МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 05 01 Информационные системы и технологии

Специализация

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка базы данных “Книжное издательство” с применением технологии резервного копирования и восстановления данных»

Выполнил студент Шимчёнок Елизавета Константиновна

(Ф.И.О.)

Руководитель проекта

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант:

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер:

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

**Оглавление**

[Введение 3](#_Toc134809350)

[1 Аналитический обзор литературы по теме проекта 4](#_Toc134809351)

[1.1 Обзор аналогов 4](#_Toc134809352)

[1.2 Изучение требований, определение вариантов использования 7](#_Toc134809353)

[1.3 Вывод к разделу 7](#_Toc134809354)

[2 Разработка базы данных 8](#_Toc134809355)

[2.1 Проектирование модели базы данных 8](#_Toc134809356)

[2.2 Разработка объектов базы данных 9](#_Toc134809357)

[2.2.1 Таблицы 9](#_Toc134809362)

[2.2.2 Пользователи 9](#_Toc134809363)

[2.2.3 Процедуры 9](#_Toc134809364)

[2.2.4 Функции 10](#_Toc134809365)

[2.3 Вывод к разделу 10](#_Toc134809366)

[3 Описание процедур импорта и экспорта 11](#_Toc134809367)

[4 Тестирование производительности 13](#_Toc134809368)

[5 Описание используемых технологий 15](#_Toc134809369)

[5.1 Резервное копирование и восстановление 15](#_Toc134809370)

[5.2 Вывод к разделу 16](#_Toc134809371)

[Заключение 17](#_Toc134809372)

[Список используемых источников 18](#_Toc134809373)

[ПРИЛОЖЕНИЕ А 19](#_Toc134809374)

[ПРИЛОЖЕНИЕ Б 22](#_Toc134809375)

[ПРИЛОЖЕНИЕ В 23](#_Toc134809376)

[ПРИЛОЖЕНИЕ Г 28](#_Toc134809377)

# **Введение**

Книги играют значительную роль в нашей жизни. Чтение книг расширяет знания студентов, улучшает их интеллект, знакомит студентов с различными обществами и цивилизациями по всему миру. Более того, чтение книг развивает воображение и творческие способности в сознании учащегося.

Однако книга не может выпустить себя сама, поэтому на помощь приходит издатель – посредник между автором и читателем. Он берет на себя все расходы по распространению книги и привлечению новой аудитории. Он стратег и тактик, благодаря которому тысячи произведений находят своего читателя». Пользоваться издательствами советуют по некоторым причинам.

В издательствах работают опытные редакторы, которые тесно сотрудничают с авторами, чтобы улучшить качество их работы. Эти редакторы предоставляют отзывы, предлагают исправления и помогают авторам преобразовать свои рукописи в законченный продукт.

У издательств есть ресурсы для производства книг в больших количествах и распространения их в книжных магазинах, библиотеках и интернет-магазинах. Это помогает обеспечить доступность книг для широкой аудитории и доступность в различных форматах.

В издательствах есть отделы маркетинга и рекламы, которые помогают продвигать книги через рекламу, социальные сети, авторские мероприятия и другие каналы. Это помогает создать шумиху и интерес к книге до ее выпуска, что может увеличить ее шансы на успех.

Издательства управляют правами на книгу, которые включают в себя лицензионные права на перевод, права на экранизацию и телевизионную адаптацию, а также права на аудиокниги. Это может помочь авторам получить дополнительный доход от своей работы и расширить свою аудиторию.

Для того, чтобы делать все вышеперечисленное, издательствам требуется база данных. База данных – это организованная структура данных, предназначенная для хранения взаимосвязанной информации, систематизированная таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины. Реляционная база данных – база данных, основанная на реляционной модели данных.

Целью данного курсового проекта является разработка реляционной базы данных на тему «Книжное издательство». В качестве СУБД для базы данных была выбрана Oracle Database 12c, в соответствии с требованиями к курсовой работе.

В связи с целью были выдвинуты следующие задачи:

– исследование аналогов;

– изучение требований к будущему проекту и его назначения;

– проектирование базы данных;

– создание пользователей и определение их доступа к базе данных;

– разработка необходимых объектов для работы с данными.

# **Аналитический обзор литературы по теме проекта**

## **Обзор аналогов**

Постепенно с развитием программного обеспечения ЭВМ появились идеи создания управляющих систем, которые позволяли бы накапливать, хранить и обновлять взаимосвязанные данные по целому комплексу решаемых задач. Эти идеи нашли свое воплощение в системах управления базами данных (СУБД). СУБД взаимодействуют не с локальными, а взаимосвязанными по информации массивами, называемыми базами данных. С появлением персональных компьютеров СУБД становятся наиболее популярным средством обработки табличной информации.

Можно рассмотреть структуру организации работы базы данных, связанных с учетом книг, их авторов и статистикой по популярности.

Первым приложением рассмотрим LibraryThing (рисунок 1.1). Оно является одним из популярных приложений для учета книг и организации вашей личной библиотеки. Оно предлагает множество функций, которые помогут вам управлять вашей коллекцией книг.

В данном приложении существует возможность сканирования штрихкодов. Приложение позволяет сканировать штрихкоды книг для быстрого добавления их в вашу библиотеку.

Также возможна каталогизация книг. Можно создавать подробные записи для каждой книги, включая информацию об авторе, издательстве, годе издания и обложке.

LibraryThing позволяет также создавать собственные категории и теги для организации ваших книг по различным критериям.

Пользователи могут присоединяться к группам, обсуждать книги, рекомендовать чтение и обмениваться рецензиями с другими членами сообщества.

Идеальное решение для тех, кто стремится поддерживать аккуратный учет своих литературных погружений. Это приложение, которое позволяет записывать прочитанные книги, оставлять собственные рецензии, создавать виртуальные книжные полки с учетом жанров (или создать собственную систему классификации) и добавлять интересующие вас книги в список желаний.

Однако, приложение не только предоставляет удобные функции организации литературной коллекции, но также предлагает потрясающую статистику, которая позволяет вам получить полное представление о вашем чтении. Вы сможете узнать, сколько книг было прочитано за год или месяц, среднее количество страниц в день и многое другое.

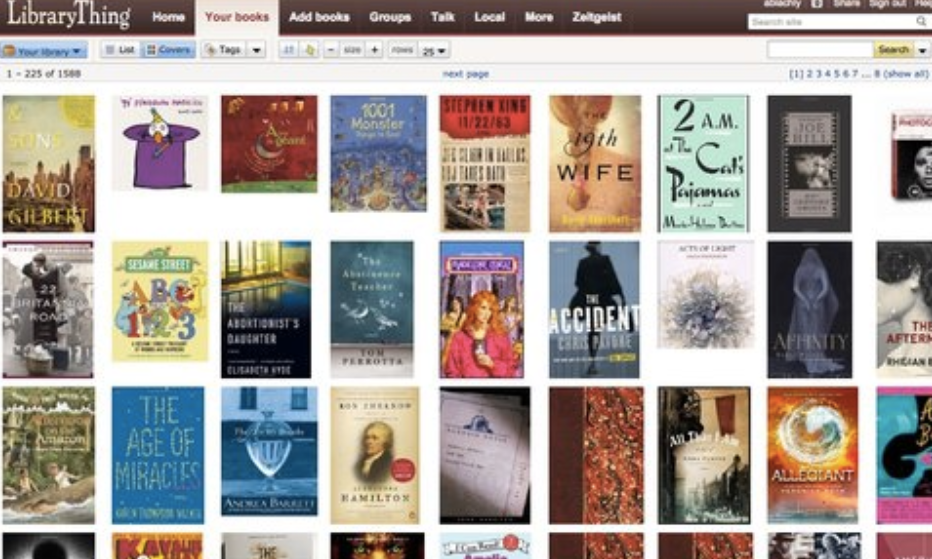


Рисунок 1.1 – LibraryThing

Shelfari – это еще одно популярное приложение для учета книг, созданное Amazon (рисунок 1.2). Оно предлагает ряд интересных функций, которые помогут вам управлять вашей личной библиотекой.

Визуальное представление полки – особенность данного приложения. Shelfari предлагает визуальное представление полки с книгами, которое позволяет легко просматривать свою коллекцию. Здесь можно писать рецензии и ставить оценки книгам, а также читать рецензии других пользователей. Shelfari предлагает рекомендации книг на основе ваших предпочтений и интересов, также можно рекомендовать книги другим пользователям.

Приложение также позволяет пользователям отмечать прочитанные ими книги как завершенные. Они могут вводить информацию о количестве прочитанных книг, жанрах и других деталях, чтобы иметь полный обзор о своих литературных предпочтениях и достижениях.

Приложение включает в себя интеграцию с Amazon и позволяет легко добавлять книги из магазина Amazon в пользовательскую библиотеку.

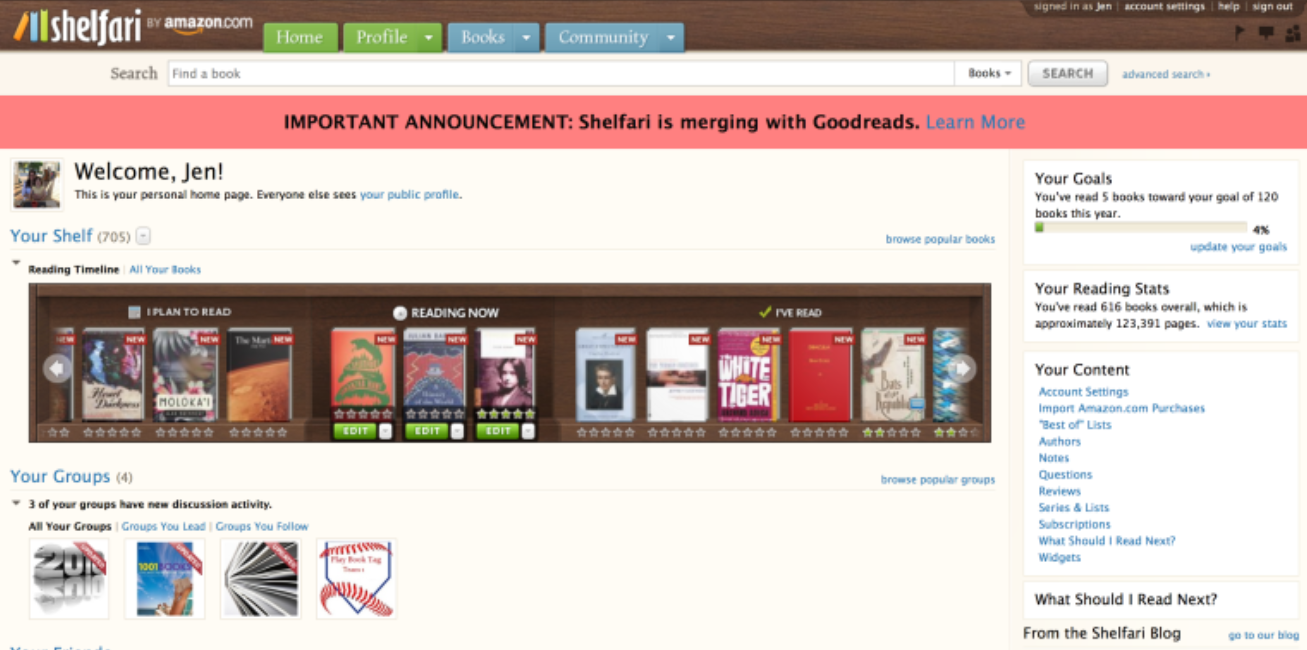


Рисунок 1.2 – Shelfari

Следующим приложением было выбрано ReadMore (рисунок 1.3).

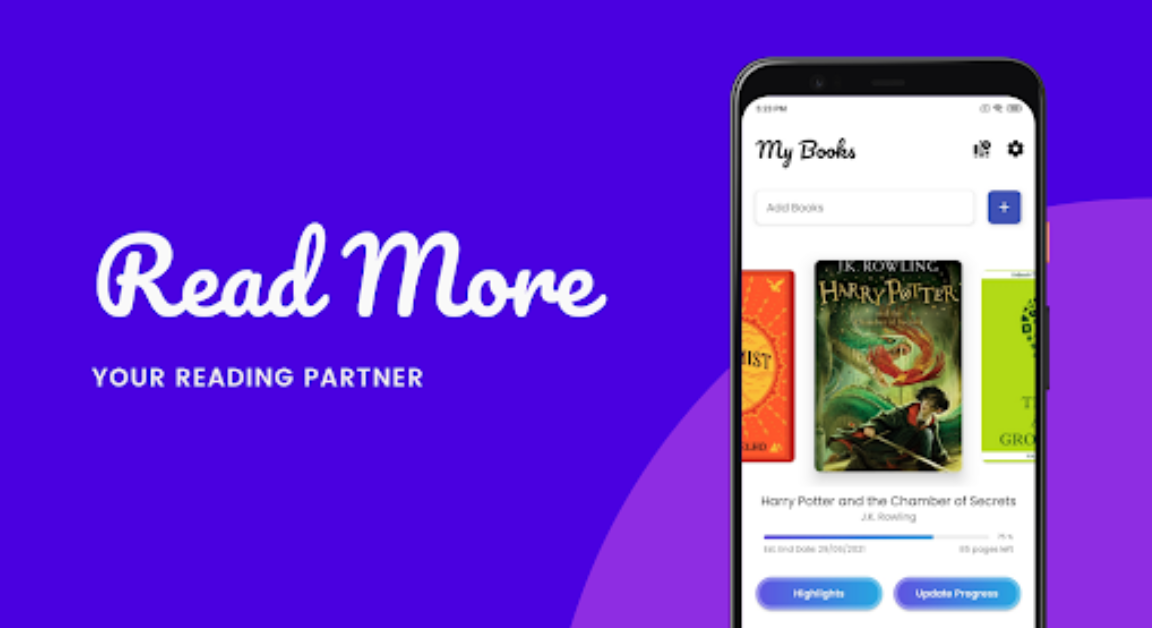


Рисунок 1.3 – ReadMore

Приложение ReadMore – это трекер привычки к чтению. Приложение предлагает множество функций, которые помогут выработать привычку читать каждый день (в приложении нет книг). Оно предлагает установить ежедневную цель чтения и постепенно ее увеличивать, вдохновляя своими статистиками с помощью мониторинга привычек.

Здесь можно составить список книг для чтения, чтобы можно было переходить от одной книги к другой без опасений забыть последнюю страницу.

В приложении также есть возможность создать коллекцию цитат и делиться ей с другими пользователями, а также отслеживать завершенные количество, жанр и другую информацию об уже завершенных книгах.

ReadMore предлагает составить план чтения для более удобного и легкого чтения, что позволяет читать несколько книг одновременно. Приложение существует только в мобильной версии. Однако это означает, что пользователи могут иметь доступ к своим чтениям и мониторингу привычки в любое время и в любом месте через свои мобильные устройства.

Приложение ReadMore – это полезный инструмент для развития привычки чтения. Оно предлагает ряд функций, которые помогут пользователям установить и отслеживать свою ежедневную цель чтения.

LibraryThing и Shelfari предоставляют более широкий спектр функций, связанных с организацией библиотеки и взаимодействием с другими читателями. Они позволяют пользователям создавать подробные списки книг, добавлять теги и категории для классификации, писать рецензии и оценивать книги, присоединяться к сообществам и группам с общими интересами, обмениваться рекомендациями и участвовать в дискуссиях. Эти приложения создают общую платформу для взаимодействия и обмена литературным опытом с другими любителями чтения.

ReadMore, с другой стороны, сосредоточено на развитии привычки чтения и предлагает более специализированные функции для этой цели. Оно позволяет пользователям устанавливать ежедневные цели чтения и отслеживать свой прогресс, создавать списки книг для чтения и переключаться между ними без потери места, сохранять цитаты из прочитанных книг и делиться ими с другими пользователями. ReadMore сосредоточено на индивидуальном развитии читательских навыков и стимулирует пользователей читать регулярно и увеличивать объемы прочитываемого материала.

## **Изучение требований, определение вариантов использования**

На основе рассмотренных аналогов можно сделать вывод, что людям нравится использовать приложения для организации какой-либо сферы своей жизни, в данном случае это может быть развитие, образование или даже развлечение. Упорядочивание своей жизни в наши времена стало достаточно популярно, благодаря чему появляется множество приложений с множеством удобств, включая статистику, уведомления и приятный дизайн.

Таким образом, основным требованием к разрабатываемому проекту является обеспечение удобного и простого способа работы с данными, независимо от того, будет работать профессионал с этими данными или новичок.

Данный проект будет нацелен на работников издательства и людей, которые с ним сотрудничают – авторы изданий, сотрудники типографий и заказчики, скупающие печатную продукцию. Он может использоваться в качестве мониторинга информации о заказах и их статусе выполнения, а также для просмотра информации о том, с кем сотрудничает издательство.

## **Вывод к разделу**

В данном разделе были рассмотрены аналоги реализуемых методов и приложения в целом. У каждого аналога есть свои плюсы и минусы, на что не стоит закрывать глаза при проектировании своего приложения. Далее будут рассмотрены проектирование и разработка базы данных.

# **Разработка базы данных**

## **Проектирование модели базы данных**

При разработке курсового проекта главной задачей являлось проектирование базы данных. Для этого была использована Oracle 12c.

База данных состоит из 8 таблиц, диаграмма которых представлена на рисунке 2.1.

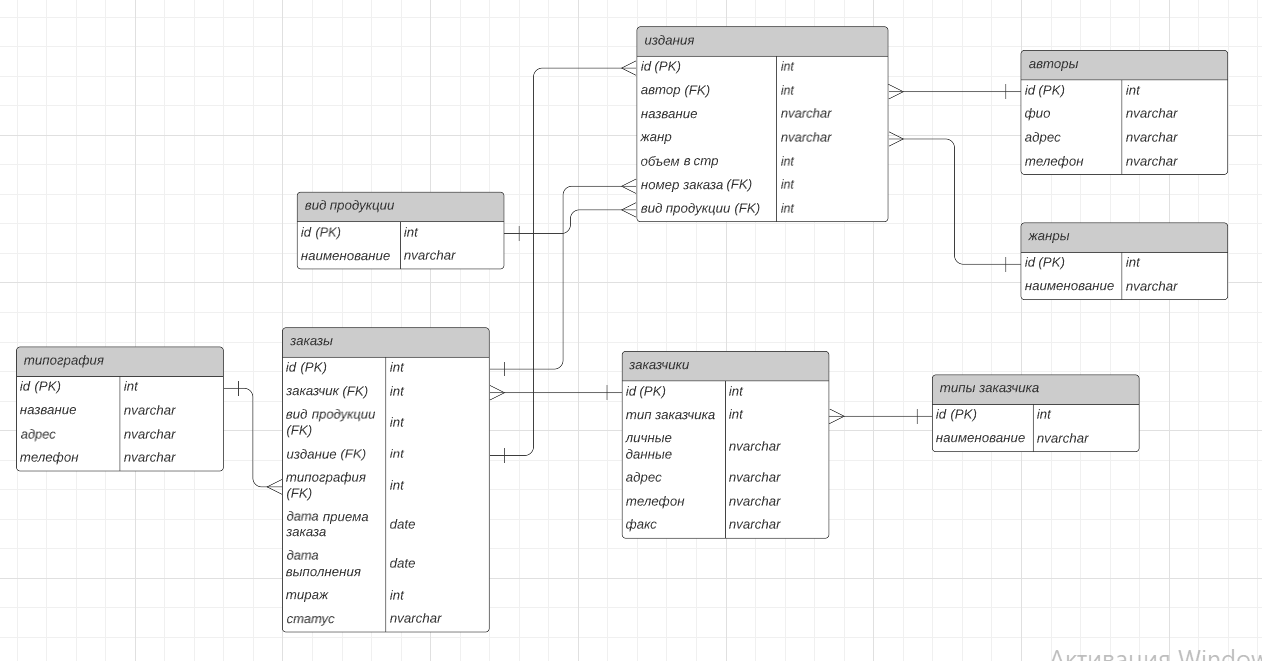


Рисунок 2.1 – Диаграмма таблиц пользовательских данных

Таблицы связаны между собой внешними ключами. В целом разработанная база данных должна хранить информацию о книгах, их авторах, виде издания (журнал/книга), заказах, заказчиках и типографиях, с которыми сотрудничает наше издательство.

Таблица «Авторы» содержит информацию о каждом авторе, который выпускает свое издание в издательстве.

Таблица «Издания» имеет больше всего связей. Она соединена внешним ключом с таблицами, содержащими информацию о заказах, жанрах, авторах и видах изданий. Сама хранит в себе данные о конкретных печатных продуктах, включая их название, автора, объем в страницах и тд.

Таблица «Вид продукции» состоит из двух полей: id (PK) и наименование. Точно также как и таблица «Жанры» и «Тип заказчика». Они предназначены для определения устоявшегося набора жанров и типов для более удобной работы в базе данных.

Таблица «Заказы» содержит информацию о идентификационном номере, заказчиках (FK), виде продукции (FK), издании (FK), типографии (FK), дате приема заказа, дате выполнения заказа и тд. Главное поле, которое потребуется для мониторинга работы издательства – статус заказа. Он описывает выполнен ли заказ или еще в процессе. Также в таблице можно посмотреть в какие сроки выполнен тот или иной заказ.

Таблица «Заказчики» хранит информацию о заказах. Она связана с таблицей «Заказы» и требуется для связи с покупателями.

Таблица «Типография» содержит информацию о типографиях, с которыми сотрудничает издательство для того, чтобы печатать заказы. Издательство может работать с несколькими типографиями одновременно, поэтому потребовалась таблица.

## **Разработка объектов базы данных**

Прежде, чем начать создавать таблицы, сначала потребовалось создать табличное пространство и выделить квоту общему пользователю, чтобы работать конкретно в этом пространстве. Далее можно было приступать к созданию объектов базы данных. При разработке курсового проекта понадобились следующие объекты:

– таблицы;

– пользователи;

– хранимые процедуры;

– функции.



### **Таблицы**

Таблицы были описаны в предыдущем разделе. Листинги к созданию таблиц будут приведены в приложении А. Далее необходимо было создать пользователей.

### **Пользователи**

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

При проектировании базы данных было использовано 3 пользователя: администратор, заказчик, автор. Каждый пользователь имеет разные привилегии в соответствии с его статусом. Администратор может изменять данные в базе. Добавлять, удалять и обновлять записи. Клиент может только просматривать информацию о типографиях, изданиях и заказах в ограниченном режиме.

Все скрипты создания пользователей и их ролей представлены в приложении Б данной записки.

### **Процедуры**

В проекте использовались хранимые процедуры.

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Таким образом, процедура будет принимать соответствующие аргументы при каждом ее вызове. Все хранимые процедуры, созданные в данном курсовом проекте, содержат обработку исключений для того, чтобы ограничить пользователя от возможных ошибок.

При разработке курсового проекта было создано множество процедур для следующих целей:

* выборка данных из таблиц;
* добавление/удаление/обновление жанров;
* добавление/удаление/обновление видов продукции;
* добавление/удаление/обновление авторов;
* добавление/удаление/обновление заказчиков;
* добавление/удаление/обновление заказов;
* добавление/удаление/обновление типографий;
* добавление/удаление/обновление изданий;
* экспорт и импорт таблицы в формат xml.

Все процедуры представлены в приложении В данной записки.

### **Функции**

Функции базы данных oracle выполняют те же самые задачи, что и процедуры. Главным их различием является то, что функции могут возвращать значения. Это может быть полезно в некоторых ситуациях.

В процессе реализации работы с данными были созданы функции следующего назначения – создание массива строк для вывода данных.

Все функции представлены в приложении Г данной записки.

## **Вывод к разделу**

В данном разделе была спроектирована база данных и описана разработка всех объектов базы данных для книжного издательства. Были созданы таблицы с внешними связями, пользователи, которые будут работать с данными из таблиц, и процедуры и функции, позволяющие различным пользователям совершать разные действия в связи с их ограниченным доступом. После того, как база данных готова к использованию, можно приступать к реализации выбранной технологии – резервное копирование и восстановление.

# **Описание процедур импорта и экспорта**

XML – это формат данных, используемый для обмена данными в форме, которая может быть легко использована и распространена. В данной курсовой работе используются пакеты DBMS\_XMLPARSER и DBMS\_XSLPROCESSOR для импорта xml, и DBMS\_XMLDOM для экспорта данных в xml формат. В данном курсовом проекте функции экспорта и импорта используются для таблицы PUBLICATIONS. Процедура экспорта представлена в листинге 3.1.

|  |
| --- |
| create or replace  procedure EXPORT\_PUBLICATIONS\_TOXML  is  DOC DBMS\_XMLDOM.DOMDocument;  XDATA XMLTYPE;  CURSOR XMLCUR IS  SELECT XMLELEMENT("PUBLICATIONS",  XMLAttributes('http://www.w3.org/2001/XMLSchema' AS "xmlns:xsi",  'http://www.oracle.com/Users.xsd' AS "xsi:nonamespaceSchemaLocation"),  XMLAGG(XMLELEMENT("Autoes",  xmlelement("publication\_id", PUBLICATIONS.PUBLICATION\_ID),  xmlelement("publication\_title", PUBLICATIONS.PUBLICATION\_TITLE),  xmlelement("publication\_author\_id", PUBLICATIONS.PUBLICATION\_AUTHOR\_ID),  xmlelement("publication\_genre\_id", PUBLICATIONS.PUBLICATION\_GENRE\_ID),  xmlelement("publication\_date", TO\_CHAR(PUBLICATIONS.publication\_date, 'DD-MM-YY')),  xmlelement("publication\_type\_id", PUBLICATIONS.publication\_type\_id),  xmlelement("volume\_in\_pages", PUBLICATIONS.volume\_in\_pages)  ))) from PUBLICATIONS;  begin  open xmlcur;  loop  fetch xmlcur into xdata;  exit when xmlcur%notfound;  end loop;  close xmlcur;  DOC := DBMS\_XMLDOM.NewDOMDocument(XDATA);  DBMS\_XMLDOM.WRITETOFILE(DOC, 'XML\_DIR/PUBLICATIONS.xml');  END; |

Листинг 3.1 – Создание процедуры Export\_Publications\_ToXml

Как видно в листинге 3.1, экспорт производится для таблицы «Издания», хранящей информацию о книгах и других видах продукции. В ней хранится поле «Дата публикации» типа Date, из-за чего в методе импорта могут возникнуть проблемы. Для их решения было реализовано преобразование формата поля при помощи функции TO\_CHAR.

Процедура IMPORT\_PUBLICATIONS\_FROMXML выполняет импорт данных из XML-файла в таблицу TESTPUBLICATIONS. В процедуре создается временный LOB для хранения содержимого XML-файла. Открывается BFILE в режиме только для чтения. Используя DBMS\_LOB.LOADCLOBFROMFILE, содержимое XML-файла загружается в LOB, и BFILE закрывается. Затем происходит фиксация изменения с помощью COMMIT. И создается экземпляр XML-парсера с помощью Dbms\_Xmlparser.Newparser. С помощью DBMS\_XMLPARSER.PARSECLOB, XML-парсер (P) разбирает содержимое LOB. Получается DOM-документ с помощью DBMS\_XMLPARSER.GETDOCUMENT. Получается корневой элемент (V\_ROOT\_ELEMENT) DOM-документа с помощью DBMS\_XMLDOM.Getdocumentelement и список дочерних узлов корневого элемента (V\_CHILD\_NODES) с помощью DBMS\_XMLDOM. GETCHILDRENBYTAGNAME. В цикле FOR происходит обход каждого дочернего узла. С помощью DBMS\_XSLPROCESSOR.VALUEOF и Dbms\_Xslprocessor.Valueof значения из XML-узлов извлекаются и присваиваются соответствующим полям в структуре PUBL. Выполняется операция INSERT INTO.

Для сохранения и управления XML-данными в реляционной таблице применяется специальный тип данных XMLType. Тип данных XMLType поставляется с набором специальных XML-методов, которые можно использовать для работы с объектами XMLType. Эти методы можно применять для выполнения как типичных операций в базе данных, наподобие проверки на предмет существования узла или извлечения узла, так и нескольких специальных операций, позволяющих получать доступ к XML-данными и манипулировать ими в виде части обычного SQL-оператора.

# **Тестирование производительности**

Тестирование производительности базы данных является важной задачей, поскольку позволяет оценить и улучшить работу системы баз данных. Оно позволяет определить максимальную пропускную способность базы данных, то есть количество операций чтения и записи, которые она может обрабатывать за определенное время. Это помогает оптимизировать конфигурацию и настройки базы данных для обеспечения требуемой производительности.

Также тестирование производительности позволяет измерить время отклика базы данных на запросы. Это важно для обеспечения быстрого и отзывчивого пользовательского опыта. Измерение времени отклика помогает идентифицировать узкие места в системе, которые могут быть оптимизированы.

Для тестирования производительности созданной базы данных, было проведено исследование, для которого требовалось заполнить таблицу на 100 000 строк. Это было произведено с помощью анонимного блока, приведенного в листинге 4.1.

|  |
| --- |
| declare  begin  for a in 1 .. 100000  loop  ADD\_AUTHOR('name' || a, 'surname' || a, ’11.04,1970’, 'country' || a, ‘gmail’ || a);  end loop;  end; |

Листинг 4.1 – Заполнение таблицы Authors данными

Далее осуществляется выборка автора с заданным сочетанием букв в имени, листинг 4.2.

|  |
| --- |
| declare  t1 number := dbms\_utility.get\_time();  begin  for r in (SELECT FIRST\_NAME FROM AUTHORS WHERE FIRST\_NAME like 'name1%') loop null; end loop;  dbms\_output.put\_line('query elapsed: '||(dbms\_utility.get\_time() - t1)/100);  END; |

Листинг 4.2 – Выборка автора с условием

Стоимость данной операции представлена на рисунке 4.1.



Рисунок 4.1 – Стоимость запроса

Теперь, чтобы повысить производительность запроса, был создан индекс, который представлен в листинге 4.3.

|  |
| --- |
| create index FIRST\_NAME on AUTHORS(FIRST\_NAME, LAST\_NAME); |

Листинг 4.3 – Создание индекса FIRST\_NAME

После создание индекса снова была снова посчитана стоимость запроса. Данная операция представлена на рисунке 4.2.



Рисунок 4.2 – Стоимость запроса после создания индекса

Как видно из рисунка 4.2, стоимость запроса уменьшилась. Был использован индекс по двум столбцам, для более быстрого нахождения информации. Запросы, которые будут выводить информацию этих атрибутов, станут более производительны.

Таким образом, с помощью индексов мы можем увеличить производительность наших процедур, что позволит нам быстрее получать необходимую информацию.

В целом, тестирование производительности базы данных помогает обеспечить эффективную работу системы, оптимизировать ее производительность, выявить и устранить проблемы.

# **Описание используемых технологий**

## **Резервное копирование и восстановление**

Резервная копия – это набор всех файлов, которые необходимы для восстановления корректной работы базы данных при частичном удалении файлов или нарушении структуры данных.

Существует два способа резервного копирования: холодное резервное копирование и горячее резервное копирование. Холодным резервным копированием называют копирование файлов, в то время, когда база данных отключена, то есть является недоступной для пользователей. Горячее резервное копирование – это создание копии файлов, когда база данных включена и доступна пользователям. Был выбран второй вариант реализации.

Также существует способ реализации с помощью средств операционной системы, копируя каждый отдельный файл данных, или при помощи встроенного в установочный пакет oracle компонента Recovery Manager (rman). С целью демонстрации удобства и быстроты разработки был выбран второй вариант реализации.

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.

Включение режима резервного копирования снижает производительность базы данных. В режиме резервного копирования обычно записываются все изменения, происходящие в базе данных, в журналы транзакций. Это может привести к увеличению объема журналов, поскольку они сохраняют информацию о каждой операции. Большой объем журналов может отрицательно сказаться на производительности базы данных, особенно если она активно используется и выполняются множество транзакций.

В случае сбоя во время режима резервного копирования восстановление базы данных может занять больше времени. Это связано с тем, что заголовки файлов данных не обновляются в режиме резервного копирования, и для восстановления базы данных необходимо провести дополнительные шаги, чтобы учесть несохраненные изменения. Это может замедлить процесс восстановления и повлиять на доступность базы данных.

Однако, необходимость включения режима резервного копирования базы данных обусловлена важностью обеспечения сохранности данных. Резервное копирование является важным элементом стратегии обеспечения бизнес-непрерывности и защиты от потери данных.

Для того, чтобы создать резервную копию необходимо подготовить файл со специфичным для данного компонента расширением – «.rman», в котором описан код создания полной копии базы данных с тэгом «fullbackup», листинг 3.1.

|  |
| --- |
| run  {  BACKUP DATABASE  TAG 'fullbackup';  } |

Листинг 5.1 – Содержимое файла full\_backup\_database.rman

Для осуществления восстановления файлов базы данных был разработан файл restore\_database.rman, содержимое которого показано на листинге 4.3.

|  |
| --- |
| startup mount;  run  {  restore database;  recover database;  }  sql 'alter database open'; |

Листинг 5.2 – Содержимое файла restore\_database.rman

Таким образом, было проведено резервное копирование всех файлов базы данных с использованием утилиты Recovery Manager.

## **Вывод к разделу**

В данном разделе была рассмотрена применяемая в курсовом проекте технология, а именно горячее резервное копирование и восстановление данных при помощи компоненты Recovery Manager. Далее рассмотрим реализованное программное средство и руководство пользователя к нему.

# **Заключение**

В процессе решения поставленной задачи была достигнута поставленная цель по созданию базы данных «Книжное издательство». В данной работе использовалось СУБД Oracle DataBase 12c. При разработке курсового проекта использовались объекты: табличное пространство, таблицы, хранимые процедуры, индексы, функции и пользователи.

Основной целью курсового проекта стало проектирование базы данных. При разработке выполнены следующие пункты:

* управление базой данных;
* изменение информации об изданиях, заказах, заказчиках;
* вывод отчетов об изданиях и их авторах и заказах;
* вывод данных об авторах.

Приложение прошло тестирование при использовании в БД большого количество данных.

Также были реализованы процедуры для импорта, экспорта данных в формат XML. Процедуры импорта и экспорта данных в формат XML позволяют обменяться данными между вашей базой данных и другими системами или приложениями, используя широко распространенный формат XML. Это удобное и гибкое решение для обмена информацией.

Была реализована технология «Резервное копирование и восстановление». Данная технология является важной составляющей стратегии обеспечения бизнес-непрерывности и защиты данных. Она обеспечивает сохранность данных и возможность восстановления базы данных в случае сбоев или потери информации.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная база данных работает верно, а требования технического задания выполнены в полном объёме. База данных хорошо разработана и предоставляет пользователю необходимые инструменты для работы с информацией. Они обеспечивают удобство использования и помогают управлять и анализировать данные, что является ключевым в разработке баз данных. Это является хорошим показателем качества разработанной базы данных в целом.

# **Список используемых источников**

1. Официальный сайт Oracle [Электронный ресурс]. Режим доступа: https://www.oracle.com/database/database-vault/index.html. Дата доступа: 01.05.2023.
2. Export and Import In Oracle XML DB Data [Электронный ресурс]. Режим доступа: https://docs.oracle.com/database/121/ADXDB/xdb26imp.htm. Дата доступа: 01.05.2023.
3. Getting Started with RMAN [Электронный ресурс]. Режим доступа: https://docs.oracle.com/cd/E11882\_01/backup.112/e10642/rcmquick.htm#BRADV89346. Дата доступа: 02.05.2023.

# **ПРИЛОЖЕНИЕ А**

CREATE TABLESPACE TSPublishingHouse

Datafile 'D:\app\ora\_install\_user\admin\orcl\TSPublishingHouse.dbf'

size 10M

autoextend ON NEXT 5M

maxsize unlimited;

CREATE TABLE Authors (

AUTHOR\_ID INT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

LAST\_NAME varchar(50) NOT NULL,

BIRTH\_DATE DATE NOT NULL,

COUNTRY varchar(50),

EMAIL VARCHAR(100) NOT NULL

)

TABLESPACE TSPublishingHouse;

COMMIT;

CREATE TABLE Genres (

GENRE\_ID int primary key,

genre\_name VARCHAR(50) NOT NULL, UNIQUE

)

TABLESPACE TSPublishingHouse;

COMMIT;

CREATE TABLE ProductTypes (

product\_type\_id INT PRIMARY KEY,

product\_type\_name VARCHAR(50) NOT NULL -- книга / газета / журнал и тд

)

TABLESPACE TSPublishingHouse;

COMMIT;

CREATE TABLE Typographys (

typography\_id INT PRIMARY KEY,

typography\_name VARCHAR(100) NOT NULL,

address VARCHAR(200),

contact\_number VARCHAR(20)

)

TABLESPACE TSPublishingHouse;

COMMIT;

ALTER TABLE ProductTypes

ADD CONSTRAINT product\_type\_name\_unique UNIQUE (product\_type\_name);

CREATE TABLE Publications ( -- издания / книги

publication\_id INT PRIMARY KEY,

publication\_title VARCHAR(100) NOT NULL,

publication\_author\_id INT,

publication\_genre\_id INT,

publication\_date DATE,

publication\_type\_id INT,

volume\_in\_pages INT, -- кол-во страниц

FOREIGN KEY (publication\_author\_id) REFERENCES Authors(author\_id),

FOREIGN KEY (publication\_genre\_id) REFERENCES Genres(genre\_id),

FOREIGN KEY (publication\_type\_id) REFERENCES ProductTypes(product\_type\_id)

)

TABLESPACE TSPublishingHouse;

COMMIT;

CREATE TABLE CustomerTypes (

customer\_type\_id INT PRIMARY KEY,

customer\_type\_name VARCHAR(50) NOT NULL -- книжный магазин / бизнес-организация /

-- онлайн-платформа / библиотека /

-- организаторы книжных выставок

)

TABLESPACE TSPublishingHouse;

COMMIT;

CREATE TABLE Customers (

customer\_id INT PRIMARY KEY,

customer\_type INT,

customer\_name VARCHAR(100) NOT NULL, -- имя компании / человека

email VARCHAR(100),

country VARCHAR(50),

FOREIGN KEY (customer\_type) REFERENCES CustomerTypes(customer\_type\_id)

)

TABLESPACE TSPublishingHouse;

COMMIT;

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

order\_customer\_id INT,

order\_acceptance\_date DATE, -- дата приема заказа

order\_completion\_date DATE, -- дата выполнения заказа

order\_circulation INT, -- тираж / кол-во заказанного издания

order\_publication\_id INT,

order\_typography\_id INT,

order\_status VARCHAR(20) NOT NULL, -- выполнено, не выполнено, в процессе и тд

FOREIGN KEY (order\_publication\_id) REFERENCES Publications(publication\_id),

FOREIGN KEY (order\_typography\_id) REFERENCES Typographys(typography\_id),

FOREIGN KEY (order\_customer\_id) REFERENCES Customers(customer\_id)

)

TABLESPACE TSPublishingHouse;

COMMIT;

# **ПРИЛОЖЕНИЕ Б**

alter session set "\_ORACLE\_SCRIPT"=true;

CREATE ROLE ADMIN\_ROLE;

grant all on AUTHORS to ADMIN\_ROLE;

GRANT ALL ON CUSTOMERS TO ADMIN\_ROLE;

grant all on CUSTOMERTYPES to ADMIN\_ROLE;

grant all on GENRES to ADMIN\_ROLE;

GRANT ALL ON ORDERS TO ADMIN\_ROLE;

GRANT ALL ON PRODUCTTYPES TO ADMIN\_ROLE;

GRANT ALL ON PUBLICATIONS TO ADMIN\_ROLE;

GRANT ALL ON TYPOGRAPHYS TO ADMIN\_ROLE;

grant create any sequence to ADMIN\_ROLE with admin option;

grant create session to ADMIN\_ROLE with admin option;

grant create table to ADMIN\_ROLE with admin option;

grant create any sequence to ADMIN\_ROLE with admin option;

grant create user to ADMIN\_ROLE with admin option;

grant drop user to ADMIN\_ROLE with admin option;

grant insert any table to ADMIN\_ROLE with admin option;

grant update any table to ADMIN\_ROLE with admin option;

grant delete any table to ADMIN\_ROLE with admin option;

grant select on dba\_source to ADMIN\_ROLE;

grant select on dba\_objects to ADMIN\_ROLE;

GRANT SELECT ON DBA\_INDEXES TO ADMIN\_ROLE;

CREATE ROLE CLIENT\_ROLE;

GRANT SELECT ON ORDERS\_VIEW TO CUSTOMER\_ROLE;

GRANT SELECT ON PUBLICATIONS\_VIEW TO CUSTOMER\_ROLE;

grant select on TYPOGRAPHYS to CUSTOMER\_ROLE;

GRANT CREATE SESSION TO CUSTOMER\_ROLE;

# **ПРИЛОЖЕНИЕ В**

CREATE OR REPLACE PROCEDURE DELETE\_PRODUCT\_TYPE(P\_NAME IN VARCHAR2)

AS

BEGIN

delete from PRODUCTTYPES

WHERE PRODUCT\_TYPE\_NAME = P\_NAME;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

rollback;

DBMS\_OUTPUT.PUT\_LINE('Error deleting product type: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE UPDATE\_PRODUCT\_TYPE(OLD\_NAME IN VARCHAR2, NEW\_NAME IN VARCHAR2)

AS

BEGIN

UPDATE PRODUCTTYPES

SET PRODUCT\_TYPE\_NAME = NEW\_NAME

WHERE PRODUCT\_TYPE\_NAME = OLD\_NAME;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error updating product type: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE ADD\_PRODUCT\_TYPE(p\_name IN VARCHAR2)

AS

v\_id NUMBER;

v\_count NUMBER;

BEGIN

LOOP

-- Генерация случайного пятизначного ID

v\_id := TRUNC(DBMS\_RANDOM.VALUE(10000, 99999));

-- Проверка на уникальность ID

SELECT COUNT(\*) INTO v\_count

FROM PRODUCTTYPES

WHERE PRODUCT\_TYPE\_ID = v\_id;

EXIT WHEN v\_count = 0;

END LOOP;

INSERT INTO PRODUCTTYPES

VALUES (v\_id, p\_name);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error adding product type: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE DELETE\_CUSTOMER\_TYPE(P\_NAME IN VARCHAR2)

AS

BEGIN

delete from CUSTOMERTYPES

WHERE CUSTOMER\_TYPE\_NAME = P\_NAME;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

rollback;

DBMS\_OUTPUT.PUT\_LINE('Error deleting customer type: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE ADD\_CUSTOMER\_TYPE(p\_name IN VARCHAR2)

AS

v\_id NUMBER;

v\_count NUMBER;

BEGIN

LOOP

-- Генерация случайного пятизначного ID

v\_id := TRUNC(DBMS\_RANDOM.VALUE(10000, 99999));

-- Проверка на уникальность ID

SELECT COUNT(\*) INTO v\_count

FROM CUSTOMERTYPES

WHERE CUSTOMER\_TYPE\_ID = v\_id;

EXIT WHEN v\_count = 0;

END LOOP;

INSERT INTO CUSTOMERTYPES

VALUES (v\_id, p\_name);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error adding customer type: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE UPDATE\_CUSTOMER\_TYPE(OLD\_NAME IN VARCHAR2, NEW\_NAME IN VARCHAR2)

AS

BEGIN

UPDATE CUSTOMERTYPES

SET CUSTOMER\_TYPE\_NAME = NEW\_NAME

WHERE CUSTOMER\_TYPE\_NAME = OLD\_NAME;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error updating customer type: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE ADD\_AUTHOR(P\_NAME IN VARCHAR2,

P\_SURNAME IN VARCHAR2, P\_BIRTHDAY IN VARCHAR2,

p\_country in varchar2, p\_email IN VARCHAR2)

AS

v\_id NUMBER;

V\_COUNT NUMBER;

V\_BIRTHDAY DATE; -- Дата

BEGIN

LOOP

-- Генерация случайного пятизначного ID

v\_id := TRUNC(DBMS\_RANDOM.VALUE(10000, 99999));

-- Проверка на уникальность ID

select count(\*) into v\_count

from AUTHORS

WHERE author\_id = v\_id;

-- Если ID уже существует, перегенерировать и повторить проверку

EXIT WHEN V\_COUNT = 0;

END LOOP;

-- Преобразование строки в дату

V\_BIRTHDAY := TO\_DATE(P\_BIRTHDAY, 'DD-MM-YYYY');

INSERT INTO AUTHORS

VALUES (v\_id, p\_name, p\_surname, V\_BIRTHDAY, p\_country, p\_email);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

rollback;

DBMS\_OUTPUT.PUT\_LINE('Error adding author: ' || SQLERRM);

END;

create or replace procedure UPDATE\_AUTHOR(

P\_ID IN VARCHAR2, NEW\_NAME IN VARCHAR2,

NEW\_SURNAME in varchar2, NEW\_BIRTHDAY in VARCHAR2,

NEW\_COUNTRY in varchar2, NEW\_EMAIL in varchar2)

AS

V\_BIRTHDAY DATE; -- Дата

BEGIN

-- Преобразование строки в дату

V\_BIRTHDAY := TO\_DATE(NEW\_BIRTHDAY, 'DD-MM-YYYY');

UPDATE AUTHORS

set FIRST\_NAME = NEW\_NAME,

LAST\_NAME = NEW\_SURNAME,

BIRTH\_DATE = V\_BIRTHDAY,

COUNTRY = NEW\_COUNTRY,

EMAIL = NEW\_EMAIL

WHERE AUTHOR\_ID = P\_ID;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error updating author: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE DELETE\_AUTHOR(P\_ID IN NUMBER)

AS

BEGIN

DELETE FROM AUTHORS

WHERE AUTHOR\_ID = P\_ID;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error deliting author: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE ADD\_TYPOGRAPHY(p\_name IN VARCHAR2, p\_address IN VARCHAR2, p\_contact\_name IN VARCHAR2)

AS

v\_id NUMBER;

v\_count NUMBER;

BEGIN

LOOP

-- Генерация случайного пятизначного ID

v\_id := TRUNC(DBMS\_RANDOM.VALUE(10000, 99999));

-- Проверка на уникальность ID

SELECT COUNT(\*) INTO V\_COUNT

FROM TYPOGRAPHYS

WHERE TYPOGRAPHY\_ID = v\_id;

-- Если ID уже существует, перегенерировать и повторить проверку

EXIT WHEN v\_count = 0;

END LOOP;

-- Вставка нового жанра

INSERT INTO TYPOGRAPHYS

VALUES (v\_id, p\_name, p\_address, p\_contact\_name);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error adding typography: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE UPDATE\_TYPOGRAPHY(OLD\_NAME IN VARCHAR2, NEW\_NAME IN VARCHAR2,

NEW\_ADDRESS IN VARCHAR2, NEW\_CONTACT\_NAME IN VARCHAR2)

AS

BEGIN

UPDATE TYPOGRAPHYS

SET TYPOGRAPHY\_NAME = NEW\_NAME, ADDRESS = NEW\_ADDRESS, CONTACT\_NUMBER = NEW\_CONTACT\_NAME

WHERE TYPOGRAPHY\_NAME = OLD\_NAME;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error updating typography: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE DELETE\_TYPOGRAPHY(P\_NAME IN VARCHAR2)

AS

BEGIN

DELETE FROM TYPOGRAPHYS

WHERE TYPOGRAPHY\_NAME = P\_NAME;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error deliting typography: ' || SQLERRM);

END;

# **ПРИЛОЖЕНИЕ Г**

create or replace

FUNCTION GET\_ALL\_GENRES

RETURN SYS.ODCIVARCHAR2LIST

AS

genre\_array SYS.ODCIVARCHAR2LIST := SYS.ODCIVARCHAR2LIST(); -- Создание пустого массива

BEGIN

FOR genre\_rec IN (SELECT GENRE\_ID, GENRE\_NAME FROM GENRES) LOOP

genre\_array.EXTEND;

genre\_array(genre\_array.LAST) := genre\_rec.GENRE\_ID || ',' || genre\_rec.GENRE\_NAME; -- Соединяем ID и Name в формате "ID,Name"

END LOOP;

RETURN genre\_array;

END;

create or replace

FUNCTION GET\_ALL\_CUSTOMERS

RETURN SYS.ODCIVARCHAR2LIST

AS

customer\_array SYS.ODCIVARCHAR2LIST := SYS.ODCIVARCHAR2LIST(); -- Создание пустого массива

BEGIN

FOR CUSTOMER\_REC IN (

SELECT CUSTOMERS.CUSTOMER\_ID,

CUSTOMERTYPES.CUSTOMER\_TYPE\_NAME,

CUSTOMERS.CUSTOMER\_NAME,

CUSTOMERS.COUNTRY,

CUSTOMERS.EMAIL

FROM CUSTOMERS INNER JOIN CUSTOMERTYPES

on CUSTOMERTYPES.CUSTOMER\_TYPE\_ID = CUSTOMERS.CUSTOMER\_ID

) LOOP

CUSTOMER\_ARRAY.EXTEND;

CUSTOMER\_ARRAY(CUSTOMER\_ARRAY.LAST) := CUSTOMER\_REC.CUSTOMER\_ID || ',' ||

CUSTOMER\_REC.CUSTOMER\_TYPE\_NAME || ',' || CUSTOMER\_REC.CUSTOMER\_NAME || ',' ||

CUSTOMER\_REC.COUNTRY || ',' || CUSTOMER\_REC.EMAIL;

END LOOP;

RETURN customer\_array;

END;