МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 Информационные системы и технологии

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ**

по дисциплине «Базы данных»

Тема: «Реализация базы данных книжного магазина с использованием технологий резервного копирования и восстановления»

Исполнитель

студент 2 курса 1 группы Шишова Татьяна Сергеевна

(Ф.И.О.)

Руководитель старший преподаватель кафедры ИСиТ *Блинова Е.А.* (учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой ­

Руководитель Блинова Е.А.

(подпись)

**Оглавление**

[Введение 3](#_Toc135899602)

[1.Постановка задачи 4](#_Toc135899603)

[1.1 Обзор аналогичных решений 4](#_Toc135899604)

[2. Проектирование базы данных 14](#_Toc135899605)

[2.1. Разработка структуры базы данных 14](#_Toc135899606)

[2.2. Пользовательские роли. 14](#_Toc135899607)

[3. Разработка объектов базы данных 16](#_Toc135899608)

[3.1. Таблицы. 16](#_Toc135899609)

[3.2. Пользователи. 19](#_Toc135899610)

[3.3. Хранимые процедуры 20](#_Toc135899611)

[3.4. Индексы. 21](#_Toc135899612)

[3.5. Последовательности и триггеры. 21](#_Toc135899613)

[4.Описание процедур импорта и экспорта данных 22](#_Toc135899614)

[4.1 Описание процедуры импорта данных. 22](#_Toc135899615)

[4.2 Описание процедуры экспорта данных 22](#_Toc135899616)

[5. Описание технологии и ее применения в базе данных 24](#_Toc135899617)

[6. Тестирование производительности 29](#_Toc135899618)

[7. Руководство пользователя 31](#_Toc135899619)

[Заключение 34](#_Toc135899620)

[Cписок источников 35](#_Toc135899621)

[ПРИЛОЖЕНИЕ А 36](#_Toc135899622)

[ПРИЛОЖЕНИЕ Б 39](#_Toc135899623)

# Введение

База данных – это совокупность взаимосвязанных данных. Благодаря ей можно хранить большой объём данных, а с помощью запросов получать необходимые данные в упорядоченном виде. Системой для управления базой данных была выбрана СУБД Oracle. Oracle является одной из самых популярных БД и уже давно занимает основную долю на рынке реляционных баз данных. Многие крупные торговые системы в мире основаны на базах данных Oracle. Oracle предоставляет широкий спектр решений для управления базами данных и приложениями, а также облачных решений для бизнеса. С помощью Oracle можно: создавать и управлять базами данных, разрабатывать приложения на языках программирования Java и PL/SQL, управлять бизнес-процессами с помощью приложений Oracle BPM, а также анализировать данные и строить отчеты с помощью Oracle Business Intelligence.

Темой курсового является разработка книжного магазина. Книги являются одной из главных составляющих нашей жизни. С помощью книг и литературы мы можем получать и делиться разной информацией. Книжные магазины в свою очередь выступают самым популярным способом приобретения литературы, это вызвано тем, что многие люди предпочитают читать книгу держа в руках, а не взаимодействовать с ее электронным вариантом. Так как подобного рода магазины пользуются большой популярностью, а, как известно, книг в мире несчётное количество, для оптимизированной работы магазина необходимо систематизировать большие объемы информации о книгах, клиентах и заказах. В этом случае как раз может помочь база данных, разработке которой и посвящен данный курсовой проект.

# 1.Постановка задачи

# 1.1 Обзор аналогичных решений

Первым аналогичным решением будет книжный магазин “Белкнига”. Веб-сайт belkniga.by имеет одну цветовую палитру: белый, серый и зеленый. При переходе в раздел каталог может показаться, что классификация книг по жанрам невелика, но при выборе определенной группы появляется более обширный выбор тематики и ниже представлена литература данной группы. (Рисунок 1.1 а-б).

Одним из преимуществ данного сайта является то, что у определенного жанра и поджанра есть свой уникальный идентификатор, он представлен на рисунке 1.1 в. При открытии книги появляется информация о ней: название, год издания, автор, издательство, формат и т.д. (Рисунок 1.1 г-д)

Преимущества данного веб-сайта:

* Жанры и поджанры имеют идентификатор;
* У книг указан ISBN (Международный стандартный книжный номер) ;
* Большое разнообразие жанров и поджанров.

Недостатки:

* Веб-сайт очень медленно работает;
* У книг нету уникального номера для магазина;
* Отсутствуют примеры страниц у книг;
* Отсутствует информация об импортерах;
* У некоторых книг не хватает характеристик или описания;
* Присутствует “неопределенный” жанр, к которому приобщают книги, которые не подходят под другие жанры.
* Вход в аккаунт только по почте или телефону

Вывод: данный веб-сайт не является самым качественным и лучшим аналогом среди книжных магазинов, так как у него достаточно много существенных недостатков, а преимуществ очень мало.

|  |  |
| --- | --- |
| а | |
| б |
| в |
| г |
| д |

Рисунок 1.1 веб-сервис belkniga.by: a — страница каталог, б — результат выбора жанра из раздела каталог, в — пример идентификатора для поджанра,

г — страница при выборе книги, д — характеристики к книге.

Вторым аналогичным решением является веб-магазин Oz.by. В данном интернет-магазине можно приобрести большее количество разнообразной литературы и не только. При открытии раздела “Книги” появляется страница с выбором наиболее популярных жанров, информация о новой литературе, ссылка на “топ-1000 книг с выгодными скидками”, а также ссылки на подборки книг в подарок. (Рисунок 1.2 а).

На сайте Oz.by большой выбор литературы по фильтром: начиная от фильтрации по жанрам, авторам и году издания заканчивая типом обложки, рейтингу и издателем. (Рисунок 1.2 б).

На странице с книгой присутствует название, уникальный номер, фото внешнего вида и примеры страниц книги, также есть описание и комментарий автора. Чуть ниже находятся необходимые характеристики книги и отзывы покупателей. (Рисунок 1.2 в-г).

Для входа в аккаунт в этом книжном присутствует возможность войти по почте или номеру телефона, а также Вконтакте, mail.ru, Google+, что очень упрощает вход и регистрацию. (Рисунок 1.2 д).

Преимущества данного веб-магазина:

* Большой выбор литературы;
* Выгодные скидки;
* Есть отзывы покупателей;
* Есть вся необходимая информация в характеристиках книги;
* Большое разнообразие фильтров;
* Сайт выполнен в приятном стилевом оформлении;
* Большое количество подборок литературы;
* Разнообразная возможность получить доступ к своему аккаунту или зарегистрироваться.
* При покупке книг накапливаются бонусы и дается скидка на товары

Недостатки:

* На некоторые книги завышена цена;
* Тяжело найти непопулярную литературу.

Вывод: данный веб-сайт представляет из себя очень хороший и качественный книжный магазин, с минимальными недостатками и с большим количеством преимуществ.

|  |
| --- |
| а |
| б |
| в |
| г |
| д |

Рисунок 1.2 веб-сервис OZ.by: a — страница “Книги”, б — выбор фильтров, в — страница книги, г — пример характеристики книги,

д — окно регистрации и входа

Abebooks.com является американским веб-магазином книг. На главной странице сайта сразу же находится поле для поиска книги по автору, названию ключевому слову или ISBN. В данном веб-магазине присутствует большое количество подборок. Одним из преимуществ, по сравнению с предыдущими аналогами это раздел “Used books”, который позволяет подарить вторую жизнь уже использованным книгам. (Рисунок 1.3 а-б).

Данный веб-сайт имеет возможность расширенного поиска, в котором указывается не только название и автор книги, но и можно выбрать издателя, а также сортировку результата. (Рисунок 1.3 в).

При открытии страницы с книгой появляется ее описание, оценка пользователей и цены, в зависимости от того, новая книга или уже использованная, а также тип ее переплета. Ниже находится рекомендация книг с похожей тематикой и еще ниже указаны цены на эту книгу в зависимости от места, где ее можно купить. (Рисунок 1.2 г).

Преимущества данного веб-магазина:

* Большой выбор литературы;
* Можно дать вторую жизнь книге;
* Есть отзывы покупателей;
* Большое разнообразие фильтров;
* Сайт выполнен в приятном стилевом оформлении;
* Большое количество подборок литературы.

Недостатки:

* Нету характеристик книги
* Нету литературы на других языках кроме английского.
* Вход в аккаунт только по номеру телефона или почте

Вывод: данный веб-сайт является очень неплохим аналогом. На данном сайте выполняется почти все необходимые функциональные требования, особенно глобальный поиск здесь выполнен очень качественно и хорошо, но очень существенным недостатком является отсутствие характеристик книг, так что данный аналог и не худший и не лучший.

|  |
| --- |
| а |
| б |
| в |
| г |

Рисунок 1.3 веб-сервис abebooks.com: a — главная страница,

б — подборки литературы, в — продвинутый поиск, г — страница с информацией о книге и где ее можно купить.

Вывод: в ходе написания данного подраздела, были рассмотрены и проанализированы аналогичные решения. Были составлены преимущества и недостатки каждого аналога и выявлено, что наиболее качественным книжным магазином, с большим выбором книг и фильтрацией оказался веб-сайт магазина Oz.by. Для разработки своего продукта необходимо будет на примере Oz.by разработать правильно структурированную и оформленную базу данных с книгами, со всеми необходимыми характеристиками, чтобы пользователю было проще найти необходимую книгу и осуществлять фильтрацию во время поиска. Фильтрацию планируется оформить на примере Oz.by: возможность поиска по типу литературы, автору, году изданию, типу обложки, языку, издательству, наличию. Также для создания аккаунта, на примере Oz.by необходимо будет указать разнообразные вариации для регистрации и входа в аккаунт: вход через телефон, email, google+, ВКонтакте.

# 2. Проектирование базы данных

2.1. Разработка структуры базы данных**.**

Одним из самых первых этапов в разработке БД является определение того, какие таблицы будут использоваться и какие связи будут между ними. В ходе анализа было определено, что будут созданы таблицы: Author, Authors, Book, Genre, Genres, PublisherHouse, Orders, OrderInf, UserInf, UserAccount.

Схема разработанной базы данных со всеми связями:

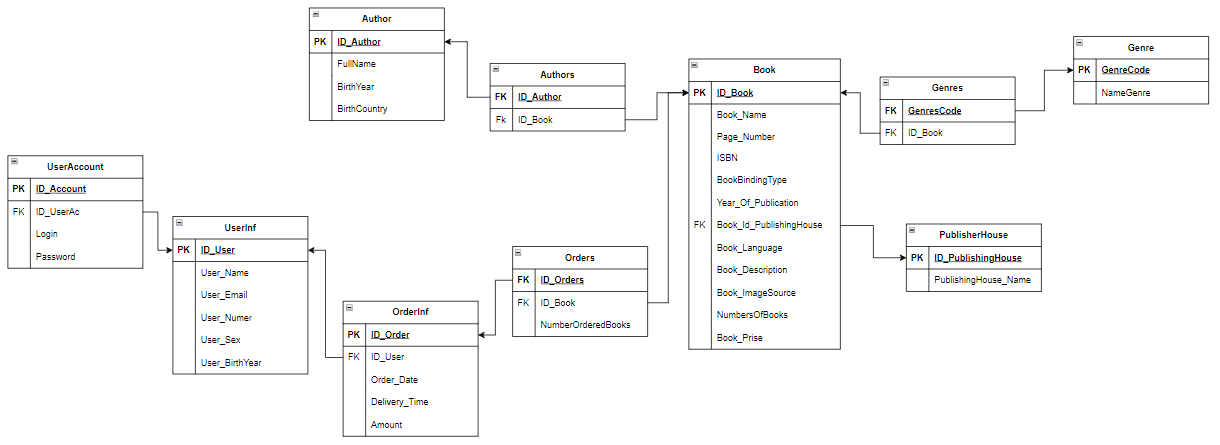


Рисунок 2.1 Схема базы данных

Как можно заметить из схемы все данные сгруппированы по логически соответствующим им таблицам, а также содержат логические связи.

# 2.2. Пользовательские роли.

Для корректной работы с данными в БД, следует определить какие пользовательские роли будут созданы и какие возможности работы с БД они будут иметь.

Всего будет присутствовать 3 роли: гость (Guest), пользователь (User), администратор (Administrator).

При посещении веб-сайта без регистрации все пользователи получают роль гость**.** В этой роли у пользователя есть возможности:

* Просматривать каталог с книгами;
* Смотреть характеристики книги;
* Смотреть отзывы о книге.

При регистрации роль пользователя из гость переходит в пользователь. В роли User сохраняются возможности Guest и добавляются следующие возможности:

* Добавить в избранные книги;
* Добавить книгу в корзину;
* Заказать книгу;
* Если книги нет в наличии, можно оставить на нее заявку, чтобы при ее появлении пользователю пришло сообщение;
* Писать комментарии и отзывы о книгах.

При подключении пользователя в режим Administrator, ему доступно:

* + Такие же возможности, что и User-пользователям;
  + Удалять и редактировать информацию о наличие книг;
  + Собирать информацию о заказах;
  + Выполнять проверку и обработку комментариев;
  + Вносить изменения в БД с информацией о книгах

Ниже представлена Use Case диаграмма для лучшего понимания, что может делать пользователь в определенной роли.

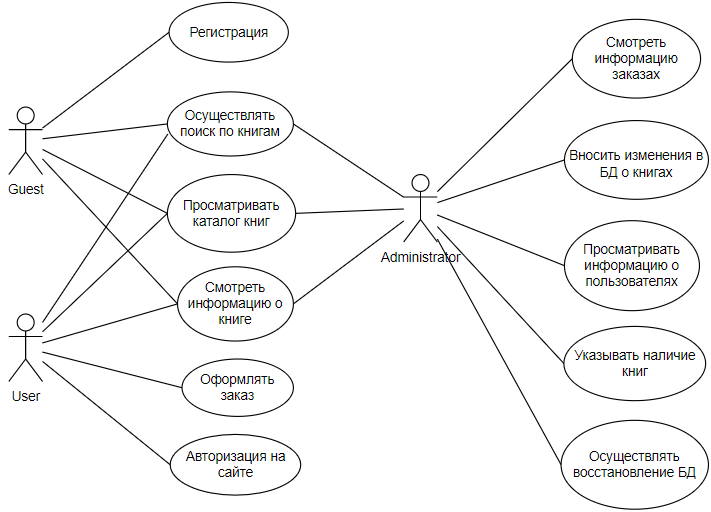


Рисунок 2.2 Use Case диаграмма

Вывод: Входе разработки данного раздела были определенны необходимые таблицы, их поля и ограничения, а также связи между таблицами. Были определены необходимые роли и их возможности. Была разработана Use Case диаграмма.

# 3. Разработка объектов базы данных

Перед разработкой объектов для нашей базы данных необходимо её создать. Так для работы была создана база данных orclCurs в рамках СУБД Oracle.

# 3.1. Таблицы.

Для разработки проекта “Книжного магазина” необходимо было создать 10 таблиц: Author, Authors, Book, Genre, Genres, PublisherHouse, Orders, OrderInf, UserInf, UserAccount. Ниже представлены таблицы с определенными им типами данных,описанием значения каждого столбца, а также со всеми ограничениями.

Таблица 2.1 – Book

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | constrains | Значение поля |
| ID\_Book | int | PK | Первичный ключ |
| Book\_Name | varchar(200) | NOT NULL | Название книги |
| Page\_Number | number | NOT NULL | Количество страниц |
| ISBN | varchar(20) | NOT NULL | Международный стандартный книжный номер |
| BookBindingType | varchar(40) | NOT NULL | Тип переплёта |
| Year\_Of\_Publication | number | NOT NULL | Год издания |
| Book\_ID\_PublishingHouse | number | FK | Внешний ключ к PublisherHouse (ID\_PublishingHouse) |
| Book\_Language | varchar(30) | NOT NULL | Язык книги |
| Book\_Description | varchar(3000) | NOT NULL | Описание книги |
| Book\_ImageSource | varchar(40) | NOT NULL | Изображение книги |
| NumbersOfBooks | number |  | Количество книг в наличии |
| Book\_Prise | Number(5,2) | NOT NULL | Цена книги |

Данная таблица демонстрирует содержание самой главной таблицы с информацией о книгах.

Таблица 2.2 – Authors

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | constrains | Значение поля |
| ID\_Author | int | FK | Внешний ключ к Author(ID\_Author) |
| ID\_Book | int | FK | Внешний ключ к Book(ID\_Book) |

Таблица Authors является промежуточной для связи авторов с книгой. Используется для того, чтобы книга могла быть корректно отображать соавторов.

Таблица 2.3 – Author

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | constrains | Значение поля |
| ID\_Author | int | PK | Первичный ключ |
| FullName | varchar(60) | NOT NULL | ФИО автора |
| BirthYear | number |  | Год рождения |
| BirthCountry | varchar(30) | NOT NULL | Страна рождения |

Данная таблица содержит необходимую информацию об авторах книги.

Таблица 2.4 – Genres

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | constrains | Значение поля |
| GenresCode | int | FK | Внешний ключ к Genre(GenreCode) |
| ID\_Book | int | FK | Внешний ключ к Book(ID\_Book) |

Таблица Genres является промежуточной для связи жанров с книгой. Используется для того, чтобы книга могла содержать несколько жанров.

Таблица 2.5 – Genre

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | constrains | Значение поля |
| GenreCode | int | PK | Первичный ключ |
| NameGenre | varchar(60) | NOT NULL | Название жанра |

Genre позволяет указать название жанра.

Таблица 2.6 – PublisherHouse

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | constrains | Значение поля |
| ID\_PublishingHouse | int | PK | Первичный ключ |
| PublishingHouse\_Name | varchar(30) | NOT NULL | Название издательства |

Данная таблица хранит информацию об издательстве, выпустившем книгу.

Таблица 2.7 – Orders

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | constrains | Значение поля |
| ID\_Orders | int | FK | Внешний ключ к Order(ID\_Order) |
| ID\_Book | int | FK | Внешний ключ к Book(ID\_Book) |
| NumberOrderedBooks | number | NOT NULL | Количество заказанных книг |

Таблица Orders является промежуточной для комбинирования заказов если пользователь решит заказать несколько книг.

Таблица 2.8 – OrderInf

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | constrains | Значение поля |
| ID\_Order | int | PK | Первичный ключ |
| ID\_User | int | FK | Внешний ключ к User(ID\_User) |
| Order\_Date | date | NOT NULL | Дата заказа |
| Delivery\_Time | date | NOT NULL | Срок доставки |
| Amount | Number(5,2) | NOT NULL | Сумма заказа |

Данная таблица содержит подробную информацию о заказе.

Таблица 2.9 – UserInf

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | constrains | Значение поля |
| ID\_User | number | PK | Первичный ключ |
| User\_Name | varchar(100) | NOT NULL | ФИО пользователя |
| User\_Email | varchar(40) | NOT NULL | Почта пользователя |
| User\_Numer | varchar(10) | NOT NULL | Номер телефона пользователя |
| User\_Sex | varchar (2) | NOT NULL, CHECK('М' or 'Ж') | Пол пользователя |
| User\_BirthYear | date | NOT NULL | Дата рождения пользователя |
| User\_Adress | varchar(100) | NOT NULL | Адрес пользователя |

UserInf содержит столбцы для хранения информации о пользователе. Используется при регистрации пользователя.

Таблица 2.10 – UserAccount

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | constrains | Значение поля |
| ID\_Account | int | PK | Первичный ключ |
| ID\_UserAc | int | FK | Внешний ключ к UserInf(ID\_User) |
| Login | varchar(40) | NOT NULL | Логин пользователя |
| Password | varchar(40) | NOT NULL | Пароль пользователя |

Таблица UserAccount содержит данные об аккаунте зарегистрированного пользователя. Данная таблица будет использоваться при попытке пользователя войти в аккаунт.

Листинг SQL-кода для создания таблиц находится в приложении А.

# 3.2. Пользователи.

Пользователь базы данных – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации.

Для двух пользователей было создано общее постоянное и временное табличное пространство, представленное на рисунке 3.2.1.

|  |
| --- |
|  |

Рисунок 3.2.1 – табличные пространства для пользователей

Далее был создан профиль безопасности для пользователей (Рисунок 3.2.2).

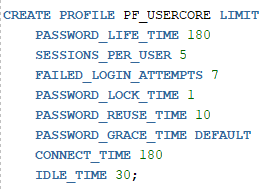


Рисунок 3.2.2. — профиль безопасности для пользователей

Во время проектировании базы данных было создано и использовано 2 пользователя. Первый – является системным администратором, имеет доступ для чтения и изменения таблиц, связанных с книгами, заказами, пользователями. Создание его роли и пользователя представлено на рисунке 3.2.3.

|  |
| --- |
|  |

Рисунок 3.2.3 — создание роли и пользователя с заданием привилегий

Вторая роль – пользователь, которой имеет возможность зарегистрироваться, авторизоваться, получить доступ к информации о книгах, заказах. (Рисунок 3.2.4.)

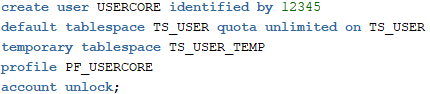


Рисунок 3.2.4 – роль для администратора и пользователь бд для него

# 3.3. Хранимые процедуры

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. При создании хранимой процедуры можно определить необязательный список параметров. Для ограничения пользователя от ошибок в хранимых процедурах присутствует обработка исключений.

При разработке курсового проекта были созданы следующие процедуры:

* bookinf- выборка данных из таблицы Book
* enterUser- добавление пользователя;
* SEARCH\_BOOKS\_ByPriceRange– фильтрация книг по цене;
* SEARCH\_BOOKS\_ByPageRange– фильтрация книг по количеству страниц;
* SEARCH\_BOOKS\_ByYear\_Of\_Publication – фильтрация книг по году издания;
* Get\_Book\_ByGenre – фильтрация книг по жанру;
* Get\_Book\_ByAuthor – фильтрация книг по автору;
* Get\_Book\_ByLanguage– фильтрация книг по языку;
* CheckBookAvailabilityCURS– проверка наличия определенной книги;
* InsertOrderInf– заказ книг;
* GetUsersAccount– получить информацию о юзере;
* ADD\_NEW\_BOOK– добавить новую книгу;
* ADD\_AUTHORS– добавить новых авторов;
* ADD\_GENRES– добавление новые жанры;
* getUsersInf– вывод всей инфы о клиентах и их аккаунтах;
* getOrdersInf– вывод всей инфы о заказах;

Листинг всех процедур находится в приложении Б.

# 3.4. Индексы.

При работе с большим количеством строк в таблицах базы данных, которые хранятся в произвольном порядке, поиск данных по заданному критерию может занимать значительное количество времени, если происходит последовательный просмотр таблицы строка за строкой. Для этого создаются индексы, индекс является объектом базы данных, который создаётся с целью повышения скорости поиска данных.

Для ускорения поиска были созданы индексы столбцов ID\_Author, GenresCode, Book\_ID\_PublishingHouse, ID\_Orders. Пример кода можно увидеть на рисунке 3.4.1.

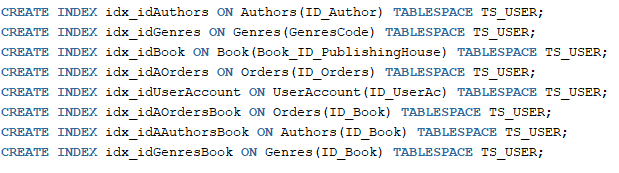


Рисунок 3.4.1 – Код индексов для ускорения поиска в бд;

Таким образом, процесс разработки индексов прошел успешно, а время поиска для таблиц Authors, Genres, Book, UserAccount было ускорено.

# 3.5. Последовательности и триггеры.

Для автоматизации ввода идентификаторов в таблицы были использованы последовательности и триггеры. Пример последовательности и триггера для вставки идентификаторов в таблицу “Author”.

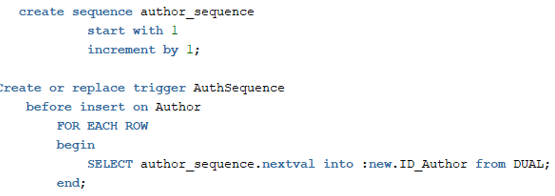


Рисунок 3.5.1 – Код последовательности и триггера;

# 4.Описание процедур импорта и экспорта данных

В данном курсовом проекте реализованы процедуры экспорта и импорта данных из XML файла в базу данных таблицы Author и наоборот.

# 4.1 Описание процедуры импорта данных.

PL/SQL инструкция Xmltype позволяет импортировать данные из файла непосредственно в таблицу базы данных или несекционированное представление. Для передачи исходного текста XML в конструктор XMLType можно использовать значения типов CLOB, BLOB, VARCHAR2, BFILE.

Если XML-файл находится на стороне клиента и является достаточно большим для передачи в виде строки в запросе, необходимо рассмотреть возможность доставки файла с содержимым XML на файловую систему сервера.

Код процедуры импорта данных в формат XML представлен на рисунке 4.1.



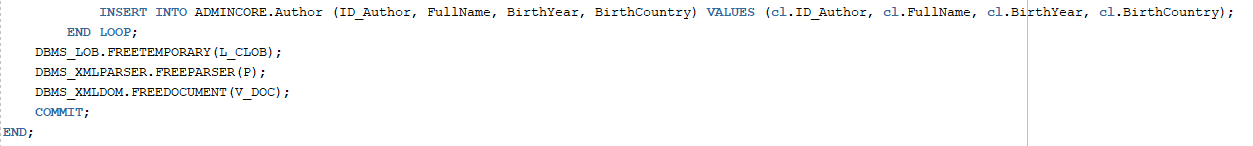


Рисунок 4.1 – Процедура импорта данных из xml в таблицу Author

# Описание процедуры экспорта данных

Для осуществления преобразования применяется инструкция xmldom.

Объектная модель документа (DOM) — это API для документов HTML и XML, которая определяет логическую структуру документов и предоставляет доступ к ним и возможность их манипуляции.

В спецификации DOM термин «документ» используется в широком смысле. XML применяется для представления различных видов информации, которая может храниться в различных системах. Эта информация обычно относится к «данным», однако XML представляет эти данные в виде документов, и пакет DBMS\_XMLDOM позволяет получать доступ как к документам, основанным на схеме, так и к документам, не основанным на схеме.

Код процедуры экспорта представлен на рисунке 4.2.



Рисунок 4.2 – Процедура экспорта

При вызове вышеуказанных процедур, данные из таблицы Author успешно импортируются в xml-файл и экспортируются обратно.

# 5. Описание технологии и ее применения в базе данных

В процессе разработки проекта была использована технология: резервное копирование и восстановление данных.

Стандартный пользовательский метод резервного копирования состоит в применении команд операционной системы для копирования необходимых файлов в другое место и/или на ленточное устройство.

В процессе разработки проекта резервное копирование и восстановление данных происходило при помощи утилиты RMAN. Резервное копирование файлов базы данных Oracle выполняться внутри базы данных посредством самого сервера баз данных. RMAN умеет делать резервные копии и копии образов файлов данных, управляющих файлов, архивных журналов повторного выполнения, файлов SPFILE и фрагментов резервных копий RMAN.

Возможности RMAN включают следующее:

* выполнение полного резервирования и резервирования изменений;
* выполнение холодного/горячего резервирования;
* обнаружение поврежденных блоков;
* параллельное выполнения операций ввода/вывода;
* автоматическое протоколирование операций копирования и восстановления.

С помощью RMAN можно выполнять инкрементное резервное копирование. Размер резервных копий в таком случае зависит не от размера базы данных, а скорее от уровня активности внутри нее, поскольку во время инкрементного резервного копирования не измененные блоки пропускаются.

Подключаться к RMAN можно путем ввода в приглашении операционной системы команды rman. Если при подключении будет указано что наша БД не включена, то ее будет необходимо включить.

После ввода команды rman будет появляться приглашение RMAN>, позволяющее вводить различные команды RMAN. (Рисунок 5.1)

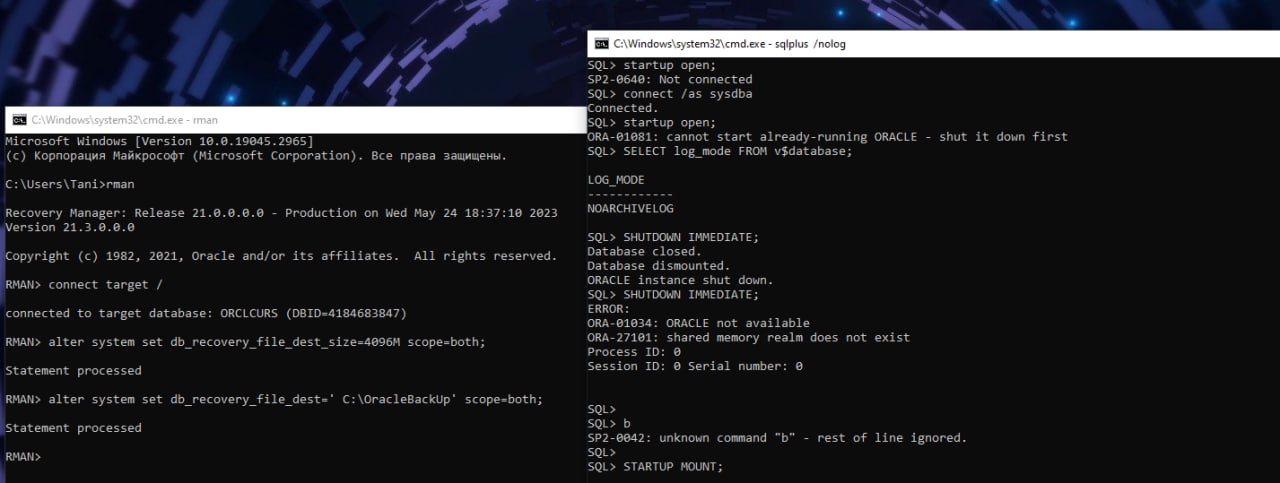


Рисунок 5.1 – Подключение к rman;

В курсовом проекте был использован метод холодного резервирования. Режим архивирования журналов выключен, СУБД выполняется в режиме NOMOUNT. Недостаток данного метода: из «холодной» резервной копии можно восстановить только то состояние базы данных, которое было в момент останова; транзакции, сделанные после рестарта базы, в «холодную» резервную копию не попадут.

Для резервирования файлов данных используется команда: BACKUP DATABASE TAG "FULL\_DATABASE\_DATAFILES". (Рисунок 5.2.)

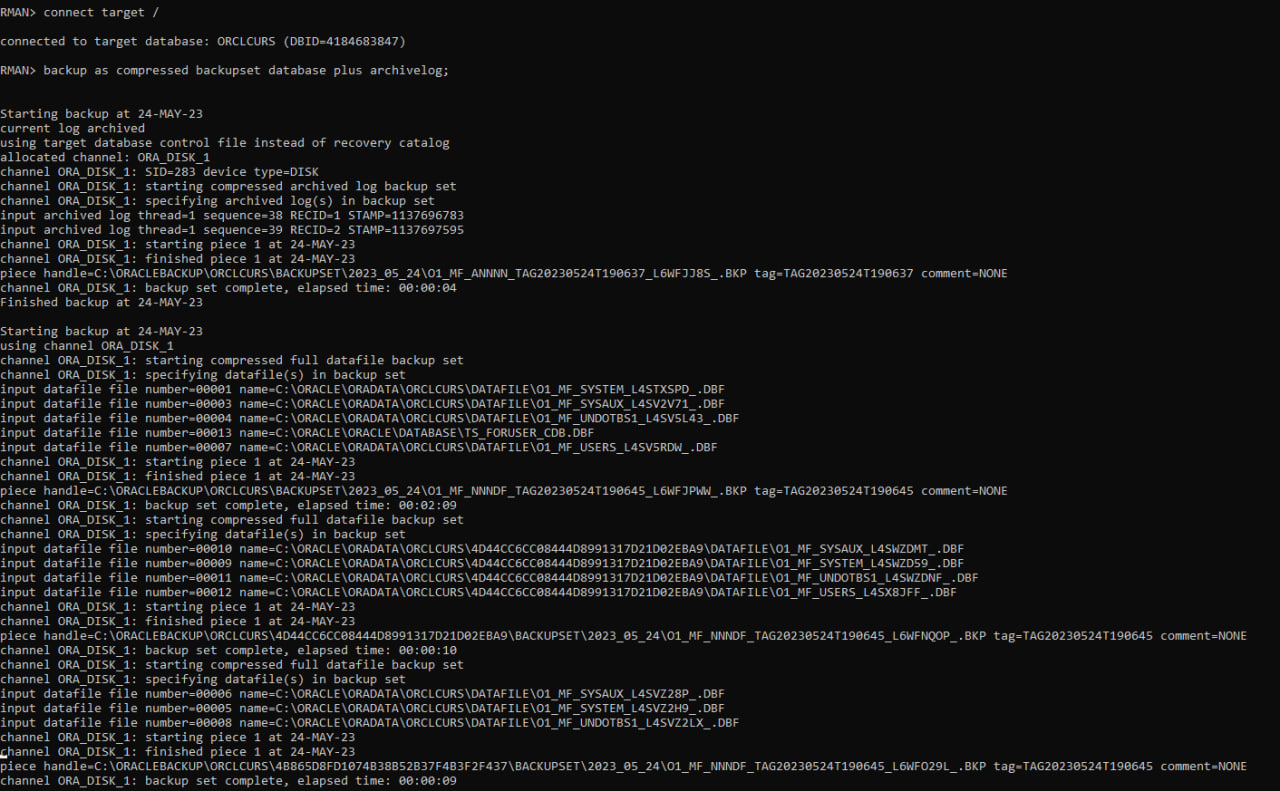


Рисунок 5.2 – Резервирование файлов данных

Для резервирования контрольных файлов используется команда: BACKUP CURRENT CONTROLFILE TAG "FULL\_DATABASE\_CONTROLFILE". Список файлов резервных копий на рисунке 5.3.

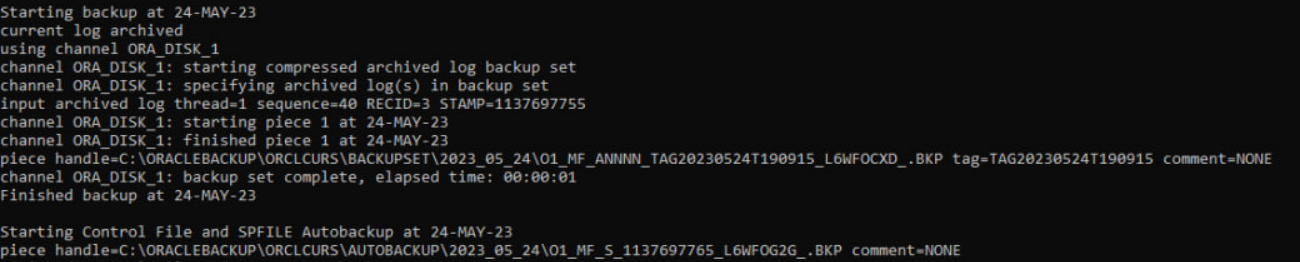


Рисунок 5.3 – список файлов резервных копий

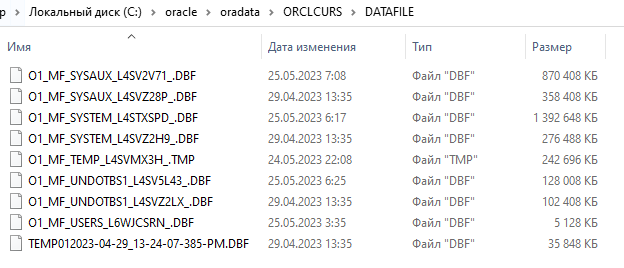
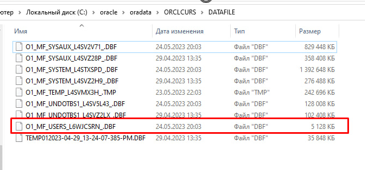
 

Рисунок 5.4 – 1. Оригинальные файлы, 2. Восстановленные файлы;

Чтобы получить информацию о созданных бэкапах нужно выполнить команду: RMAN> LIST BACKUP;

Восстановление данных. Для восстановления данных целевая БД должна находиться в состоянии NOMOUNT/ MOUNT/ OPEN в зависимости от характера восстановления, например:

* NOMOUNT: для восстановления контрольных файлов БД (фактически – СУБД)
* MOUNT: для восстановления БД целиком или табличного пространства SYSTEM
* OPEN: для восстановление табличных пространств, помимо SYSTEM (в этом случае перед процедурой восстановления само табличное пространство потребуется перевести в состояние OFFLINE).

Для восстановления удаленной таблицы мы будем использовать Data Pump.

Технология Data Pump, которая впервые появилась в Oracle Database 10g, представляет собой работающую на стороне сервера инфраструктуру для быстрого перемещения данных между базами Oracle. Она позволяет администраторам баз данных переносить большие объемы данных и метаданных на очень большой скорости по сравнению с прежней технологией для экспорта и импорта. Для достижения максимальной пропускной способности она осуществляет управление множеством параллельных потоков данных.

После восстановления удаленных файлов, нам необходимо закрыть подключение в sqlplus, так как подключения будут конфликтовать друг между другом. Далее, нам необходимо создать директорию, в которой будут находиться наши таблицы. На изображении ниже мы помещаем состояние нашей таблицы в директорию (рисунок 5.5)

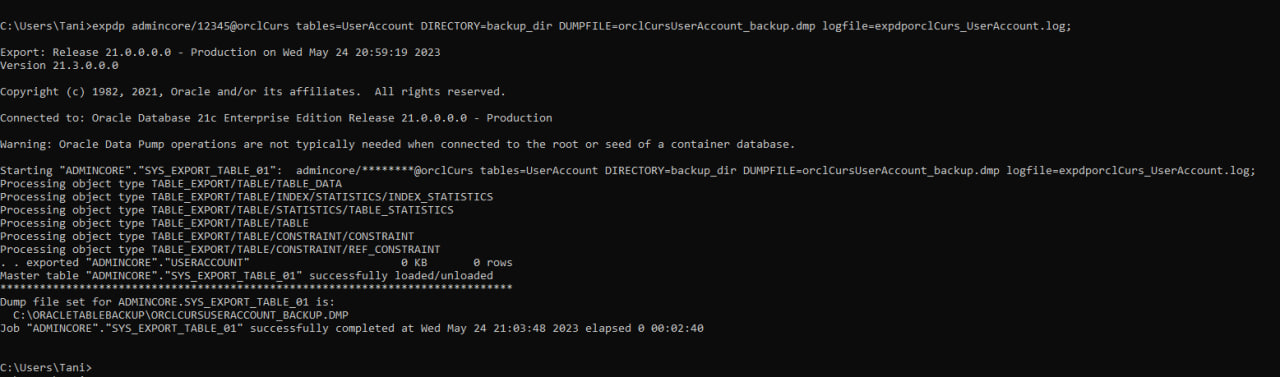


Рисунок 5.5 – сохранение текущего состояния

Получаем данные, с помощью которых мы сможем восстановить удаленную таблицу (Рисунок 5.6)

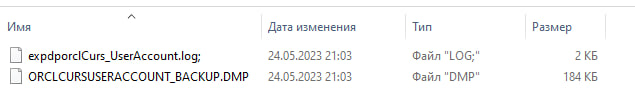


Рисунок 5.6 – файлы для восстановления

Удаляем нашу таблицу (рисунок 5.7)

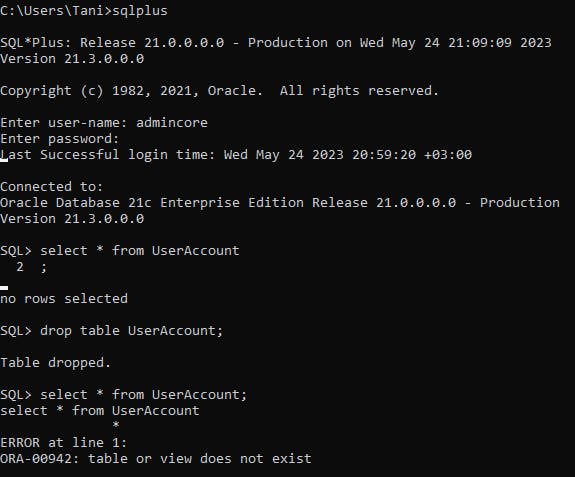


Рисунок 5.7 – удаление таблицы

А теперь восстанавливаем таблицу (рисунок 5.8)



Рисунок 5.8 – восстановление таблицы

Результат представлен на рисунке 5.9



Рисунок 5.9 – восстановлена таблица

Как можно увидеть таблица успешно восстановилась.

# 6. Тестирование производительности

Одним из требованием к разработке БД была проверка производительности. Для неё необходимо заполнить БД большим количеством различных данных и узнать время выполнения одного запроса.

Для данной задачи мы создали процедуру, которая будет выполнять вставку строк в специально созданную таблицу. Разработанный анонимный блок позволяет добавить большое количество строк за одно выполнение (рисунок 6.1).

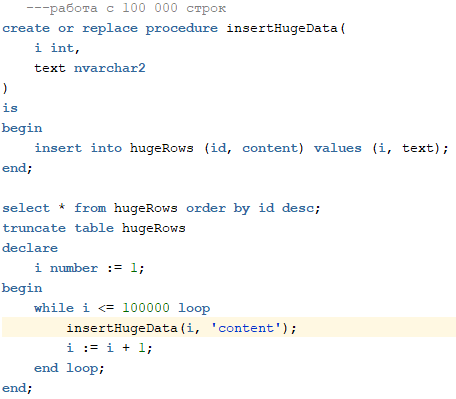




Рисунок 6.1 – Процедура и анонимный блок для заполнения таблицы большим количеством данных;

Далее протестируем стоимость выполнения запроса к нескольким таблицам с использованием конструкции join. Запрос со стоимостью представлен на рисунке 6.2.

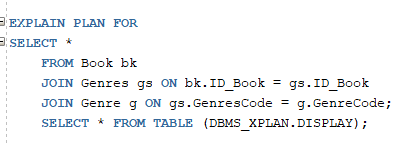
 

Рисунок 6.2 – запрос к таблицам Book, Genres, Genre

Далее создадим индекс на внешние ключи, которые используются в данном запросе, и повторно проверим стоимость запроса (рисунок 6.4).



Рисунок 6.4 – стоимость запроса после создания индексов

Как видим, стоимость запроса уменьшилась. Однако, даже без индексов база данных успешно прошла тестирование на производительность.

# Руководство пользователя

Для частичной работы с базой данных было разработано приложение на WPF. Сперва было создано подключение к базе данных. После чего, с помощью Entity Framework была сформирована edm модель для подключения, которое и служило доступом к объектам базы данных.

По открытию приложения, пользователя ожидает каталог от аккаунта гостя (рисунок 7.1).

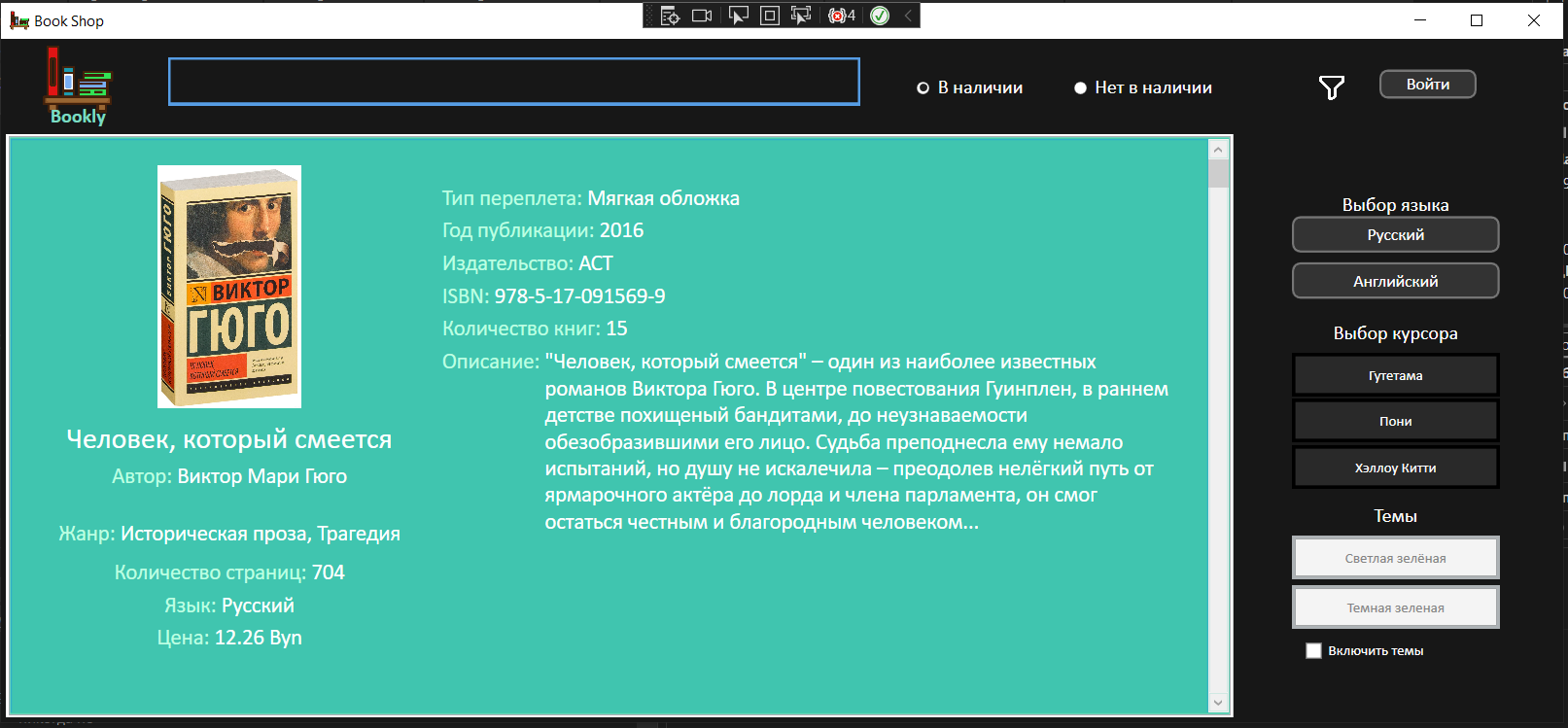


Рисунок 7.1 – каталог гостя;

Гость может делать поиск по названию книги (рисунки 7.2 и 7.3).

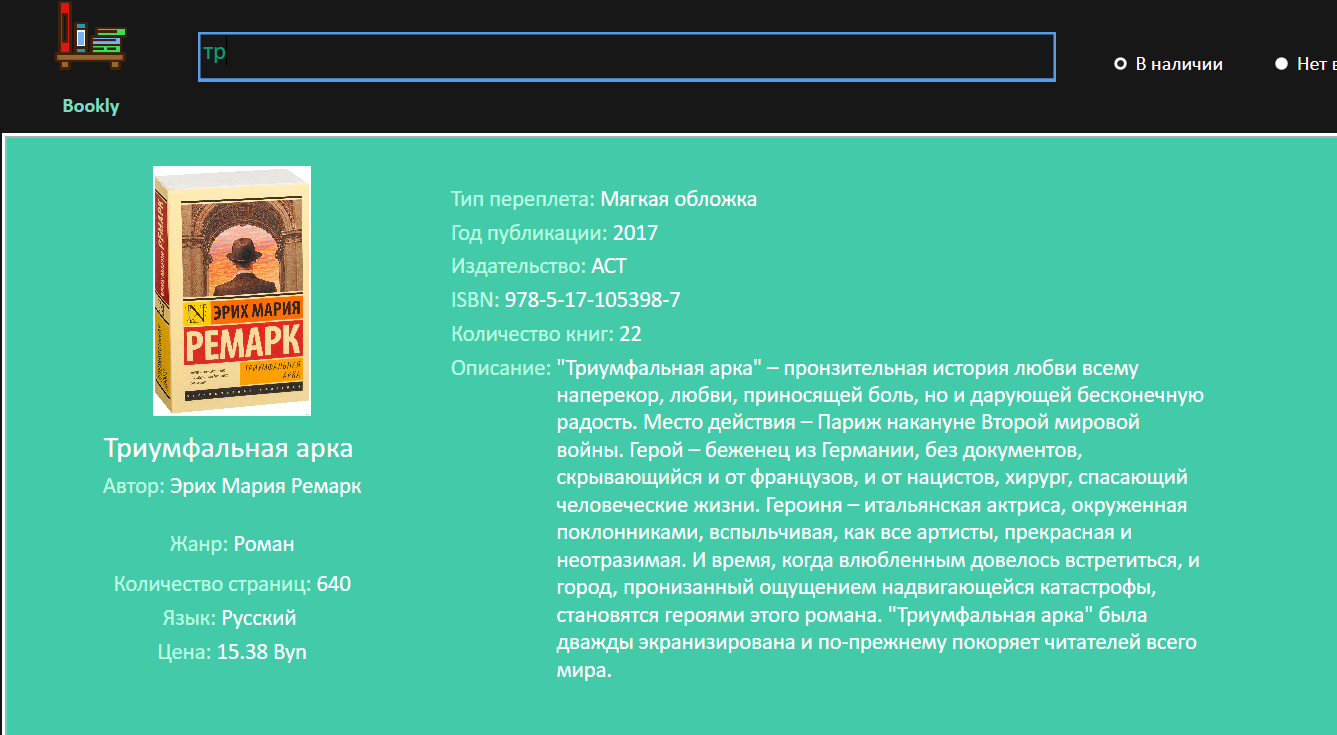


Рисунок 7.2 – поиск книг на букву “Тр”;

Если у пользователя не было аккаунта, он может зарегистрироваться(рисунок 7.4).

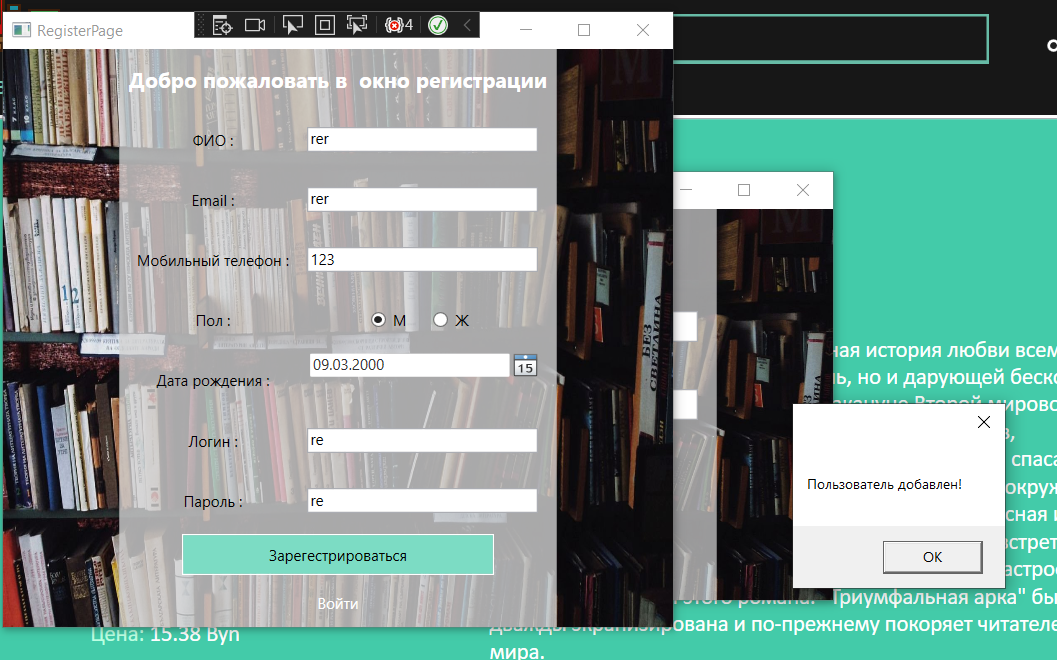


Рисунок 7.3– форма регистрации;

Если аккаунт существует, то можно войти в аккаут. На примере вход в аккаунт админа (рисунок 7.4).

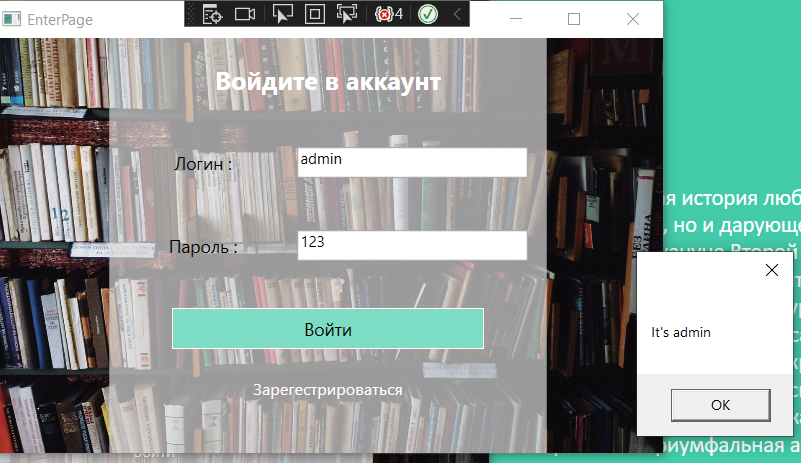


Рисунок 7.4 – форма авторизации

После входа в аккаунт админа появляется интерфейс с отображением таблицы с книгами (рисунок 7.5)

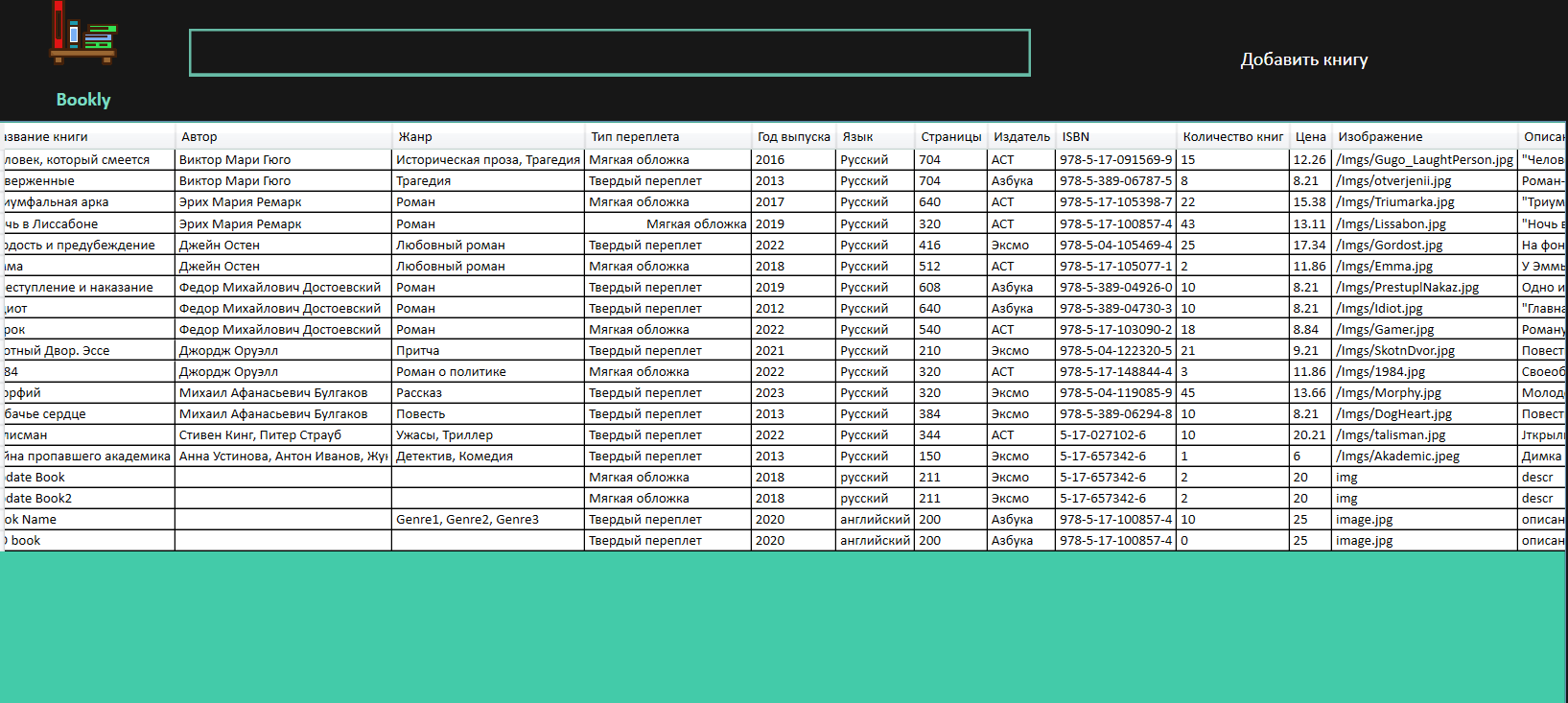


Рисунок 7.5 – интерфейс админа

Для выполнения оставшегося функционала необходимо перейти в sqldeveloper и выполнять необходимые процедуры для фильтрации, поиска, добавлении обновлении книг, добавлении пользователей.

Исходя из проделанной работы, мы можем отметить, что разработанное приложение выполняет свои функции: регистрация, авторизация клиента, вывод необходимой информации о книгах, отображение в аккаунте админа данных.

# Заключение

В ходе выполнения поставленных задач была достигнута основная цель - создание базы данных для программного средства "Книжный магазин". Для реализации проекта использовалась СУБД Oracle DataBase 21c, а разработка desktop-приложения осуществлялась с использованием языка C# и технологии WPF для построения клиентских приложений Windows. В процессе курсового проекта были задействованы различные объекты, включая таблицы, хранимые процедуры, индексы, последовательности и триггеры.

В ходе разработки процедур были выполнены следующие задачи, состоящие из:

* реализации системы авторизации и регистрации пользователей;
* возможность добавления и обновления информации о книгах;
* получение информации о книгах;
* поиск и фильтрация книг по различным параметрам;
* возможность оформления заказа;
* проверка наличия книг;

Благодаря последовательностям и триггерам был автоматизирован и упрощен ввод уникальных идентификаторов.

Приложение прошло тестирование на большом объеме данных в БД. Также были разработаны процедуры для импорта и экспорта данных в формате XML. Была внедрена технология "Резервное копирование и восстановление". На основании результатов работы программы можно сделать вывод, что разработанное приложение функционирует корректно, а технические требования были выполнены.

# Cписок источников

1. Язык C# и .NET Framework [Электронный ресурс] / Режим доступа – URL: http://professorweb.ru/my/csharp/charp\_theory/level1/infonet.php – Дата доступа: 18.03.2023.
2. Руководство по ADO.NET и работе с базами данных [Электронный ресурс] / Режим доступа – URL: – https://metanit.com/sharp/adonet/. – Дата доступа: 22.03.2023.
3. Документация Oracle [Электронный ресурс] / Режим доступа – URL: https://docs.oracle.com/cd/B28359\_01/server.111/b31222/toc.htm – Дата доступа: 16.03.2023.
4. Официальный сайт Oracle [Электронный ресурс] / Режим доступа – URL: <https://www.oracle.com/database/database-vault/index.html> – Дата доступа: 17.03.2023.

5 Продукты Oracle [Электронный ресурс] / Режим доступа – URL: http://www.interface.ru/home.asp?artId=24678 – Дата доступа: 17.03.2023.

ПРИЛОЖЕНИЕ А

Создание базы данных

|  |
| --- |
| -- Создание таблицы Author  CREATE TABLE Author (  ID\_Author int PRIMARY KEY,  FullName VARCHAR(60)NOT NULL,  BirthYear number,  BirthCountry VARCHAR(30) NOT NULL  ) TABLESPACE TS\_USER;  -- Создание таблицы Genre  CREATE TABLE Genre (  GenreCode int PRIMARY KEY,  NameGenre varchar(60)NOT NULL  ) TABLESPACE TS\_USER;  -- Создание таблицы PublisherHouse  CREATE TABLE PublisherHouse (  ID\_PublishingHouse int PRIMARY KEY,  PublishingHouse\_Name VARCHAR(30) NOT NULL  ) TABLESPACE TS\_USER;  -- Создание таблицы UserInf  CREATE TABLE UserInf(  ID\_User int PRIMARY KEY,  User\_Name VARCHAR(100) NOT NULL,  User\_Email VARCHAR(40) NOT NULL,  User\_Numer VARCHAR(10) NOT NULL,  User\_Sex VARCHAR(2) NOT NULL ,  User\_BirthYear date NOT NULL,  CHECK (User\_Sex = 'М' or User\_Sex = 'Ж')  ) TABLESPACE TS\_USER;  -- Создание таблицы UserAccount  CREATE TABLE UserAccount (  ID\_Account INT PRIMARY KEY,  ID\_UserAc INT ,  Login VARCHAR(40) NOT NULL, ---upd  Password VARCHAR(40) NOT NULL,  FOREIGN KEY (ID\_UserAc) REFERENCES UserInf(ID\_User)  ) TABLESPACE TS\_USER;  -- Создание таблицы Order  CREATE TABLE OrderInf (  ID\_Order int PRIMARY KEY,  ID\_User int,  Order\_Date date NOT NULL,  Delivery\_Time date NOT NULL,  Amount number(5,2) NOT NULL,  FOREIGN KEY (ID\_User) REFERENCES UserInf(ID\_User)  ) TABLESPACE TS\_USER;  -- Создание таблицы Book  CREATE TABLE Book (  ID\_Book int PRIMARY KEY,  Book\_Name varchar(200) NOT NULL,  Page\_Number number NOT NULL,  ISBN varchar(20) NOT NULL,  BookBindingType varchar(40) NOT NULL,  Year\_Of\_Publication number NOT NULL,  Book\_ID\_PublishingHouse int,  Book\_Language varchar(30) NOT NULL,  Book\_Description varchar(3000) NOT NULL,  Book\_ImageSource varchar(40) NOT NULL,  NumbersOfBooks number,  Book\_Prise number(5,2) NOT NULL,  FOREIGN KEY (Book\_ID\_PublishingHouse) REFERENCES PublisherHouse (ID\_PublishingHouse)  ) TABLESPACE TS\_USER;  -- Создание таблицы Authorss  CREATE TABLE Authors(  ID\_Author int,  ID\_Book int,  FOREIGN KEY (ID\_Author) REFERENCES Author(ID\_Author),  FOREIGN KEY (ID\_Book) REFERENCES Book(ID\_Book)  ) TABLESPACE TS\_USER;  -- Создание таблицы Genres  CREATE TABLE Genres (  GenresCode int,  ID\_Book int,  FOREIGN KEY (ID\_Book) REFERENCES Book(ID\_Book),  FOREIGN KEY (GenresCode) REFERENCES Genre(GenreCode)  ) TABLESPACE TS\_USER;  -- Создание таблицы Orders  CREATE TABLE Orders (  ID\_Orders int,  ID\_Book int,  NumberOrderedBooks number NOT NULL,  FOREIGN KEY (ID\_Orders) REFERENCES OrderInf(ID\_Order),  FOREIGN KEY (ID\_Book) REFERENCES Book(ID\_Book)  ) TABLESPACE TS\_USER;  create table hugeRows(  id int primary key,  content nvarchar2(10)  );  CREATE INDEX idx\_idAuthors ON Authors(ID\_Author) TABLESPACE TS\_USER;  CREATE INDEX idx\_idGenres ON Genres(GenresCode) TABLESPACE TS\_USER;  CREATE INDEX idx\_idBook ON Book(Book\_ID\_PublishingHouse) TABLESPACE TS\_USER;  CREATE INDEX idx\_idAOrders ON Orders(ID\_Orders) TABLESPACE TS\_USER;  CREATE INDEX idx\_idUserAccount ON UserAccount(ID\_UserAc) TABLESPACE TS\_USER;  CREATE INDEX idx\_idAOrdersBook ON Orders(ID\_Book) TABLESPACE TS\_USER;  CREATE INDEX idx\_idAAuthorsBook ON Authors(ID\_Book) TABLESPACE TS\_USER;  CREATE INDEX idx\_idGenresBook ON Genres(ID\_Book) TABLESPACE TS\_USER; |

ПРИЛОЖЕНИЕ Б

Создание процедур

|  |
| --- |
| --1) Добавить нового пользователя в "регистрации"  drop procedure enterUser  CREATE OR REPLACE PROCEDURE enterUser(  enteredLogin IN UserAccount.Login%TYPE,  enteredPassword IN UserAccount.Password%TYPE,  enterendFIO IN UserInf.User\_Name%TYPE,  enteredEmail IN UserInf.User\_Email%TYPE,  enteredNumber IN UserInf.User\_Numer%TYPE,  enteredSex IN UserInf.User\_Sex%TYPE,  enteredBirthday IN UserInf.User\_BirthYear%TYPE  )  IS  BEGIN  INSERT INTO UserInf(ID\_User, User\_Name, User\_Email, User\_Numer, User\_Sex, User\_BirthYear)  VALUES (NULL, enterendFIO, enteredEmail, enteredNumber, enteredSex, To\_Date(enteredBirthday, 'DD.MM.YYYY'));  INSERT INTO UserAccount(Id\_Account,Id\_UserAc, login, password)  VALUES (NULL, NULL, enteredLogin, enteredPassword);  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Пользователь добавлен!');  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Возникла ошибка при добавлении пользователей' || SQLERRM);  END;  --test01  exec enterUser('12','11','TESTFIO2', 'TESTEmail2','numb','Ж','27.10.1983');  select \* from UserInf  ORDER BY ID\_User Asc;  select \* from UserAccount  ORDER BY ID\_Account Asc;  --2)УПРАВЛЕНИЕ ИНФОРМАЦИЕЙ О КНИГАХ  --информации о книге (авторе+ жанре+ количесве страниц+ ISBN+ тип переплета+ год публикации+ издательсво+ язык+ описание)  create or replace procedure bookinf (  books out sys\_refcursor  )  IS  BEGIN  OPEN books FOR  SELECT ID\_BOOK, Book\_Name, Page\_Number, ISBN, BookBindingType, Year\_Of\_Publication ,  Book\_Language, Book\_ImageSource,NumbersOfBooks, Book\_Prise, Book\_Description  FROM Book  Order By ID\_BOOK asc;  END;  --test02  VAR bookCursor REFCURSOR;  EXEC bookinf(:bookCursor);  PRINT bookCursor;  select \* from book order by ID\_BOOK;  ----В наличии  create or replace procedure bookinfAvailability (  books out sys\_refcursor  )  IS  BEGIN  OPEN books FOR  SELECT ID\_BOOK, Book\_Name, Page\_Number, ISBN, BookBindingType, Year\_Of\_Publication ,  Book\_Language, Book\_ImageSource,NumbersOfBooks, Book\_Prise, Book\_Description  FROM Book  WHERE NumbersOfBooks>0  Order By ID\_BOOK asc;  END;  VAR bookCursor REFCURSOR;  EXEC bookinfAvailability(:bookCursor);  PRINT bookCursor;  ----Нет в наличии  create or replace procedure bookinf\_NON\_Availability (  books out sys\_refcursor  )  IS  BEGIN  OPEN books FOR  SELECT ID\_BOOK, Book\_Name, Page\_Number, ISBN, BookBindingType, Year\_Of\_Publication ,  Book\_Language, Book\_ImageSource,NumbersOfBooks, Book\_Prise, Book\_Description  FROM Book  WHERE NumbersOfBooks=0  Order By ID\_BOOK asc;  END;  VAR bookCursor REFCURSOR;  EXEC bookinf\_NON\_Availability(:bookCursor);  PRINT bookCursor;  ---Вся информация о жанрах  create or replace function GetALLGenresInf(func\_Book\_ID IN Book.ID\_Book%TYPE)  RETURN SYS\_REFCURSOR AS  gInf SYS\_REFCURSOR;  BEGIN  OPEN gInf FOR  SELECT g.NameGenre  FROM Book  Join Genres gs on Book.ID\_Book = gs.ID\_Book  Join Genre g on gs.GenresCode=g.GenreCode  WHERE Book.ID\_Book = func\_Book\_ID;  RETURN gInf;  END;  ---Вся информация об авторах  create or replace function GetALLAuthorsInf(func\_Book\_ID IN Book.ID\_Book%TYPE)  RETURN SYS\_REFCURSOR AS  gInf SYS\_REFCURSOR;  BEGIN  OPEN gInf FOR  SELECT au.FullName  FROM Book  Join Authors a on Book.ID\_Book = a.ID\_Book  Join Author au on au.ID\_Author = a.ID\_Author  WHERE Book.ID\_Book = func\_Book\_ID;  RETURN gInf;  END;    ---Вся информация об идателе  create or replace function GetPublisherInf(func\_Book\_ID IN Book.ID\_Book%TYPE)  RETURN SYS\_REFCURSOR AS  gInf SYS\_REFCURSOR;  BEGIN  OPEN gInf FOR  SELECT PH.PublishingHouse\_Name  FROM Book  Join PublisherHouse PH on Book.Book\_ID\_PublishingHouse = PH.ID\_PublishingHouse  WHERE Book.ID\_Book = func\_Book\_ID;  RETURN gInf;  END;  -- Select GetPublisherInf(1) from dual;    --ПОИСК по НАЗВАНИЮ  CREATE OR REPLACE FUNCTION find\_books\_by\_title(title IN VARCHAR2)  RETURN SYS\_REFCURSOR IS  books\_cursor SYS\_REFCURSOR;  BEGIN  OPEN books\_cursor FOR  SELECT Book\_Name, Year\_Of\_Publication, Book\_Prise  FROM BOOK  WHERE Book\_Name = title;    RETURN books\_cursor;  END;    --ФИЛЬТРАЦИЯ по ЦЕНЕ    CREATE OR REPLACE PROCEDURE SEARCH\_BOOKS\_ByPriceRange(  prc\_min\_price IN NUMBER,  prc\_max\_price IN NUMBER  ) IS  v\_books\_cursor SYS\_REFCURSOR;  v\_book\_Name Book.Book\_Name%TYPE;  v\_book\_NumbersOfBooks Book.NumbersOfBooks%TYPE;  v\_book\_price Book.Book\_Prise%TYPE;  BEGIN  IF prc\_min\_price > prc\_max\_price THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Минимальная цена не может быть больше максимальной');  END IF;  OPEN v\_books\_cursor FOR  SELECT Book\_Name, NumbersOfBooks, Book\_Prise  FROM Book  WHERE Book\_Prise BETWEEN prc\_min\_price AND prc\_max\_price;  LOOP  FETCH v\_books\_cursor INTO v\_book\_Name, v\_book\_NumbersOfBooks, v\_book\_price;  EXIT WHEN v\_books\_cursor%NOTFOUND;  IF v\_book\_NumbersOfBooks > 0 THEN  DBMS\_OUTPUT.PUT\_LINE('Название: ' || v\_book\_Name || ', Количество книг: ' || v\_book\_NumbersOfBooks || ', Цена: ' || v\_book\_price);  ELSE  DBMS\_OUTPUT.PUT\_LINE('Название: ' || v\_book\_Name || ', Количество книг: Нет в наличии' || ', Цена: ' || v\_book\_price);  END IF;  END LOOP;  CLOSE v\_books\_cursor;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Произошла ошибка: ' || SQLERRM);  END;  EXEC SEARCH\_BOOKS\_ByPriceRange(prc\_min\_price => 11, prc\_max\_price => 50);  --ФИЛЬТРАЦИЯ по количеству страниц  CREATE OR REPLACE PROCEDURE SEARCH\_BOOKS\_ByPageRange(  prc\_min\_pages IN NUMBER,  prc\_max\_pages IN NUMBER  ) IS  v\_books\_cursor SYS\_REFCURSOR;  v\_book\_Name Book.Book\_Name%TYPE;  v\_book\_NumbersOfBooks Book.NumbersOfBooks%TYPE;  v\_book\_pages Book.Page\_Number%TYPE;  BEGIN  IF prc\_min\_pages > prc\_max\_pages THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Минимальное количество страниц не может быть больше максимального');  END IF;  OPEN v\_books\_cursor FOR  SELECT Book\_Name, NumbersOfBooks, Page\_Number  FROM Book  WHERE Page\_Number BETWEEN prc\_min\_pages AND prc\_max\_pages;  LOOP  FETCH v\_books\_cursor INTO v\_book\_Name, v\_book\_NumbersOfBooks, v\_book\_pages;  EXIT WHEN v\_books\_cursor%NOTFOUND;  IF v\_book\_NumbersOfBooks > 0 THEN  DBMS\_OUTPUT.PUT\_LINE('Название: ' || v\_book\_Name || ', Количество книг: ' || v\_book\_NumbersOfBooks || ', Количество страниц: ' || v\_book\_pages);  ELSE  DBMS\_OUTPUT.PUT\_LINE('Название: ' || v\_book\_Name || ', Количество книг: Нет в наличии' || ', Количество страниц: ' || v\_book\_pages);  END IF;  END LOOP;  CLOSE v\_books\_cursor;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Произошла ошибка: ' || SQLERRM);  END;    EXEC SEARCH\_BOOKS\_ByPageRange(prc\_min\_pages => 200, prc\_max\_pages => 300);  --ФИЛЬТРАЦИЯ по ГОДУ ИЗДАНИЯ  CREATE OR REPLACE PROCEDURE SEARCH\_BOOKS\_ByYear\_Of\_Publication(  prc\_min\_year IN NUMBER,  prc\_max\_year IN NUMBER  ) IS  v\_books\_cursor SYS\_REFCURSOR;  v\_book\_Name Book.Book\_Name%TYPE;  v\_book\_NumbersOfBooks Book.NumbersOfBooks%TYPE;  v\_book\_Year\_Of\_Publication Book.Year\_Of\_Publication%TYPE;  BEGIN  IF prc\_min\_year > prc\_max\_year THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Минимальный год выпуска не может быть больше максимального');  END IF;  OPEN v\_books\_cursor FOR  SELECT Book\_Name, NumbersOfBooks, Year\_Of\_Publication  FROM Book  WHERE Year\_Of\_Publication BETWEEN prc\_min\_year AND prc\_max\_year;  LOOP  FETCH v\_books\_cursor INTO v\_book\_Name, v\_book\_NumbersOfBooks, v\_book\_Year\_Of\_Publication;  EXIT WHEN v\_books\_cursor%NOTFOUND;  IF v\_book\_NumbersOfBooks > 0 THEN  DBMS\_OUTPUT.PUT\_LINE('Название: ' || v\_book\_Name || ', Количество книг: ' || v\_book\_NumbersOfBooks || ', Год выпуска: ' || v\_book\_Year\_Of\_Publication);  ELSE  DBMS\_OUTPUT.PUT\_LINE('Название: ' || v\_book\_Name || ', Количество книг: Нет в наличии' || ', Год выпуска: ' || v\_book\_Year\_Of\_Publication);  END IF;  END LOOP;  CLOSE v\_books\_cursor;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Произошла ошибка: ' || SQLERRM);  END;      EXEC SEARCH\_BOOKS\_ByYear\_Of\_Publication(prc\_min\_year =>2016, prc\_max\_year => 2020);    --ФИЛЬТРАЦИЯ по ЖАНРУ  CREATE OR REPLACE PROCEDURE Get\_Book\_ByGenre(  p\_NameGenree IN Genre.NameGenre%TYPE  ) AS  prc\_books\_cursor SYS\_REFCURSOR;  getBookName Book.Book\_Name%TYPE ;  getBookPages Book.Page\_Number%TYPE;  getBookISBN Book.ISBN%TYPE ;  getBookBinding Book.BookBindingType%TYPE;  getBookYearPublic Book.Year\_Of\_Publication%TYPE;  getBookIDPublishHous Book.Book\_ID\_PublishingHouse%TYPE;  getBookLanguage Book.Book\_Language%TYPE;  getBookDescript Book.Book\_Description%TYPE;  getBookImageSoursce Book.Book\_ImageSource%TYPE;  getBookNumbersOfBooks Book.NumbersOfBooks%TYPE ;  getBookPrice Book.Book\_Prise%TYPE ;  prc\_genre\_count NUMBER; -- Declare the variable for genre count  BEGIN  -- Проверяем, существует ли указанный жанр  SELECT COUNT(\*)  INTO prc\_genre\_count  FROM Genre  WHERE NameGenre = p\_NameGenree;    IF prc\_genre\_count = 0 THEN  -- Если жанр не найден, генерируем исключение  RAISE\_APPLICATION\_ERROR(-20001, 'Жанр ' || p\_NameGenree || ' не найден.');  END IF;    OPEN prc\_books\_cursor FOR  SELECT b.Book\_Name, b.Page\_Number, b.ISBN, b.BookBindingType, b.Year\_Of\_Publication, b.Book\_ID\_PublishingHouse,  b.Book\_Language, b.Book\_Description, b.Book\_ImageSource, b.NumbersOfBooks, b.Book\_Prise  FROM Book b  JOIN Genres gs ON b.ID\_Book = gs.ID\_Book  JOIN Genre g ON gs.GenresCode = g.GenreCode  WHERE g.NameGenre = p\_NameGenree;  -- Очищаем буфер DBMS\_OUTPUT  DBMS\_OUTPUT.PUT\_LINE('Книги с жанром: ' || p\_NameGenree || ' ');  LOOP  FETCH prc\_books\_cursor INTO getBookName, getBookPages, getBookISBN, getBookBinding, getBookYearPublic,  getBookIDPublishHous, getBookLanguage, getBookDescript, getBookImageSoursce,  getBookNumbersOfBooks, getBookPrice;  EXIT WHEN prc\_books\_cursor%NOTFOUND;  -- Выводим информацию о книге  DBMS\_OUTPUT.PUT\_LINE('Название книги: ' || getBookName);  DBMS\_OUTPUT.PUT\_LINE('Количество страниц: ' || getBookPages);  DBMS\_OUTPUT.PUT\_LINE('ISBN: ' || getBookISBN);  DBMS\_OUTPUT.PUT\_LINE('Тип переплета: ' || getBookBinding);  DBMS\_OUTPUT.PUT\_LINE('Год публикации: ' || getBookYearPublic);  DBMS\_OUTPUT.PUT\_LINE('Идентификатор издательства: ' || getBookIDPublishHous);  DBMS\_OUTPUT.PUT\_LINE('Язык книги: ' || getBookLanguage);  DBMS\_OUTPUT.PUT\_LINE('Описание книги: ' || getBookDescript);  DBMS\_OUTPUT.PUT\_LINE('Источник изображения: ' || getBookImageSoursce);  DBMS\_OUTPUT.PUT\_LINE('Количество книг: ' || getBookNumbersOfBooks);  DBMS\_OUTPUT.PUT\_LINE('Цена книги: ' || getBookPrice);  DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------------------');  END LOOP;  CLOSE prc\_books\_cursor;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Нету книг с жанром ' || p\_NameGenree);  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Возникла ошибка ' || SQLERRM);  END;  exec GET\_BOOK\_BYGENRE('Детектив');  ------------------    --ФИЛЬТРАЦИЯ по АВТОРУ  CREATE OR REPLACE PROCEDURE Get\_Book\_ByAuthor(  p\_AuthorName IN Author.FullName%TYPE  ) AS  prc\_books\_cursor SYS\_REFCURSOR;  getBookName Book.Book\_Name%TYPE ;  getBookPages Book.Page\_Number%TYPE;  getBookISBN Book.ISBN%TYPE ;  getBookBinding Book.BookBindingType%TYPE;  getBookYearPublic Book.Year\_Of\_Publication%TYPE;  getBookIDPublishHous Book.Book\_ID\_PublishingHouse%TYPE;  getBookLanguage Book.Book\_Language%TYPE;  getBookDescript Book.Book\_Description%TYPE;  getBookImageSoursce Book.Book\_ImageSource%TYPE;  getBookNumbersOfBooks Book.NumbersOfBooks%TYPE ;  getBookPrice Book.Book\_Prise%TYPE ;  BEGIN  OPEN prc\_books\_cursor FOR  SELECT b.Book\_Name, b.Page\_Number, b.ISBN, b.BookBindingType, b.Year\_Of\_Publication, b.Book\_ID\_PublishingHouse,  b.Book\_Language, b.Book\_Description, b.Book\_ImageSource, b.NumbersOfBooks, b.Book\_Prise  FROM Book b  JOIN Authors a ON b.ID\_Book = a.ID\_Book  WHERE a.ID\_Author IN (SELECT ID\_Author FROM Author WHERE FullName = p\_AuthorName);  -- Очищаем буфер DBMS\_OUTPUT  DBMS\_OUTPUT.PUT\_LINE('Книги автора: ' || p\_AuthorName || ' ');  LOOP  FETCH prc\_books\_cursor INTO getBookName, getBookPages, getBookISBN, getBookBinding, getBookYearPublic,  getBookIDPublishHous, getBookLanguage, getBookDescript, getBookImageSoursce,  getBookNumbersOfBooks, getBookPrice;  EXIT WHEN prc\_books\_cursor%NOTFOUND;  -- Выводим информацию о книге  DBMS\_OUTPUT.PUT\_LINE('--------------------------------');  DBMS\_OUTPUT.PUT\_LINE('Название книги: ' || getBookName);  DBMS\_OUTPUT.PUT\_LINE('Количество страниц: ' || getBookPages);  DBMS\_OUTPUT.PUT\_LINE('ISBN: ' || getBookISBN);  DBMS\_OUTPUT.PUT\_LINE('Тип переплета: ' || getBookBinding);  DBMS\_OUTPUT.PUT\_LINE('Год публикации: ' || getBookYearPublic);  DBMS\_OUTPUT.PUT\_LINE('Идентификатор издательства: ' || getBookIDPublishHous);  DBMS\_OUTPUT.PUT\_LINE('Язык книги: ' || getBookLanguage);  DBMS\_OUTPUT.PUT\_LINE('Описание книги: ' || getBookDescript);  DBMS\_OUTPUT.PUT\_LINE('Источник изображения: ' || getBookImageSoursce);  DBMS\_OUTPUT.PUT\_LINE('Количество книг: ' || getBookNumbersOfBooks);  DBMS\_OUTPUT.PUT\_LINE('Цена книги: ' || getBookPrice);  DBMS\_OUTPUT.PUT\_LINE('--------------------------------');  END LOOP;  CLOSE prc\_books\_cursor;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Нету книги с таким автором ' || p\_AuthorName);  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Возникла ошибка ' || SQLERRM);  END;    EXEC GET\_BOOK\_BYAUTHOR ('Джордж Оруэлл');  --ФИЛЬТРАЦИЯ по ЯЗЫКУ    CREATE OR REPLACE PROCEDURE Get\_Book\_ByLanguage(  p\_Language IN Book.Book\_Language%TYPE  ) AS  prc\_books\_cursor SYS\_REFCURSOR;  getBookName Book.Book\_Name%TYPE ;  getBookPages Book.Page\_Number%TYPE;  getBookISBN Book.ISBN%TYPE ;  getBookBinding Book.BookBindingType%TYPE;  getBookYearPublic Book.Year\_Of\_Publication%TYPE;  getBookDescript Book.Book\_Description%TYPE;  getBookImageSoursce Book.Book\_ImageSource%TYPE;  getBookNumbersOfBooks Book.NumbersOfBooks%TYPE ;  getBookPrice Book.Book\_Prise%TYPE ;  BEGIN  OPEN prc\_books\_cursor FOR  SELECT b.Book\_Name, b.Page\_Number, b.ISBN, b.BookBindingType, b.Year\_Of\_Publication,  b.Book\_Description, b.Book\_ImageSource, b.NumbersOfBooks, b.Book\_Prise  FROM Book b  WHERE b.Book\_Language = p\_Language;  -- Очищаем буфер DBMS\_OUTPUT  DBMS\_OUTPUT.PUT\_LINE('Книги на языке: ' || p\_Language || ' ');  LOOP  FETCH prc\_books\_cursor INTO getBookName, getBookPages, getBookISBN, getBookBinding, getBookYearPublic,  getBookDescript, getBookImageSoursce,  getBookNumbersOfBooks, getBookPrice;  EXIT WHEN prc\_books\_cursor%NOTFOUND;  -- Выводим информацию о книге  DBMS\_OUTPUT.PUT\_LINE('Название книги: ' || getBookName);  DBMS\_OUTPUT.PUT\_LINE('Количество страниц: ' || getBookPages);  DBMS\_OUTPUT.PUT\_LINE('ISBN: ' || getBookISBN);  DBMS\_OUTPUT.PUT\_LINE('Тип переплета: ' || getBookBinding);  DBMS\_OUTPUT.PUT\_LINE('Год публикации: ' || getBookYearPublic);  DBMS\_OUTPUT.PUT\_LINE('Описание книги: ' || getBookDescript);  DBMS\_OUTPUT.PUT\_LINE('Источник изображения: ' || getBookImageSoursce);  DBMS\_OUTPUT.PUT\_LINE('Количество книг: ' || getBookNumbersOfBooks);  DBMS\_OUTPUT.PUT\_LINE('Цена книги: ' || getBookPrice);  DBMS\_OUTPUT.PUT\_LINE('--------------------------------');  END LOOP;  CLOSE prc\_books\_cursor;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Нету книги на таком языку: ' || p\_Language);  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Возникла ошибка: ' || SQLERRM);  END;    EXEC GET\_BOOK\_BYLANGUAGE('английский');    ---ПРОВЕРКА наличия определенной книги  ------------------- с курсором ----------------  CREATE OR REPLACE PROCEDURE CheckBookAvailabilityCURS(BookName IN VARCHAR2)  AS  CURSOR c\_book IS  SELECT ID\_Book, Book\_Name, NumbersOfBooks  FROM Book  WHERE Book\_Name = BookName AND NumbersOfBooks > 0;    v\_book\_id Book.ID\_Book%TYPE;  v\_book\_name Book.Book\_Name%TYPE;  v\_numbers\_of\_books Book.NumbersOfBooks%TYPE;  BEGIN  OPEN c\_book;  FETCH c\_book INTO v\_book\_id, v\_book\_name, v\_numbers\_of\_books;    IF c\_book%FOUND THEN  -- Можно выполнить дополнительные действия с полученными данными  -- Пример вывода результатов  DBMS\_OUTPUT.PUT\_LINE('Book ID: ' || v\_book\_id);  DBMS\_OUTPUT.PUT\_LINE('Book Name: ' || v\_book\_name);  DBMS\_OUTPUT.PUT\_LINE('Numbers of Books: ' || v\_numbers\_of\_books);  ELSE  DBMS\_OUTPUT.PUT\_LINE('Книга отсутствует или недоступна.');  END IF;    CLOSE c\_book;  END;  EXEC CheckBookAvailabilityCURS('NO book');          ---ЗАКАЗ КНИГ  CREATE OR REPLACE PROCEDURE InsertOrderInf(  p\_ID\_Order IN OrderInf.ID\_Order%TYPE,  p\_ID\_User IN OrderInf.ID\_User%TYPE,  p\_Order\_Date IN OrderInf.Order\_Date%TYPE,  p\_Delivery\_Time IN OrderInf.Delivery\_Time%TYPE,  p\_ID\_Orders IN Orders.ID\_Orders%TYPE,  p\_ID\_Book IN Orders.ID\_Book%TYPE,  p\_NumberOrderedBooks IN Orders.NumberOrderedBooks%TYPE  )  IS  v\_error\_message VARCHAR2(200); -- Переменная для хранения сообщения об ошибке  v\_Price Book.Book\_Prise%TYPE; -- Переменная для хранения цены книги  v\_Amount OrderInf.Amount%TYPE; -- Переменная для хранения значения Amount  BEGIN  -- Получение цены книги из таблицы Book  SELECT Book\_Prise INTO v\_Price FROM Book WHERE ID\_Book = p\_ID\_Book;  -- Рассчет Amount на основе цены и количества заказанных книг  v\_Amount := v\_Price \* p\_NumberOrderedBooks;    -- Вставка данных в таблицу OrderInf  INSERT INTO OrderInf(ID\_Order, ID\_User, Order\_Date, Delivery\_Time, Amount)  VALUES (p\_ID\_Order, p\_ID\_User, p\_Order\_Date, p\_Delivery\_Time, v\_Amount);    -- Вставка данных в таблицу Orders  INSERT INTO Orders(ID\_Orders, ID\_Book, NumberOrderedBooks)  VALUES (p\_ID\_Orders, p\_ID\_Book, p\_NumberOrderedBooks);    COMMIT; -- Фиксация изменений    EXCEPTION  WHEN NO\_DATA\_FOUND THEN  v\_error\_message := SQLERRM; -- Получение сообщения об ошибке  ROLLBACK; -- Откат транзакции  ROLLBACK; -- Откат транзакции  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка при вставке данных: ' || v\_error\_message);  ROLLBACK; -- Откат транзакции    END;    Exec InsertOrderInf(1, 1, SYSDATE, SYSDATE+2, 1, 2, 2);  --Процедуры для админа  --ДЛЯ АДМИНА  -- Получить информацию о Юзере  create or replace procedure GetUsersAccount (  accusers OUT SYS\_REFCURSOR  )  is  begin  OPEN accusers FOR  SELECT ID\_Account, Login, Password  FROM UserAccount;  end;  VAR ACCCursor REFCURSOR;  EXEC GetUsersAccount(:ACCCursor);  PRINT ACCCursor;    ---ДОБАВЛЕНИЕ книг    -- INSERT NEW BOOK  CREATE OR REPLACE PROCEDURE ADD\_NEW\_BOOK(  addBookName IN Book.Book\_Name%TYPE ,  addBookPages IN Book.Page\_Number%TYPE ,  addBookISBN IN Book.ISBN%TYPE ,  addBookBinding IN Book.BookBindingType%TYPE ,  addBookYearPublic IN Book.Year\_Of\_Publication%TYPE,  addBookIDPublishHous IN Book.Book\_ID\_PublishingHouse%TYPE,  addBookLanguage IN Book.Book\_Language%TYPE ,  addBookDescript IN Book.Book\_Description%TYPE ,  addBookImageSoursce IN Book.Book\_ImageSource%TYPE ,  addBookNumbersOfBooks IN Book.NumbersOfBooks%TYPE ,  addBookPrice IN Book.Book\_Prise%TYPE  )  IS  pr\_book\_id INTEGER;  BEGIN  -- получаем новый уникальный идентификатор записи  INSERT INTO BOOK(ID\_Book, Book\_Name, Page\_Number, ISBN, BookBindingType, Year\_Of\_Publication, Book\_ID\_PublishingHouse, Book\_Language, Book\_Description, Book\_ImageSource, NumbersOfBooks, Book\_Prise) VALUES (NULL, addBookName, addBookPages, addBookISBN, addBookBinding, addBookYearPublic, addBookIDPublishHous, addBookLanguage, addBookDescript, addBookImageSoursce, addBookNumbersOfBooks, addBookPrice);  COMMIT;  END;    EXEC ADD\_NEW\_BOOK('NO book', 200, '978-5-17-100857-4', 'Твердый переплет', 2020, 2 , 'английский', 'описанииеее', 'image.jpg', 0, 25);      --ДОБАВЛЕНИЕ АВТОРОВ Разбиение строки для авторов  CREATE OR REPLACE PROCEDURE ADD\_AUTHORS(  prc\_author IN VARCHAR2,  prc\_FullName in Author.FullName%type,  prc\_BirthYear in Author.BirthYear%type,  prc\_BirthCountry in Author.BirthCountry%type  )  IS  v\_author VARCHAR2(2000) := prc\_author; -- создаем новую переменную для разбиения строки  v\_i INTEGER := 1; -- счетчик для цикла WHILE  v\_pos INTEGER; -- позиция разделителя авторов  BEGIN  -- Разбиваем строку авторов на отдельные  WHILE v\_i <= LENGTH(v\_author)  LOOP  v\_pos := INSTR(v\_author, ';', v\_i); -- ищем позицию разделителя  IF v\_pos = 0 -- если разделитель не найден, то вставляем последнего автора и выходим из цикла  THEN  INSERT INTO Author(ID\_Author, FullName, BirthYear, BirthCountry) VALUES (null, TRIM(SUBSTR(v\_author, v\_i)), prc\_BirthYear, prc\_BirthCountry);  INSERT INTO Authors(ID\_Author, ID\_Book) VALUES ((SELECT MAX(ID\_Author) FROM Author), (SELECT MAX(ID\_Book) FROM Authors));  EXIT;    ELSE  INSERT INTO Author(ID\_Author, FullName, BirthYear, BirthCountry) VALUES (null, TRIM(SUBSTR(v\_author, v\_i, v\_pos - v\_i)), prc\_BirthYear, prc\_BirthCountry);  INSERT INTO Authors(ID\_Author, ID\_Book)VALUES ((SELECT MAX(ID\_Author) FROM Author), (SELECT MAX(ID\_Book) FROM Authors));  v\_i := v\_pos + 1;  END IF;  END LOOP;  COMMIT;  END;    Exec ADD\_AUTHORS('Жук;Косминов', 'Full Name', 1980, 'Country');  select \* from Author;  ------------------------------------------------------------  --Разбиение строки для жанров  CREATE OR REPLACE PROCEDURE ADD\_GENRES(  prc\_genres IN VARCHAR2  )  IS  v\_genres VARCHAR2(2000) := prc\_genres; -- создаем новую переменную для разбиения строки  v\_i INTEGER := 1; -- счетчик для цикла WHILE  v\_pos INTEGER; -- позиция разделителя жанров  BEGIN  -- Разбиваем строку жанров на отдельные жанры  WHILE v\_i <= LENGTH(v\_genres)  LOOP  v\_pos := INSTR(v\_genres, ';', v\_i); -- ищем позицию разделителя  IF v\_pos = 0 -- если разделитель не найден, то вставляем последний жанр и выходим из цикла  THEN  INSERT INTO Genre(GenreCode, NameGenre) VALUES (null, TRIM(SUBSTR(v\_genres, v\_i)));  INSERT INTO Genres(ID\_Book, GenresCode) VALUES ((SELECT MAX(ID\_Book) FROM Book), (SELECT MAX(GenreCode) FROM Genre));  EXIT;  ELSE -- иначе вставляем жанр и продолжаем цикл  INSERT INTO Genre(GenreCode, NameGenre) VALUES (null, TRIM(SUBSTR(v\_genres, v\_i, v\_pos - v\_i)));  INSERT INTO Genres (ID\_Book, GenresCode) VALUES ((SELECT MAX(ID\_Book) FROM Book), (SELECT MAX(GenreCode) FROM Genre));  v\_i := v\_pos + 1;  END IF;  END LOOP;  COMMIT;  END;    EXEC ADD\_GENRES('Genre1;Genre2;Genre3');  Select \* from Genre;      --ОБНОВЛЕНИЕ книг  CREATE OR REPLACE PROCEDURE UPDATE\_BOOKINF(  proc\_book\_id IN Book.ID\_Book%TYPE ,  updateBookName IN Book.Book\_Name%TYPE ,  updateBookPages IN Book.Page\_Number%TYPE ,  updateBookISBN IN Book.ISBN%TYPE ,  updateBookBinding IN Book.BookBindingType%TYPE ,  updateBookYearPublic IN Book.Year\_Of\_Publication%TYPE,  updateBookIDPublishHous IN Book.Book\_ID\_PublishingHouse%TYPE,  updateBookLanguage IN Book.Book\_Language%TYPE ,  updateBookDescript IN Book.Book\_Description%TYPE ,  updateBookImageSoursce IN Book.Book\_ImageSource%TYPE ,  updateBookNumbersOfBooks IN Book.NumbersOfBooks%TYPE ,  updateBookPrice IN Book.Book\_Prise%TYPE  )  IS  book\_exists NUMBER;  BEGIN  -- Проверяем наличие книги  SELECT COUNT(\*) INTO book\_exists FROM Book WHERE ID\_Book = proc\_book\_id;  IF book\_exists = 0 THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Книга не найдена');  ELSE  UPDATE Book  SET Book\_Name = updateBookName,  Page\_Number = updateBookPages,  ISBN = updateBookISBN,  BookBindingType = updateBookBinding,  Year\_Of\_Publication = updateBookYearPublic,  Book\_ID\_PublishingHouse = updateBookIDPublishHous,  Book\_Language = updateBookLanguage,  Book\_Description = updateBookDescript,  Book\_ImageSource = updateBookImageSoursce,  NumbersOfBooks = updateBookNumbersOfBooks,  Book\_Prise = updateBookPrice  WHERE ID\_Book = proc\_book\_id;    COMMIT;    DBMS\_OUTPUT.PUT\_LINE('Информация о книге успешно обнавлена.');  END IF;    EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка! Книга не найдена');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка: ' || SQLERRM);  END;  Select \* from Book;  EXEC UPDATE\_BOOKINF(22, 'Update Book2', 211,'5-17-657342-6', 'Мягкая обложка', 2018,3, 'русский', 'descr', 'img',2, 20);      --ОБНОВЛЕНИЕ авторов    CREATE OR REPLACE PROCEDURE UPDATE\_Author(  prc\_BookId IN Book.ID\_Book%type,  prc\_author IN VARCHAR2,  prc\_FullName in Author.FullName%type,  prc\_BirthYear in Author.BirthYear%type,  prc\_BirthCountry in Author.BirthCountry%type  )  IS  v\_author VARCHAR2(2000) := prc\_author; -- создаем новую переменную для разбиения строки  v\_i INTEGER := 1; -- счетчик для цикла WHILE  v\_pos INTEGER; -- позиция разделителя песен  BEGIN  DELETE (SELECT \*  FROM Authors aus  INNER JOIN Author aur  ON aus.ID\_AUTHOR = aur.ID\_AUTHOR  WHERE aus.ID\_AUTHOR = prc\_BookId);  -- Разбиваем строку авторов на отдельных  WHILE v\_i <= LENGTH(v\_author)  LOOP  v\_pos := INSTR(v\_author, ';', v\_i); -- ищем позицию разделителя  IF v\_pos = 0 -- если разделитель не найден, то вставляем последнего автора и выходим из цикла  THEN  update Author set FullName = TRIM(SUBSTR(v\_author, v\_i)), BirthYear = prc\_BirthYear, BirthCountry = prc\_BirthCountry;  update Authors set ID\_Author = (SELECT MAX(ID\_Author) FROM Author), ID\_AUTHOR = prc\_BookId;  -- INSERT INTO Author(ID\_Author, FullName, BirthYear, BirthCountry) VALUES (null, TRIM(SUBSTR(v\_author, v\_i)), prc\_BirthYear, prc\_BirthCountry);  -- INSERT INTO Authors(ID\_Author, ID\_Book) VALUES ((SELECT MAX(ID\_Author) FROM Author), (SELECT MAX(ID\_Book) FROM Authors));EXIT;  ELSE -- иначе вставляем автора и продолжаем цикл  INSERT INTO Author(ID\_Author, FullName, BirthYear, BirthCountry) VALUES (null, TRIM(SUBSTR(v\_author, v\_i, v\_pos - v\_i)), prc\_BirthYear, prc\_BirthCountry);  INSERT INTO Authors(ID\_Author, ID\_Book)VALUES ((SELECT MAX(ID\_Author) FROM Author), (SELECT MAX(ID\_Book) FROM Authors));  v\_i := v\_pos + 1;  END IF;  END LOOP;  COMMIT;  END;  EXEC UPDATE\_Author(22, 'Новый автор 1', 'Новое имя',1990,'Новая страна');  Select \* from Author;      --ДЛЯ АДМИНА  --Вывод всей инфы о КЛИЕНТАХ и их АККАУНТАХ  CREATE OR REPLACE PROCEDURE getUsersInf(  usersi OUT SYS\_REFCURSOR  )  IS  BEGIN  OPEN usersi FOR  SELECT uAcc.ID\_Account,uAcc.ID\_UserAc, uAcc.Login ,uAcc.Password, uInf.User\_Name, uInf.User\_Email, uInf.User\_Numer, uInf.User\_Sex, uInf.User\_BirthYear  FROM UserAccount uAcc  JOIN UserInf uInf ON uAcc.ID\_UserAc = uInf.ID\_User;  END;  --test  VAR userCurs REFCURSOR;  EXEC getUsersInf(:userCurs);  PRINT userCurs;    --Вывод всей инфы о ЗАКАЗАХ  -- SUM(bk.Book\_Prise)  CREATE OR REPLACE PROCEDURE getOrdersInf(  orderi OUT SYS\_REFCURSOR  )  IS  BEGIN  OPEN orderi FOR  SELECT ordInf.ID\_Order, ordInf.ID\_User, ordInf.Order\_Date, ordInf.Delivery\_Time, ordInf.Amount, ords.NumberOrderedBooks, bk.Book\_Name  FROM Book bk  JOIN Orders ords ON bk.ID\_Book = ords.ID\_Book  JOIN OrderInf ordInf ON ords.ID\_Orders = ordInf.ID\_Order;  END;  --test  VAR orderCurs REFCURSOR;  EXEC getOrdersInf(:orderCurs);  PRINT orderCurs;  --Резервное копирование и востановление  > sqlplus / as sysdba  SQL> startup  RMAN> connect target /  RMAN> alter system set db\_recovery\_file\_dest\_size=4096M scope=both;  RMAN> alter system set db\_recovery\_file\_dest=' D:\BackUp\' scope=both;  SQL> SELECT log\_mode FROM v$database;  SQL> SHUTDOWN IMMEDIATE; SQL> STARTUP MOUNT;  SQL> ALTER DATABASE ARCHIVELOG;  SQL> ALTER DATABASE OPEN;  SQL> ALTER SYSTEM SWITCH LOGFILE;  RMAN> backup as compressed backupset database plus archivelog;  Удаление файлов dbf  > rman  RMAN> connect target /  RMAN> shutdown abort  RMAN> startup mount  RMAN> restore database;  RMAN> recover database;  RMAN> alter database open; |