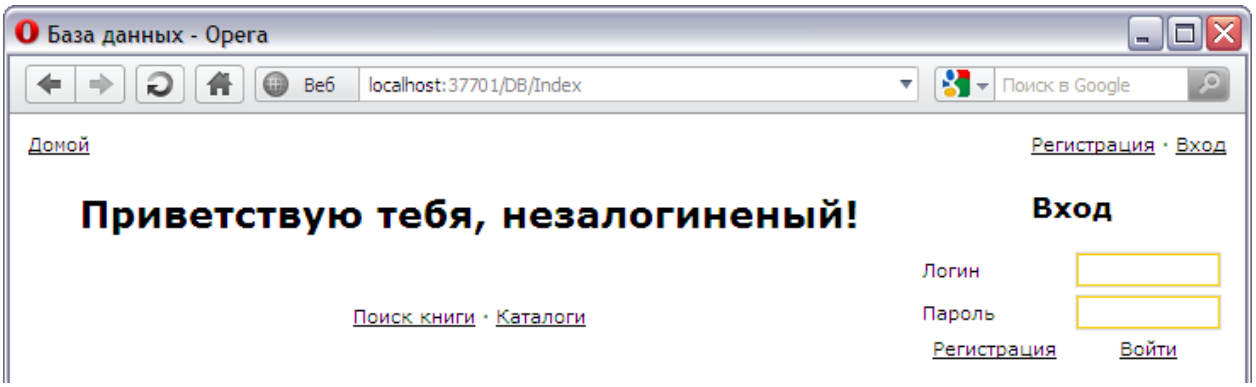


Рисунок 1

2. Пользователь вводит логин, пароль и нажимает войти и в зависимости от прав на экране появятся те или иные ссылки:

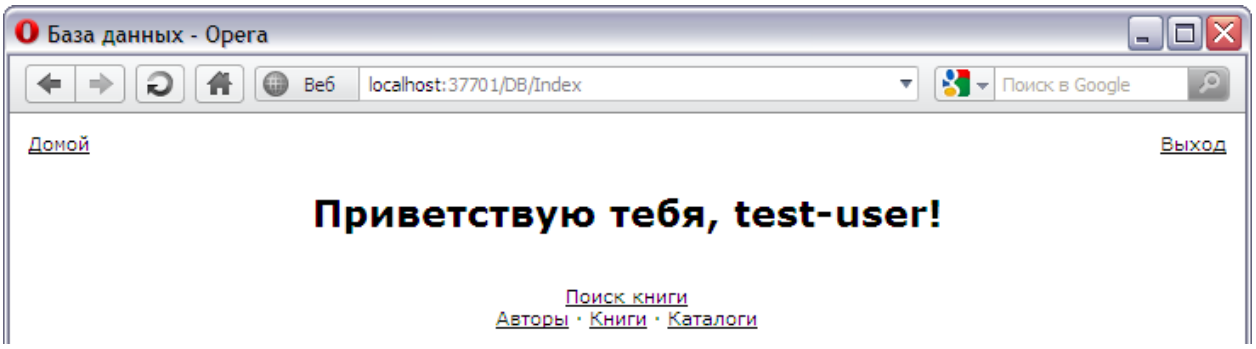


Рисунок 2

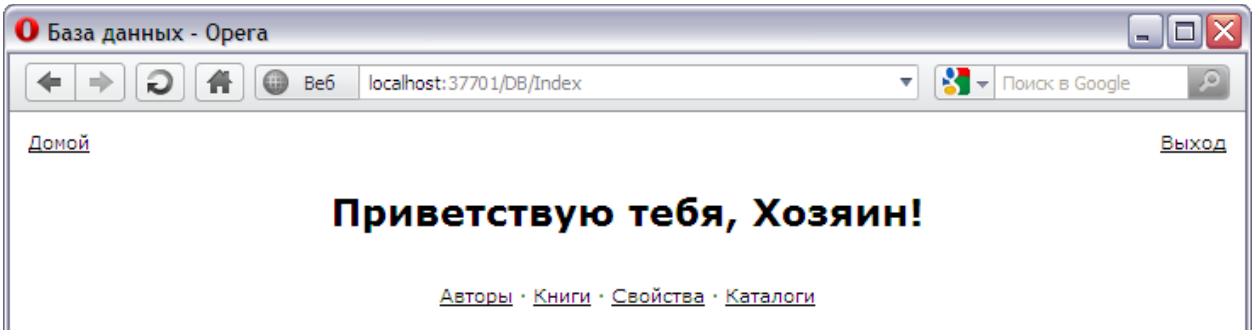
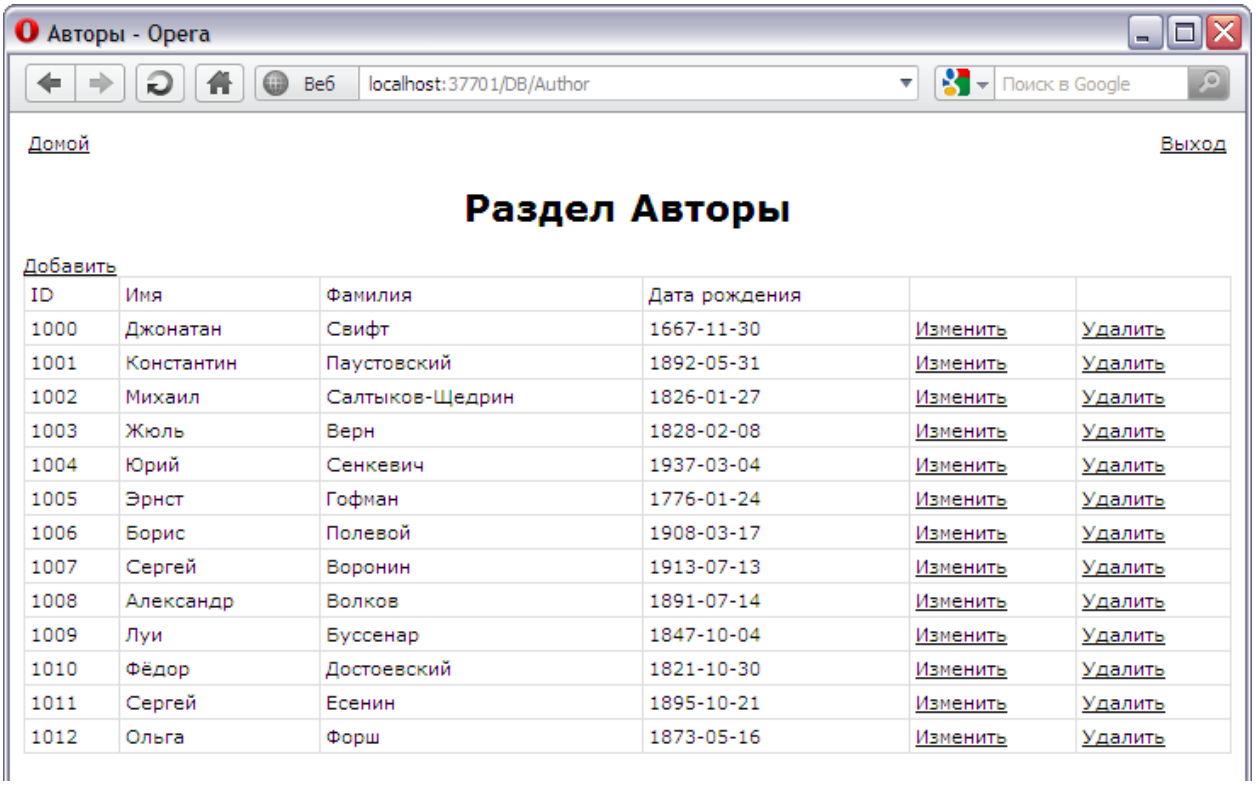


Рисунок 3

3. Выбирается ссылка для просмотра и редактирования и открывается страница с таблицей данных:



ID	Имя	Фамилия	Дата рождения		
1000	Джонатан	Свифт	1667-11-30	Изменить	Удалить
1001	Константин	Паустовский	1892-05-31	Изменить	Удалить
1002	Михаил	Салтыков-Щедрин	1826-01-27	Изменить	Удалить
1003	Жюль	Верн	1828-02-08	Изменить	Удалить
1004	Юрий	Сенкевич	1937-03-04	Изменить	Удалить
1005	Эрнст	Гофман	1776-01-24	Изменить	Удалить
1006	Борис	Полевой	1908-03-17	Изменить	Удалить
1007	Сергей	Воронин	1913-07-13	Изменить	Удалить
1008	Александр	Волков	1891-07-14	Изменить	Удалить
1009	Луи	Буссенар	1847-10-04	Изменить	Удалить
1010	Фёдор	Достоевский	1821-10-30	Изменить	Удалить
1011	Сергей	Есенин	1895-10-21	Изменить	Удалить
1012	Ольга	Форш	1873-05-16	Изменить	Удалить

Рисунок 4

4. Для добавления нужно выбрать «Добавление», а затем заполняется открывшаяся форма «Добавление ***», далее нажимается «Добавить» и происходит пополнение БД данной записью:

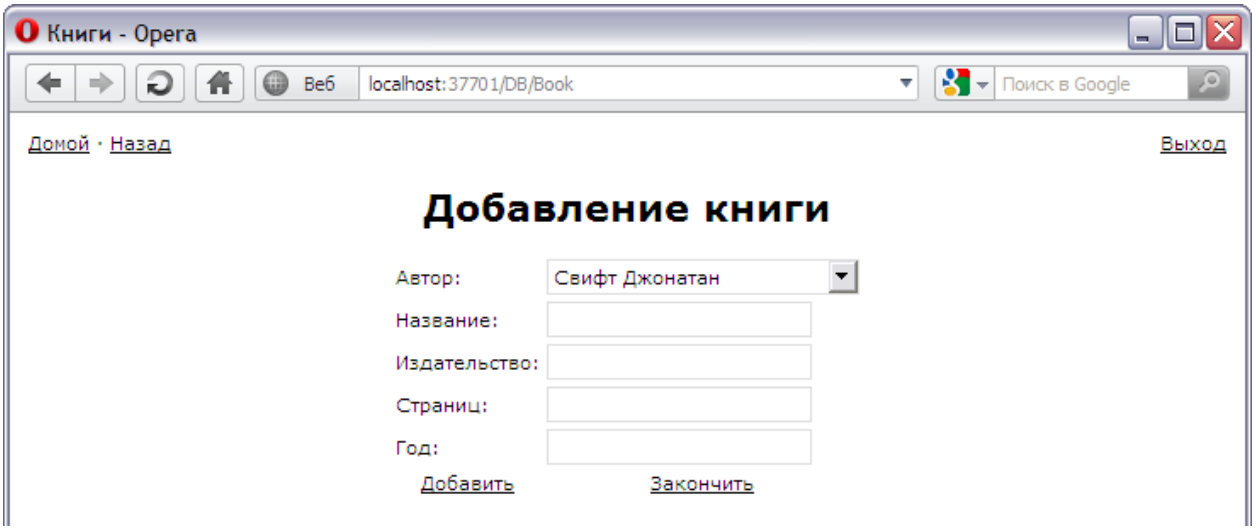


Рисунок 5

5. Для удаления записи нужно нажать ссылку «Удалить», в этом случае запись удалится автоматически.

6. При изменении записи, необходимо нажать «Изменить», система предложит форму для изменения, в которую надо будет ввести данные, и нажать «Изменить».

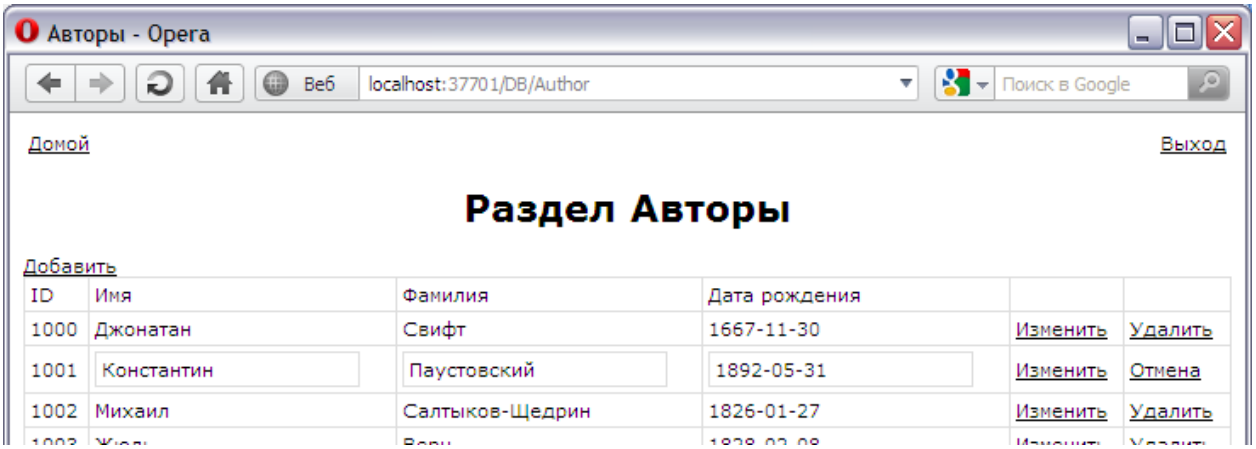


Рисунок 6

7. Сверху всех страниц находится навигационная панель, которая позволяет возвращаться на главную страницу («Домой»), завершать сессию («Выход») и другие возможности (не относящиеся к управлению БД):

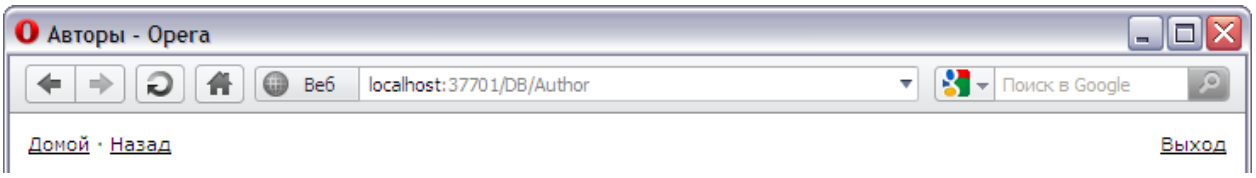


Рисунок 7

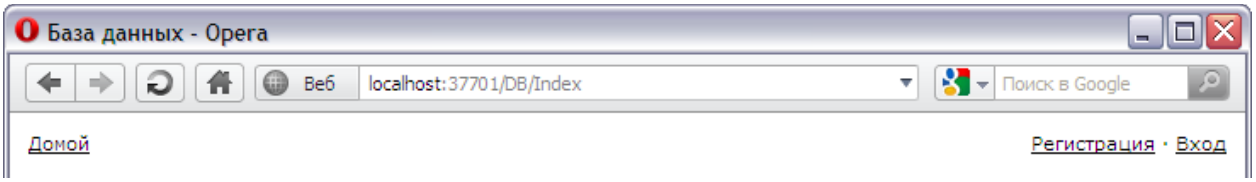


Рисунок 8

8. Для поиска (основной функцией при работе с базой данных), есть страница поиска, на который организован поэтапный поиск:

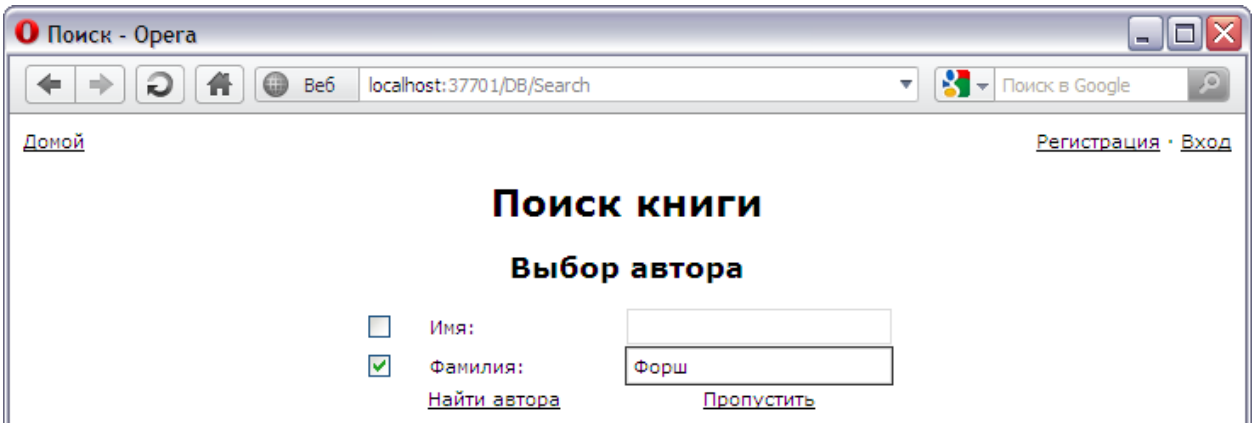


Рисунок 9

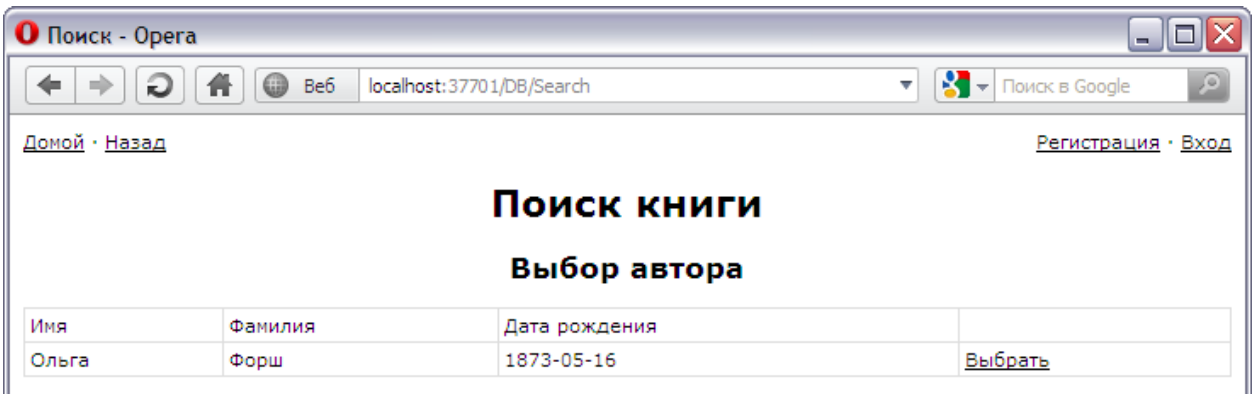


Рисунок 10

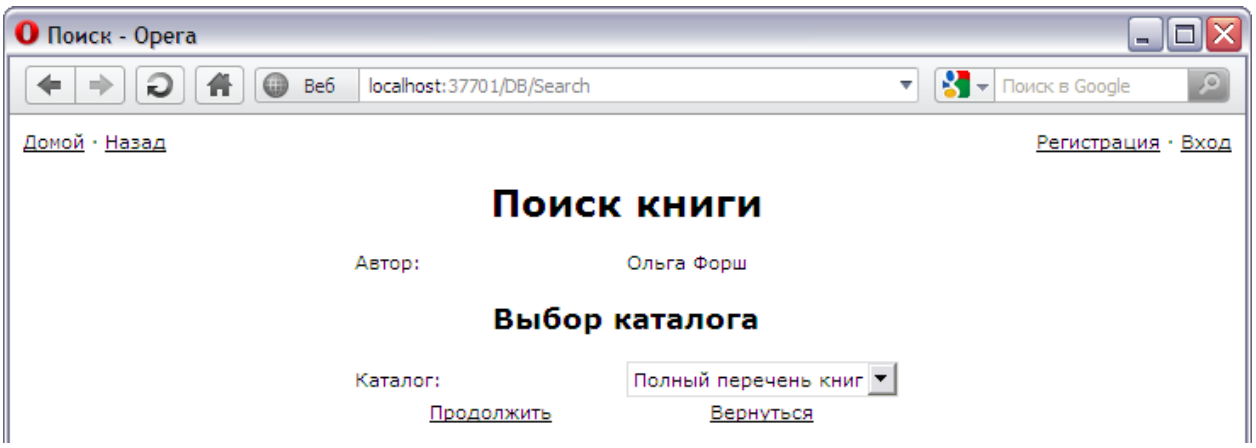


Рисунок 11

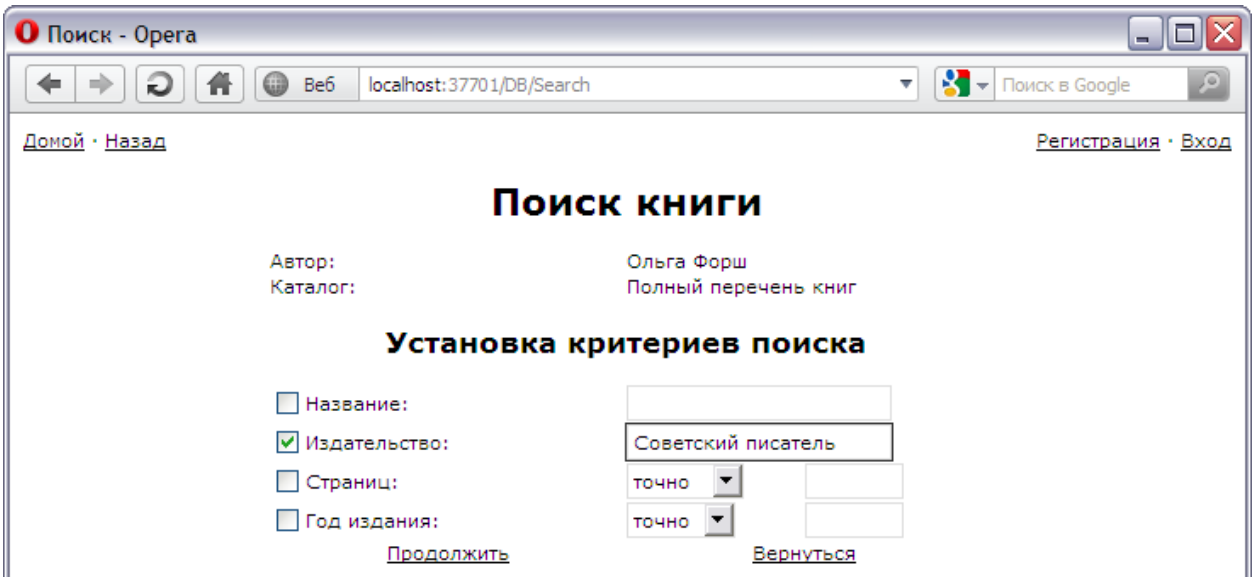


Рисунок 12

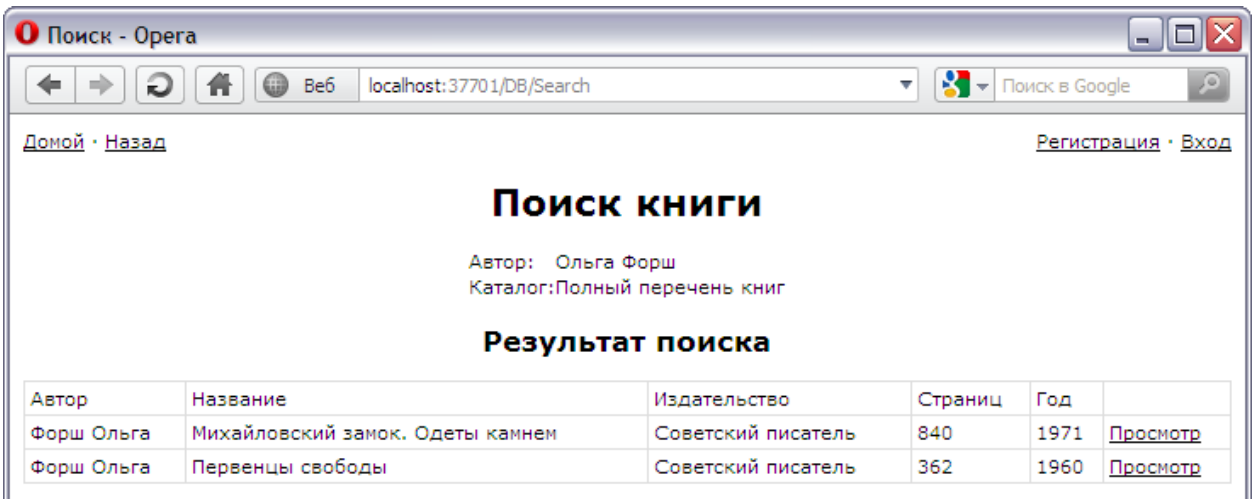


Рисунок 13

9. Для наполнения каталогов содержимым используется специфическая форма редактирования, в одном списке выбираются добавляемые книги, в другом – удаляемые (при начальном наполнении отображается только первый список), по нажатии на ссылку «Изменить», происходит одновременно и удаление, и добавление книг в каталог:

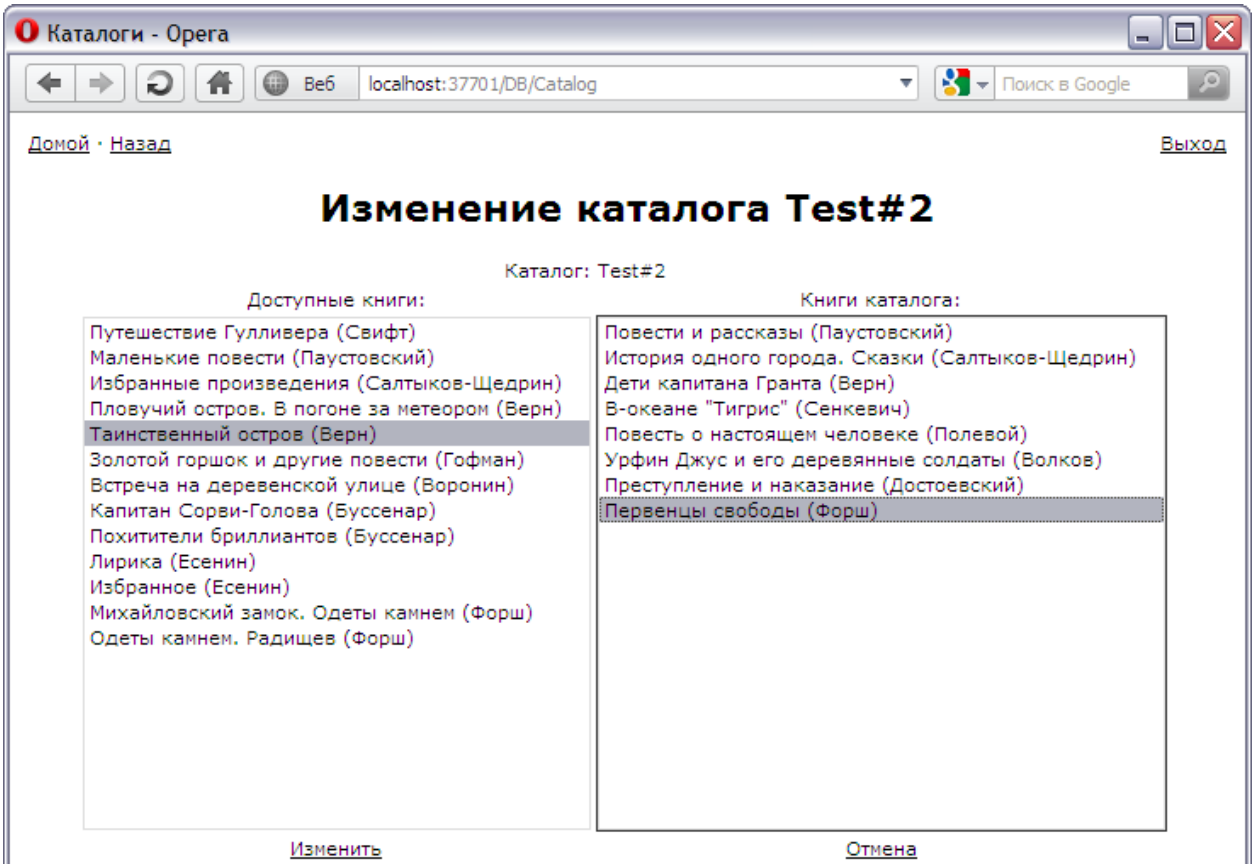


Рисунок 14

6. Список используемой литературы

1. NetBeans IDE 6.9.1. JSP Examples.
2. NetBeans IDE 6.9.1. JSTL Examples.
3. NetBeans IDE 6.9.1. Servlet Examples.
4. Java сервлеты и JSP. Сборник рецептов.2006 – Брюс У. Перии; изд. «КУДИЦ-Образ»
5. OReilly.Head.First.Servlets.and.JSP.2nd.Edition.Mar.2008.
6. www.oracle.com
7. www.htmlbook.ru

Приложения

Listener.java — прослушиватель изменний жизненного цикла сервлета

```
package Servlets;

import DataBase.*;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

public class Listener implements ServletContextListener {

    @Override
    public void contextInitialized(ServletContextEvent sce) {
        DBConnection dbc = new DBConnection();
        if(!dbc.Connect())return;
        sce.getServletContext().setAttribute("dbConnection", dbc);
        sce.getServletContext().setAttribute("dbAuthor", new DBAuthor(dbc));
        sce.getServletContext().setAttribute("dbBook", new DBBook(dbc));
        sce.getServletContext().setAttribute("dbCatalog", new DBCatalog(dbc));
        sce.getServletContext().setAttribute("dbProperty", new DBProperty(dbc));
        sce.getServletContext().setAttribute("dbUser", new DBUser(dbc));
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {
        ((DBConnection)sce.getServletContext().getAttribute("dbConnection")).Disconnect();
        sce.getServletContext().removeAttribute("dbConnection");
        sce.getServletContext().removeAttribute("dbAuthor");
        sce.getServletContext().removeAttribute("dbBook");
        sce.getServletContext().removeAttribute("dbCatalog");
        sce.getServletContext().removeAttribute("dbProperty");
        sce.getServletContext().removeAttribute("dbUser");
    }
}
```

DBConnection.java — класс подключения к БД

```
package DataBase;

import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import oracle.jdbc.OracleDriver;

public class DBConnection {

    private static final String USERNAME = "1";
    private static final String PASSWORD = "1";
    private static final String SERVER = "jdbc:oracle:thin:@server:1521:XE";
    private OracleDriver driver = new OracleDriver();
    private Statement statement = null;
    private Connection connection = null;
    private boolean state = false;

    public boolean Connect() {
        try {
            return sConnect();
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }

    public void Disconnect() {
        try {
            sDisconnect();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public boolean sConnect() throws SQLException, ClassNotFoundException {
        state = false;
        DriverManager.registerDriver(driver);
        if (connection != null) {
            connection.close();
```

```
            connection = null;
        }
        Class.forName("oracle.jdbc.OracleDriver");
        connection = DriverManager.getConnection(SERVER, USERNAME, PASSWORD);
        if (connection == null) {
            return false;
        }
        statement = connection.createStatement();
        if (statement == null) {
            return false;
        }
        DatabaseMetaData conMetaData = connection.getMetaData();
        if (conMetaData == null) {
            return false;
        }
        state = true;
        return true;
    }

    public Statement getStatement() {
        return statement;
    }

    public boolean Active() {
        return state;
    }

    public void sDisconnect() throws SQLException {
        state = false;
        connection.close();
    }

    public Connection getConnection() {
        return connection;
    }
}
```

CBook.java — класс сущности Книга

```
package Objects;

public class CBook {
    int ID, ID_AUTHOR;
    String TITLE, PUBLISHER;
    int PAGES;
    int YEAR;
    CAuthor AUTHOR;

    public CBook(int ID, int ID_AUTHOR, String TITLE, String PUBLISHER, int PAGES, int YEAR) {
        this.ID = ID;
        this.ID_AUTHOR = ID_AUTHOR;
        this.TITLE = TITLE;
        this.PUBLISHER = PUBLISHER;
        this.PAGES = PAGES;
        this.YEAR = YEAR;
    }

    public int getID() { return ID; }
    public int getID_AUTHOR() { return ID_AUTHOR; }
    public void setID_AUTHOR(int ID_AUTHOR) { this.ID_AUTHOR = ID_AUTHOR; }
    public String getTITLE() { return TITLE; }
    public void setTITLE(String TITLE) { this.TITLE = TITLE; }
    public String getPUBLISHER() { return PUBLISHER; }
    public void setPUBLISHER(String PUBLISHER) { this.PUBLISHER = PUBLISHER; }
    public int getPAGES() { return PAGES; }
    public void setPAGES(int PAGES) { this.PAGES = PAGES; }
    public int getYEAR() { return YEAR; }
    public void setYEAR(int YEAR) { this.YEAR = YEAR; }
    public CAuthor getAUTHOR() { return AUTHOR; }
    public void setAUTHOR(CAuthor AUTHOR) { this.AUTHOR = AUTHOR; }

}
```

DBBook.java — класс запросов к БД для сущности Книга

```
package DataBase;

import Objects.CBook;
import java.sql.CallableStatement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.LinkedList;
import java.util.List;

public class DBBook {

    DBConnection dbc;
    DBAuthor dba;

    public DBBook(DBConnection dbc) {
        this.dbc = dbc;
        this.dba = new DBAuthor(dbc);
    }

    public List LoadList() throws SQLException {
        if (!dbc.Active()) { throw new SQLException(); }
        List<CBook> BookList = new LinkedList<CBook>();
        String query = "Select * from Book";
        ResultSet rs = dbc.getStatement().executeQuery(query);
        while (rs.next()) {
            int ID = rs.getInt("ID_BOOK");
            int ID_Author = rs.getInt("ID_AUTHOR");
            String Title = rs.getString("TITLE");
            String Publisher = rs.getString("PUBLISHER");
            int Pages = rs.getInt("PAGES");
            int Year = rs.getInt("YEAR");
            CBook Book = new CBook(ID, ID_Author, Title, Publisher, Pages, Year);
            BookList.add(Book);
        }
        rs.close();
        for (CBook Book : BookList) {
            Book.setAUTHOR(dba.getAuthor(Book.getID_AUTHOR()));
        }
        return BookList;
    }

    public CBook getBook(int ID, List<CBook> BookList){
        CBook Result = null;
        for (CBook Book : BookList) {
            if (Book.getID() == ID) {
                Result = Book;
                break;
            }
        }
        return Result;
    }

    public List searchBook(String Title, String Publisher, String Pages,
            String Year, String ID_Catalog, String ID_Author) throws SQLException {
        if (!dbc.Active()) { throw new SQLException(); }
        List<CBook> BookList = new LinkedList<CBook>();
        String query = "Select * from Book where ";
        if (Title!=null){ query+="Title='"+Title+"' and "; }
        if (Publisher!=null){ query+="Publisher='"+Publisher+"' and "; }
        if (Pages!=null){ query+="Pages"+Pages+" and "; }
        if (Year!=null){ query+="Year"+Year+" and "; }
        if (ID_Author!=null){ query+="ID_Author="+ID_Author+" and "; }
        query+="ID_Book in (Select ID_Book from Book_Catalog where ID_Catalog="+ID_Catalog+")";
        ResultSet rs = dbc.getStatement().executeQuery(query);
        while (rs.next()) {
           int ID = rs.getInt("ID_BOOK");
            int iID_Author = rs.getInt("ID_AUTHOR");
            Title = rs.getString("TITLE");
            Publisher = rs.getString("PUBLISHER");
            int iPages = rs.getInt("PAGES");
            int iYear = rs.getInt("YEAR");
            CBook Book = new CBook(ID, iID_Author, Title, Publisher, iPages, iYear);
            BookList.add(Book);
        }
        rs.close();
        for (CBook Book : BookList) {
            Book.setAUTHOR(dba.getAuthor(Book.getID_AUTHOR()));
```

```
        }
        return BookList;
    }

    public void addBook(int ID_Author, String Title, String Publisher, int Pages, int Year)
throws SQLException {
        if (!dbc.Active()) { throw new SQLException(); }
        CallableStatement cs= dbc.getConnection().prepareCall("{call add_book(?,?,?,?,?)}");
        cs.setInt(1, ID_Author);
        cs.setString(2, Title);
        cs.setString(3, Publisher);
        cs.setInt(4, Pages);
        cs.setInt(5, Year);
        cs.execute();
        cs.close();
    }

    public void updateBook(int ID, int ID_Author, String Title, String Publisher, int Pages,
            int Year) throws SQLException {
        if (!dbc.Active()) { throw new SQLException(); }
        PreparedStatement ps = dbc.getConnection().prepareStatement(
            "Update Book set ID_Author=?,Title=?,Publisher=?,Pages=?,Year=? where ID_Book=?");
        ps.setInt(1, ID_Author);
        ps.setString(2, Title);
        ps.setString(3, Publisher);
        ps.setInt(4, Pages);
        ps.setInt(5, Year);
        ps.setInt(6, ID);
        ps.execute();
        ps.close();
    }

    public void deleteBook(int ID) throws SQLException {
        if (!dbc.Active()) { throw new SQLException(); }
        PreparedStatement ps = dbc.getConnection().prepareStatement(
            "Delete from Book_Properties where ID_Book=?");
        ps.setInt(1, ID);
        ps.execute();
        ps.close();
        ps = dbc.getConnection().prepareStatement("Delete from Book_Catalog where ID_Book=?");
        ps.setInt(1, ID);
        ps.execute();
        ps.close();
        ps = dbc.getConnection().prepareStatement("Delete from Book where ID_Book=?");
        ps.setInt(1, ID);
        ps.execute();
        ps.close();
    }
}
```

Book.java — класс взаимодействия с броузером

```
package Servlets;

import DataBase.DBAuthor;
import DataBase.DBBook;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name="Book", urlPatterns={"/Book"})
public class Book extends HttpServlet {

    DBBook dbBook;
    DBAuthor dbAuthor;

    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        dbBook = (DBBook) config.getServletContext().getAttribute("dbBook");
        dbAuthor = (DBAuthor) config.getServletContext().getAttribute("dbAuthor");
    }
```

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, SQLException {
    request.setCharacterEncoding("UTF-8");
    response.setContentType("text/html;charset=UTF-8");
    request.setAttribute("add", false);
    request.setAttribute("update", false);
    request.setAttribute("result", 0);
    if (request.getParameter("add")!=null)
    {
       request.setAttribute("add", true);
       doAddBook(request, response);
    }
    if (request.getParameter("update")!=null)
    {
       request.setAttribute("update", true);
       request.setAttribute("ID", Integer.parseInt(request.getParameter("ID")));
       doUpdateBook(request, response);
    }
    if (request.getParameter("delete")!=null)
    {
       doDeleteBook(request, response);
    }
    request.setAttribute("BookList", dbBook.LoadList());
    request.setAttribute("AuthorList", dbAuthor.LoadList());
    this.getServletContext().getRequestDispatcher("/Book.jsp").forward(request, response);
}

 protected void doAddBook(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException, SQLException {
     String ID_Author = request.getParameter("ID_AUTHOR");
    String Title = request.getParameter("TITLE");
    String Publisher = request.getParameter("PUBLISHER");
    String Pages = request.getParameter("PAGES");
    String Year = request.getParameter("YEAR");
    if (ID_Author==null || Title==null || Publisher==null || Pages==null || Year==null) {
        return;
    }
    request.setAttribute("result", 1);
    try {
        dbBook.addBook(Integer.parseInt(ID_Author),Title,Publisher,Integer.parseInt(Pages),
            Integer.parseInt(Year));
    } catch (Exception e) {
        request.setAttribute("result", -1);
    }
}

protected void doUpdateBook(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException, SQLException {
    String ID = request.getParameter("ID");
    String ID_Author = request.getParameter("ID_AUTHOR");
    String Title = request.getParameter("TITLE");
    String Publisher = request.getParameter("PUBLISHER");
    String Pages = request.getParameter("PAGES");
    String Year = request.getParameter("YEAR");
    if (ID==null || ID_Author==null || Title==null || Publisher==null || Pages==null ||
        Year==null) {
        return;
    }
    request.setAttribute("result", 2);
    try {
        dbBook.updateBook(Integer.parseInt(ID),Integer.parseInt(ID_Author),Title,Publisher,
            Integer.parseInt(Pages),Integer.parseInt(Year));
    } catch (Exception e) {
        request.setAttribute("result", -2);
    }
}

protected void doDeleteBook(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException, SQLException {
    String ID = request.getParameter("ID");
    if (ID == null) {
        return;
    }
    request.setAttribute("result", 3);
    try {
        dbBook.deleteBook(Integer.parseInt(ID));
    } catch (Exception e) {
        request.setAttribute("result", -3);
    }
```

```
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
            Logger.getLogger(Book.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
            Logger.getLogger(Book.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

}
```

Book.jsp — JSP-страница для ответа броузеру

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@page import="java.util.List" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Книги</title>
        <link rel="stylesheet" type="text/css" href="style.css" />
    </head>
    <body>
        <c:if test="${requestScope.result==1}"><script type="text/javascript">alert("Книга успешно добавлена");</script></c:if>
        <c:if test="${requestScope.result==-1}"><script type="text/javascript">alert("Ошибка: Книга не добавлена");</script></c:if>
        <c:if test="${requestScope.result==2}"><script type="text/javascript">alert("Книга успешно изменена");</script></c:if>
        <c:if test="${requestScope.result==-2}"><script type="text/javascript">alert("Ошибка: Книга не изменена");</script></c:if>
        <c:if test="${requestScope.result==3}"><script type="text/javascript">alert("Книга успешно удалена");</script></c:if>
        <c:if test="${requestScope.result==-3}"><script type="text/javascript">alert("Ошибка: Книга не удалена");</script></c:if>
        <table width="100%" id="zeroTbl">
            <tr>
                <td align="left">
                    <a href="<c:url value="/Index" />">Домой</a>
                    <c:if test="${requestScope.add}">
                        &middot;
                        <a href="<c:url value="/Book" />">Назад</a>
                    </c:if>
                </td>
                <td align="right">
                    <c:if test="${sessionScope.User.TYPE!=sessionScope.User.TYPE_GUEST}">
                        <a href="<c:url value="/Index" />?logout">Выход</a>
                    </c:if>
                    <c:if test="${sessionScope.User.TYPE==sessionScope.User.TYPE_GUEST}">
                        <a href="<c:url value="/Index" />?register">Регистрация</a>
                        &middot;
                        <a href="<c:url value="/Index" />?login">Вход</a>
                    </c:if>
                </td>
            </tr>
        </table>
        <c:if test="${requestScope.add}">
            <h1>Добавление книги</h1>
            <form action ="" method=POST id="addForm">
                <table align="center">
                    <tr>
                        <td>Автор:</td><td>
                            <select name=ID_AUTHOR>
```

```
                        <c:forEach items="${requestScope.AuthorList}" var="obj2">
                            <option value="${obj2.ID}">${obj2.SURNAME} ${obj2.NAME}</option>
                        </c:forEach>
                    </select>
                </td>
            </tr><tr>
                <td>Название:</td><td><input type=text size=20 name=TITLE /></td>
            </tr><tr>
                <td>Издательство:</td><td><input type=text size=20 name=PUBLISHER /></td>
            </tr><tr>
                <td>Страниц:</td><td><input type=text size=20 name=PAGES /></td>
            </tr><tr>
                <td>Год:</td><td><input type=text size=20 name=YEAR /></td>
            </tr><tr>
                <td align="center"><a href="javascript:addForm.submit();">Добавить</a></td>
                <td align="center"><a href="<c:url value="/Book" />">Закончить</a></td>
            </tr>
        </table>
    </form>
</c:if>
<c:if test="${!requestScope.add}">
    <h1>Раздел Книги</h1>
    <a href="<c:url value="/Book" />?add">Добавить</a>
    <form action ="" method=POST id="editForm">
        <table align="center" width="100%" class="mainTbl">
            <tr>
                <td>ID</td><td>Автор</td><td>Название</td><td>Издательство</td><td>Страниц</td><td>Год</td><td></td><td></td><td></td>
            </tr>
            <c:forEach items="${requestScope.BookList}" var="obj">
                <tr>
                    <td>${obj.ID}</td>
                    <td>
                        <c:if test="${(requestScope.ID!=obj.ID || requestScope.result!=0) && obj.AUTHOR!=null}">
                            ${obj.AUTHOR.SURNAME} ${obj.AUTHOR.NAME}
                        </c:if>
                        <c:if test="${requestScope.ID==obj.ID && requestScope.result==0 && obj.AUTHOR!=null}">
                            <select name=ID_AUTHOR>
                                <c:forEach items="${requestScope.AuthorList}" var="obj2">
                                    <option value="${obj2.ID}"<c:if test="${obj.ID_AUTHOR==obj2.ID}"> selected</c:if>>
                                        ${obj2.SURNAME} ${obj2.NAME}
                                    </option>
                                </c:forEach>
                            </select>
                        </c:if>
                        <c:if test="${obj.AUTHOR==null}">ERROR</c:if>
                    </td>
                    <td>
                        <c:if test="${requestScope.ID==obj.ID && requestScope.result==0}">
                            <input type=text size=20  name=TITLE value="
                                </c:if>
```

```
                                ${obj.TITLE}
                                <c:if test="${requestScope.ID==obj.ID && requestScope.result==0}">"/></c:if>
                            </td>
                            <td>
                                <c:if test="${requestScope.ID==obj.ID && requestScope.result==0}">
                                    <input type=text size=20 name=PUBLISHER value="
                                        </c:if>
                                        ${obj.PUBLISHER}
                                        <c:if test="${requestScope.ID==obj.ID && requestScope.result==0}">"/></c:if>
                            </td>
                            <td>
                                <c:if test="${requestScope.ID==obj.ID && requestScope.result==0}">
                                    <input type=text size=5 name=PAGES value="
                                        </c:if>
                                        ${obj.PAGES}
                                        <c:if test="${requestScope.ID==obj.ID && requestScope.result==0}">"/></c:if>
                            </td>
                            <td>
                                <c:if test="${requestScope.ID==obj.ID && requestScope.result==0}">
                                    <input type=text size=5 name=YEAR value="
                                        </c:if>
                                        ${obj.YEAR}
                                        <c:if test="${requestScope.ID==obj.ID && requestScope.result==0}">"/></c:if>
                            </td>
                            <td>
                                <a href="<c:url value="/Property" />?ID_BOOK=${obj.ID}">Свойства</a>
                            </td>
                            <td>
                                <c:if test="${requestScope.ID==obj.ID && requestScope.result==0}">
                                    <a href="javascript:editForm.submit();">Изменить</a>
                                </c:if>
                                <c:if test="${requestScope.ID!=obj.ID || requestScope.result!=0}">
                                    <a href="<c:url value="/Book" />?ID=${obj.ID}&update">Изменить</a>
                                </c:if>
                            </td>
                            <td>
                                <c:if test="${requestScope.ID==obj.ID && requestScope.result==0}">
                                    <a href="<c:url value="/Book" />">Отмена</a>
                                </c:if>
                                <c:if test="${requestScope.ID!=obj.ID || requestScope.result!=0}">
                                    <a href="<c:url value="/Book" />?ID=${obj.ID}&delete">Удалить</a>
                                </c:if>
                            </td>
                        </tr>
                    </c:forEach>
                </table>
            </form>
        </c:if>
    </body>
</html>
```

web.xml — дескриптор развертывания

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
    <listener>
        <description>ServletContextListener</description>
        <listener-class>Servlets.Listener</listener-class>
    </listener>
    <servlet>
        <servlet-name>SearchServlet</servlet-name>
        <servlet-class>Servlets.SearchServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>AuthorServlet</servlet-name>
        <servlet-class>Servlets.Author</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>BookServlet</servlet-name>
        <servlet-class>Servlets.Book</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>CatalogServlet</servlet-name>
        <servlet-class>Servlets.Catalog</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>PropertyServlet</servlet-name>
        <servlet-class>Servlets.Property</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>MainServlet</servlet-name>
        <servlet-class>Servlets.Main</servlet-class>
        <load-on-startup>0</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>SearchServlet</servlet-name>
        <url-pattern>/Search</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>AuthorServlet</servlet-name>
        <url-pattern>/Author</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>BookServlet</servlet-name>
        <url-pattern>/Book</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>CatalogServlet</servlet-name>
        <url-pattern>/Catalog</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>PropertyServlet</servlet-name>
        <url-pattern>/Property</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>MainServlet</servlet-name>
        <url-pattern>/Index</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

SQL-запросы для создания и наполнения БД

Создание таблиц

```
CREATE TABLE AUTHOR
(
  ID_AUTHOR NUMBER NOT NULL,
  NAME VARCHAR2(128 BYTE) NOT NULL,
  SURNAME VARCHAR2(128 BYTE) NOT NULL,
  YEAR DATE NOT NULL
, CONSTRAINT AUTHOR_PK PRIMARY KEY
  (
    ID_AUTHOR
  )
  ENABLE
)
```

```
CREATE TABLE BOOK
(
  ID_BOOK NUMBER NOT NULL,
  ID_AUTHOR NUMBER NOT NULL,
  TITLE VARCHAR2(256 BYTE),
  PUBLISHER VARCHAR2(128 BYTE),
  PAGES NUMBER,
  YEAR NUMBER(4, 0)
, CONSTRAINT BOOK_PK PRIMARY KEY
  (
    ID_BOOK
  )
  ENABLE
)
```

```
CREATE TABLE BOOK_CATALOG
(
  ID_BOOK_CATALOG NUMBER NOT NULL,
  ID_BOOK NUMBER,
  ID_CATALOG NUMBER
, CONSTRAINT BOOK_CATALOG_PK PRIMARY KEY
  (
    ID_BOOK_CATALOG
  )
  ENABLE
)
```

```
CREATE TABLE USERS
(
  LOGIN VARCHAR2(64 BYTE) NOT NULL,
  PASSWORD VARCHAR2(64 BYTE) NOT NULL,
  PRIOR_TYPE CHAR(3 BYTE)
, CONSTRAINT USERS_PK PRIMARY KEY
  (
    LOGIN
  )
  ENABLE
)
```

```
CREATE TABLE BOOK_PROPERTIES
(
  ID_BOOK_PROPERTIES NUMBER NOT NULL,
  ID_BOOK NUMBER NOT NULL,
  ID_PROPERTY NUMBER NOT NULL,
  VALUE VARCHAR2(256 BYTE)
, CONSTRAINT BOOK_PROPERTIES_PK PRIMARY KEY
  (
    ID_BOOK_PROPERTIES
  )
  ENABLE
)
```

```
CREATE TABLE PROPERTY
(
  ID_PROPERTY NUMBER NOT NULL,
  TITLE VARCHAR2(128 BYTE) NOT NULL
, CONSTRAINT PROPERTY_PK PRIMARY KEY
  (
    ID_PROPERTY
  )
  ENABLE
)
```

```
CREATE TABLE CATALOG
(
  ID_CATALOG NUMBER NOT NULL,
  PARENT_ID_CATALOG NUMBER,
  TITLE VARCHAR2(128 BYTE)
, CONSTRAINT CATALOG_PK PRIMARY KEY
  (
    ID_CATALOG
  )
  ENABLE
)
```

Создание связей посредством внешних ключей

```
ALTER TABLE BOOK
ADD CONSTRAINT BOOK_AUTHOR_FK FOREIGN KEY
(
  ID_AUTHOR
)
REFERENCES AUTHOR
(
  ID_AUTHOR
)
 ENABLE
;
```

```
ALTER TABLE BOOK_CATALOG
ADD CONSTRAINT BOOK_CATALOG_BOOK_FK FOREIGN
KEY
(
  ID_BOOK
)
REFERENCES BOOK
(
  ID_BOOK
)
 ENABLE
;
```

```
ALTER TABLE CATALOG
ADD CONSTRAINT CATALOG_CATALOG_FK FOREIGN KEY
(
  PARENT_ID_CATALOG
)
REFERENCES CATALOG
(
  ID_CATALOG
)
 ENABLE
;
```

```
ALTER TABLE BOOK_CATALOG
ADD CONSTRAINT BOOK_CATALOG_CATALOG_FK
FOREIGN KEY
(
  ID_CATALOG
)
REFERENCES CATALOG
(
  ID_CATALOG
)
 ENABLE
;
```

```
ALTER TABLE BOOK_PROPERTIES
ADD CONSTRAINT BOOK_PROPERTIES_BOOK_FK
FOREIGN KEY
(
  ID_BOOK
)
REFERENCES BOOK
(
  ID_BOOK
)
 ENABLE
;
```

```
ALTER TABLE BOOK_PROPERTIES
ADD CONSTRAINT BOOK_PROPERTIES_PROPERTY_FK
FOREIGN KEY
(
  ID_PROPERTY
)
REFERENCES PROPERTY
(
  ID_PROPERTY
)
 ENABLE
;
```

Создание Sequences'ов для генерации уникальных идентификаторов

```
CREATE SEQUENCE AUTHOR_ INCREMENT BY 1 MAXVALUE 1999 MINVALUE 1000 CACHE 20;
CREATE SEQUENCE PROPERTY_ INCREMENT BY 1 MAXVALUE 2999 MINVALUE 2000 CACHE 20;
CREATE SEQUENCE BOOK_ INCREMENT BY 1 MAXVALUE 4999 MINVALUE 3000 CACHE 20;
CREATE SEQUENCE CATALOG_ INCREMENT BY 1 MAXVALUE 5999 MINVALUE 5000 CACHE 20;
CREATE SEQUENCE B_PROPERTIES_ INCREMENT BY 1 MAXVALUE 7999 MINVALUE 6000 CACHE 20;
CREATE SEQUENCE B_CATALOG_ INCREMENT BY 1 MAXVALUE 9999 MINVALUE 8000 CACHE 20;
```

Заполнение таблицы Author

```
insert into author values (author_.nextval,'Джонатан','Свифт','30-NOV-1667');
insert into author values (author_.nextval,'Константин','Паустовский','31-MAY-1892');
insert into author values (author_.nextval,'Михаил','Салтыков-Щедрин','27-JAN-1826');
insert into author values (author_.nextval,'Жюль','Верн','08-FEB-1828');
insert into author values (author_.nextval,'Юрий','Сенкевич','04-MAR-1937');
insert into author values (author_.nextval,'Эрнст','Гофман','24-JAN-1776');
insert into author values (author_.nextval,'Борис','Полевой','17-MAR-1908');
insert into author values (author_.nextval,'Сергей','Воронин','13-JUL-1913');
insert into author values (author_.nextval,'Александр','Волков','14-JUL-1891');
insert into author values (author_.nextval,'Луи','Буссенар','04-OCT-1847');
insert into author values (author_.nextval,'Фѐдор','Достоевский','30-OCT-1821');
insert into author values (author_.nextval,'Сергей','Есенин','21-OCT-1895');
```

Заполнение таблицы Book

```
insert into book values (book_.nextval,1000,'Путешествие Гулливера','Минск Валев',1992,335);
insert into book values (book_.nextval,1001,'Маленькие повести','Карелия',1978,286);
insert into book values (book_.nextval,1001,'Повести и рассказы','Лениздат',1989,672);
insert into book values (book_.nextval,1002,'Избранные произведения','Детская литература',1976,
                                                                                        480);
insert into book values (book_.nextval,1002,'История одного города. Сказки','Лениздат',1976,288);
insert into book values (book_.nextval,1003,'Дети_капитана Гранта','Радянська школа',1985,416);
insert into book values (book_.nextval,1003,'Пловучий остров. В погоне за метеором','Баку',1957,
                                                                                        547);
insert into book values (book_.nextval,1003,'Таинственный остров','Минск Юнацтво',1984,543);
insert into book values (book_.nextval,1004,'В океане "Тигрис"','Советская Россия',1982,240);
insert into book values (book_.nextval,1005,'Золотой горшок и другие повести',
                                                        'Детская литература',1983,366);
insert into book values (book_.nextval,1006,'Повесть о настоящем человеке','Правда',1977,336);
insert into book values (book_.nextval,1007,'Встреча на деревенской улице','Советский писатель',
                                                                                1980,288);
insert into book values (book_.nextval,1008,'Урфин Джус и его деревянные солдаты',
                                                         'Советская_Россия',1963,236);
insert into book values (book_.nextval,1009,'Капитан Сорви-Голова','АгроЭкспресс',1991,265);
insert into book values (book_.nextval,1009,'Похитители бриллиантов','Минск Юнацтво',1982,382);
insert into book values (book_.nextval,1010,'Преступление и наказание',
                                                    'Художественная литература',1979,560);
insert into book values (book_.nextval,1011,'Лирика','Западно-Сибирское',1977,112);
insert into book values (book_.nextval,1011,'Избранное','Лениздат',1975,496);
```

Заполнение таблицы Catalog

```
insert into catalog values (catalog_.nextval,null,'Полный перечень книг');
```

Заполнение таблицы Property

```
insert into property values (property_.nextval,'Количество');
```

Заполнение таблицы Book Properties

```
insert into book_properties values (b_properties_.nextval,3015,2000,'2');
```

Заполнение таблицы Users

```
insert into property users ('Vasena','2580',126);
```

Заполнение таблицы Book Catalog

```
insert into book_catalog values (b_catalog_.nextval,3000,4001);
insert into book_catalog values (b_catalog_.nextval,3001,4001);
insert into book_catalog values (b_catalog_.nextval,3002,4001);
insert into book_catalog values (b_catalog_.nextval,3003,4001);
insert into book_catalog values (b_catalog_.nextval,3004,4001);
insert into book_catalog values (b_catalog_.nextval,3005,4001);
insert into book_catalog values (b_catalog_.nextval,3006,4001);
insert into book_catalog values (b_catalog_.nextval,3007,4001);
insert into book_catalog values (b_catalog_.nextval,3008,4001);
insert into book_catalog values (b_catalog_.nextval,3009,4001);
insert into book_catalog values (b_catalog_.nextval,3010,4001);
insert into book_catalog values (b_catalog_.nextval,3011,4001);
insert into book_catalog values (b_catalog_.nextval,3012,4001);
insert into book_catalog values (b_catalog_.nextval,3013,4001);
insert into book_catalog values (b_catalog_.nextval,3014,4001);
insert into book_catalog values (b_catalog_.nextval,3015,4001);
insert into book_catalog values (b_catalog_.nextval,3016,4001);
insert into book_catalog values (b_catalog_.nextval,3017,4001);
```

Создание собственных процедур для работы с БД

```
CREATE procedure add_book
( book_author_id in book.id_author%type,
  book_title in book.title%type,
  book_publisher in book.publisher%type,
  book_pages in book.pages%type,
  book_year in book.year%type
) as
cursor c_catalog
is select id_catalog from catalog
where parent_id_catalog is null;
v_catalog_id catalog.id_catalog%type;
begin
insert into book
values(book_.nextval,book_author_id,book_title,book_publisher,book_pages,book_year);
open c_catalog;
loop
fetch c_catalog into v_catalog_id;
exit when c_catalog%notfound;
insert into book_catalog values (b_catalog_.nextval,book_.currval,v_catalog_id);
end loop;
close c_catalog;
end add_book;
```

```
CREATE procedure delete_book_from_catalog
( book_id in book.id_book%type
, catalog_id in catalog.id_catalog%type
) as
cursor c_catalog
is select id_catalog from catalog
where parent_id_catalog=catalog_id;
v_child_catalog_id catalog.id_catalog%type;
begin
open c_catalog;
loop
fetch c_catalog into v_child_catalog_id;
exit when c_catalog%notfound;
delete_book_from_catalog(
  book_id,v_child_catalog_id);
end loop;
close c_catalog;
delete from book_catalog
where id_catalog=catalog_id and
id_book=book_id;
end delete_book_from_catalog;
```

```
CREATE procedure delete_catalog
( catalog_id in catalog.id_catalog%type
) as
cursor c_catalog
is select id_catalog from catalog
where parent_id_catalog=catalog_id;
v_child_catalog_id catalog.id_catalog%type;
begin
open c_catalog;
loop
fetch c_catalog into v_child_catalog_id;
exit when c_catalog%notfound;
delete_catalog(v_child_catalog_id);
end loop;
close c_catalog;
delete from book_catalog
where id_catalog=catalog_id;
delete from catalog
where id_catalog=catalog_id;
end delete_catalog;
```