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 1-40 05 01-03 Информационные системы и технологии

(издательско-полиграфический комплекс)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Реализация базы данных «Магазин электроники» с использованием технологии применения мультимедийных типов данных в БД»

Выполнил студент Парибок Илья Александрович

(Ф.И.О.)

Руководитель проекта асс. Сазонова Д. В.

(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: асс. Сазонова Д. В

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

Нормоконтролер: асс. Сазонова Д. В

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

Курсовой проект защищен с оценкой

Минск 2023

**Оглавление**

[Введение 3](#_Toc134777569)

[1 Постановка задачи 4](#_Toc134777570)

[2 Разработка модели базы данных 5](#_Toc134777571)

[3 Разработка необходимых объектов 11](#_Toc134777572)

[3.1 Таблицы 11](#_Toc134777573)

[3.2 Процедуры 11](#_Toc134777574)

[4 Описание процедур импорта и экспорта 14](#_Toc134777575)

[5 Тестирование производительности 16](#_Toc134777576)

[6 Описание технологии 18](#_Toc134777577)

[6.1 Технология «Мультимедийные типы данных в БД» 18](#_Toc134777578)

[6.2 Применение технологии 18](#_Toc134777579)

[7 Описание приложения 20](#_Toc134777580)

[8 Руководство пользователя 22](#_Toc134777581)

[Заключение 24](#_Toc134777582)

[Список используемой литературы 25](#_Toc134777583)

[Приложение А 26](#_Toc134777584)

[Приложение Б 30](#_Toc134777585)

# Введение

Курсовая работа по администрированию баз данных и приложений посвящена реализации базы данных «Магазин электроники» на основе технологии применения мультимедийных типов данных в БД. Важность такой базы данных обусловлена ее применением в современном мире, где электроника играет все более важную роль в повседневной жизни. В настоящее время многие магазины электроники используют в своей работе различные базы данных, что значительно упрощает управление заказами, отслеживание запасов, наполнение интернет-магазинов и управление кадровым составом.

Реализация данной работы поможет совершенствованию и автоматизации этих процессов, а также даст возможность изучить и применить в практике технологию баз данных на конкретном примере. База данных будет реализована в СУБД Oracle 12c, и будет содержать данные о товарах, складах, заказах, клиентах и оставленных отзывах.

База данных используется для хранения информации о производимой или продаваемой продукции. Она включает в себя детализированное описание продукта, его технические характеристики, справочный номер, цену, доступность. Также в базе данных хранятся все клиенты компании и её сотрудники. Большое количество малых бизнесов имеет свой сайт и базу данных, содержащую информацию о клиентской базе и широком спектре товаров и услуг.

Таким образом, в настоящее время большинство бизнес-моделей используют базы данных, как неотъемлемый и облегчающий жизнь инструмент.

1. Постановка задачи

Разработать структуру базы данных, обеспечивающую управление запасами товаров в магазине, включая добавление и удаление товаров со склада. Для этого нужно создать таблицу, содержащую информацию о типе (категории) товара, его стоимости, свободном числе и месте на складе. В таблице должны быть соответствующие внешние ключи для связи с таблицами «заказчики», «сотрудники» и другими.

Реализовать управление кадрами магазина, включая добавление, удаление и изменение характеристик сотрудников. Для этого должна быть создана таблица, содержащая информацию о каждом сотруднике, его имени, фамилии, отчестве, должности, зарплате и т.д. В таблице должны быть соответствующие внешние ключи для связи с таблицами «заказы», «товары» и другими.

Реализовать управление заказами, включая изменение этапов выполнения заказов, изменение статусов заказов и добавление даты завершения заказов. Для этого должна быть создана таблица, содержащая информацию о каждом заказе, его номере, дате и статусе, а также информацию о заказчике и товарах.

Реализовать процесс оформления заказа, включая добавление товаров в корзину и покупку товаров клиентом. Для этого нужно создать таблицу, где пользователь сможет добавлять товары, а также таблицу, содержащую информацию о каждой продаже, ее номере, дате и связанную с ней информацию.

Разработать возможность для клиентов создавать аккаунт и управлять им, включая добавление и изменение имени, фамилии.

Также необходимо протестировать производительность базы данных на таблицах, содержащих не менее 100 000 строк.

1. Разработка модели базы данных

Для создания электронной коммерции необходима надежная база данных, которая оптимизирована для хранения, организации и извлечения информации о товарах, клиентах и заказах. Представленная схема базы данных Oracle обладает всеми необходимыми компонентами для управления эффективными онлайн-магазинами. Схема БД представлена на рисунке 1.

Схема состоит из нескольких таблиц: Orders, Customers, Products, Categories, Cart, Sales, Employees и Reviews. Каждая из таблиц содержит набор полей, которые сохраняют информацию о соответствующих объектах, таких как заказы, клиенты, продукты, категории, продажи, работники и отзывы. Кроме того, таблицы направлены на удовлетворение основных операций электронной коммерции, таких как оформление заказа, покупка, отзывы, рейтинги и управление продуктами и покупателями.

Таким образом, использование этой базы данных Oracle может обеспечить стабильность и эффективность работы онлайн-магазина и помочь максимизировать доходы и удобство для покупателей.

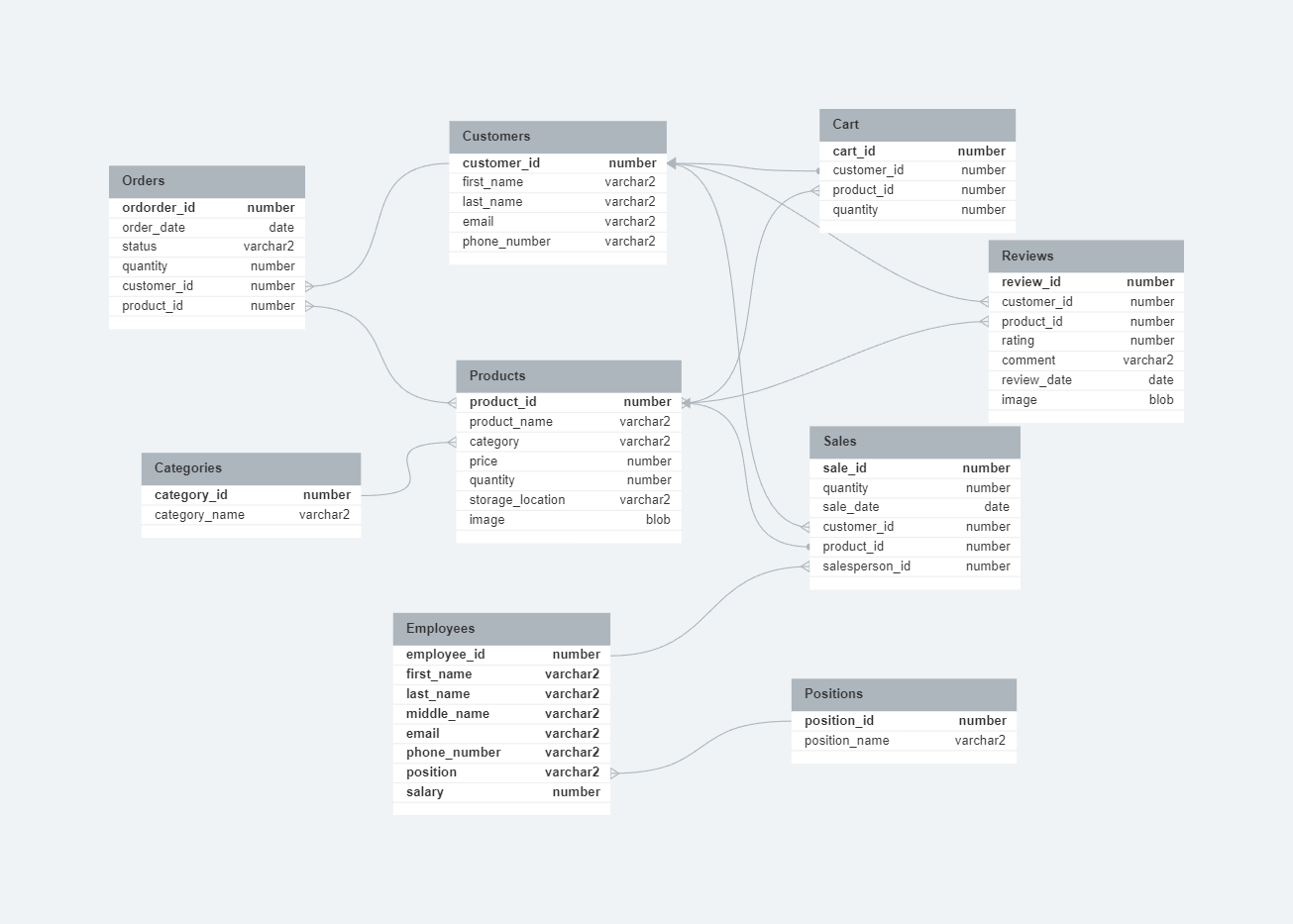


Рисунок 1 – Схема базы данных

Таблица Orders содержит информацию о заказах покупателей. Каждая запись таблицы представляет один заказ и содержит информацию о номере заказа, дате размещения заказа, статусе заказа и связанных с ним клиенте и продукте.

Таблица Orders связана с таблицами Customers и Products через поля customer\_id и product\_id соответственно. Связь между таблицей Orders и каждой из этих таблиц является «многие к одному». Таким образом, в таблице Orders хранится информация о том, какой клиент сделал какой заказ, а также какой продукт был заказан. Через связи с таблицами Customers и Products также можно получить более подробную информацию о клиенте и продукте. Описание её полей представлено в таблице 1.

Таблица 1 – Поля таблицы «Orders»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип данных |
| order\_id | Идентификатор заказа | number |
| order\_date | Дата заказа | date |
| status | Статус заказа | varchar2 |
| customer\_id | Идентификатор покупателя | number |
| product\_id | Идентификатор продукта | number |
| quantity | Количество продукта | number |

Таблица Customers содержит информацию о покупателях. Она хранит данные о каждом клиенте, в том числе их имя, фамилию, адрес электронной почты и номер телефона. Кроме того, в таблице также хранятся идентификаторы заказов, которые клиент разместил, а также их корзины и отзывы, связанные с продуктами, которые они приобрели.

Таблица Customers связана с таблицами Orders, Cart, Sales, Reviews через поле customer\_id в каждой из этих таблиц. Связь между таблицей Customers и каждой из этих таблиц является «один ко многим». С помощью этих связей можно определить, какие заказы были размещены конкретным клиентом, какие продукты были добавлены в корзину, какие отзывы были оставлены о продуктах клиентом и т.д. Описание её полей представлено в таблице 2.

Таблица 2 – Поля таблицы «Customers»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип данных |
| customer\_id | Идентификатор покупателя | number |
| first\_name | Имя покупателя | varchar2 |
| last\_name | Фамилия покупателя | varchar2 |
| email | Адрес электронной почты покупателя | varchar2 |
| Phonenumber | Номер телефона | varchar2 |

Таблица Products содержит информацию обо всех продуктах, предлагаемых в онлайн-магазине. Она хранит информацию о наименовании продукта, категории к которой он относится, цене, количестве товаров на складе и местоположении, а также об изображении продукта.

Таблица Products связана с таблицами Orders, Cart, Sales и Reviews через поле product\_id в каждой из этих таблиц. Связь между таблицей Products и каждой из этих таблиц является «один ко многим», так как каждый продукт может быть частью множества заказов и отзывов. Также таблица Products связана с таблицей Categories через поле category, где категория продукта сохраняется как уникальный идентификатор категории. Это позволяет легко искать продукты в таблице Products, относящиеся к определенной категории, и управлять списком продуктов в пределах конкретной категории. Описание её полей представлено в таблице 3.

Таблица 3 – Поля таблицы «Products»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип данных |
| product\_id | Идентификатор продукта | number |
| product\_name | Название продукта | varchar2 |
| category | Категория продукта | varchar2 |
| price | Цена продукта | number |
| quantity | Количество продукта | number |
| storage\_location | Место хранения продукта | varchar2 |
| image | Изображение продукта (в формате BLOB) | blob |

Таблица Categories содержит список всех категорий продуктов, которые предлагаются в онлайн-магазине. Каждая запись представляет одну категорию и имеет уникальный идентификатор category\_id. На данной табллице также будет тестироваться производительность.

Таблица Categories связана с таблицей Products через поле category, где категория продукта сохраняется как уникальный идентификатор категории. Таким образом, с помощью этой связи можно быстро найти все продукты, относящиеся к конкретной категории. В таблице Categories также есть поле category\_name, которое хранит название каждой категории. Данное поле является уникальным, что позволяет избежать дублей категорий. Описание её полей представлено в таблице 4.

Таблица 4 – Поля таблицы «Categories»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип данных |
| category\_id | Идентификатор категории продукта | number |
| category\_name | Название категории продукта | varchar2 |

Таблица Cart содержит информацию о продуктах, которые были добавлены в корзину покупателя. Каждая запись представляет один продукт в корзине и содержит информацию о количестве продуктов и связанных с ним покупателе и продукте.

Таблица Cart связана с таблицами Customers и Products через поля customer\_id и product\_id соответственно. Связь между этими таблицами является «многие к одному», так как каждый покупатель может добавить множество продуктов в свою корзину. Таблица Cart также связана с таблицей Orders. Когда покупатель размещает заказ с товарами из своей корзины, Order создает новую запись о заказе со связкой на каждый добавленный продукт. Таким образом, Cart позволяет журналировать продукты, добавленные пользователями в свои корзины, что полезно для учета продаж и управления складским запасом. Описание её полей представлено в таблице 5.

Таблица 5 – Поля таблицы «Cart»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип данных |
| cart\_id | Идентификатор корзины | number |
| customer\_id | Идентификатор покупателя | number |
| product\_id | Идентификатор продукта | number |
| quantity | Количество продукта в корзине | number |

Таблица Sales содержит информацию о продажах в онлайн-магазине. Она хранит информацию о каждой сделанной продаже, такую как количество проданного товара, дату и время продажи, а также связанных с ней клиенте, продукте и сотруднике, ответственном за продажу.

Таблица Sales связана с таблицами Customers, Products и Employees через поля customer\_id, product\_id и salesperson\_id, соответственно. Связи между этими таблицами также являются «многие к одному», так как каждая продажа относится к одному продукту и одному покупателю, а также к одному продавцу. С помощью таблицы Sales администратор может получить информацию об уровне продаж, а также о том, какие продукты пользуются наибольшим спросом у клиентов. Описание её полей представлено в шестой таблице.

Таблица 6 – Поля таблицы «Sales»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип данных |
| sale\_id | Идентификатор продажи | number |
| quantity | Количество продукта, проданного за одну сделку | number |
| sale\_date | Дата продажи | date |
| customer\_id | Идентификатор покупателя | number |
| product\_id | Идентификатор продукта | number |
| salesperson\_id | Идентификатор продавца | number |

Таблица Employees содержит информацию о сотрудниках онлайн-магазина. Она хранит информацию о каждом сотруднике, включая имя, фамилию, адрес электронной почты, номер телефона, должность, зарплату и связанный с ним идентификатор.

Таблица Employees связана с таблицей Positions через поле position\_id. Связь между этими таблицами является «многие к одному», так как каждый сотрудник может быть описан определенной позицией, а каждая позиция может быть выполнена множеством сотрудников.

Вдобавок таблица Employees связана с таблицей Sales через поле salesperson\_id. Связь между этими таблицами также является «многие к одному», поскольку каждая продажа может быть связана с ответственным за нее сотрудником. Используя таблицу Employees, администратор онлайн-магазина может получить информацию об уровне эффективности и производительности сотрудников, а также они могут использовать данные для принятия решений о том, какие навыки необходимы для определенных позиций, и о том, какие дополнительные возможности нужно предоставить сотрудникам для повышения производительности и удовлетворенности. Описание её полей представлено в таблице 7.

Таблица 7 – Поля таблицы «Employees»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип данных |
| employee\_id | Идентификатор сотрудника | number |
| first\_name | Имя сотрудника | varchar2 |
| last\_name | Фамилия сотрудника | varchar2 |
| middle\_name | Отчество сотрудника (если имеется) | varchar2 |
| email | Адрес электронной почты сотрудника | varchar2 |
| phone\_number | Номер телефона сотрудника | varchar2 |
| position | Должность сотрудника | varchar2 |
| salary | Зарплата сотрудник | number |

Таблица Positions содержит список должностей, которые могут быть занимаемыми сотрудниками в онлайн-магазине. Каждая запись представляет одну должность и имеет уникальный идентификатор position\_id и название позиции в поле position\_name.

Таблица Positions связана с таблицей Employees через поле position\_id. Связь между этими таблицами является "один ко многим", поскольку каждая позиция может быть связана с несколькими сотрудниками. Используя таблицу Positions, администратор может получить информацию о том, какие позиции доступны, кто занимает каждую позицию и на какой зарплате. Эта информация может быть использована для управления. Описание её полей представлено в таблице 8.

Таблица 8 – Поля таблицы «Positions»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип данных |
| position\_id | Идентификатор должности | number |
| position\_name | Название должности | varchar2 |

Таблица Reviews содержит отзывы покупателей об определенных продуктах, которые они приобрели в онлайн-магазине. Каждая запись представляет собой один отзыв и содержит информацию о продукте, связанном с этим отзывом, а также имя, текст и оценку отзыва.

Таблица Reviews связана с таблицами Customers и Products через поля customer\_id и product\_id соответственно. Связь между таблицей Reviews и каждой из этих таблиц является «многие к одному», так как каждый отзыв относится к конкретному продукту и конкретному покупателю. Через связи с таблицами Customers и Products также можно получить более подробную информацию о клиенте и продукте. Используя таблицу Reviews, администратор может получить обратную связь от клиентов и использовать эту информацию для улучшения качества своего сервиса и продуктов. Так же пользователи могут оставить в фото в отзыве, для более полной обратной связи.

Таблица 9 – Поля таблицы «Reviews»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип данных |
| review\_id | Идентификатор отзыва | number |
| customer\_id | Идентификатор покупателя, оставившего отзыв | number |
| product\_id | Идентификатор продукта, на который оставлен отзыв | number |
| rating | Рейтинг продукта (от 1 до 5) | number |
| comment | Комментарий к отзыву (если имеется) | varchar2 |
| review\_date | Дата оставления отзыва | date |
| image | Изображение (в формате BLOB) прикрепленное к отзыву | blob |

Каждая таблица в базе данных онлайн-магазина имеет определенную функцию и связывается с другими таблицами в базе данных. Например, таблица Orders связана с таблицами Customers и Products, а таблица Cart связана с таблицами Customers и Products, чтобы отслеживать заказы и состояние продуктов. Таблицы Product и Categories используются для хранения информации о продуктах и категориях, соответственно. Таблица Employees содержит информацию о сотрудниках, а таблица Positions обеспечивает сопоставление позиций сотрудников. Наконец, таблицы Sales и Reviews используются для учета продаж и обратной связи от клиентов.

Каждая таблица в этой базе данных связана с другими таблицами, что позволяет управлять заказами, отслеживать их статус и связывать заказы с конкретными продуктами и клиентами. База данных помогает упростить процесс управления продажами и предоставляет администратору больше информации для принятия решений. Код создание таблиц представлен в приложение А.

1. **Разработка** необходимых объектов

3.1 Таблицы

В любой базе данных главным элементом является набор таблиц, которые связаны между собой. Таблицы отвечают за структурирование и ограничение хранимой информации, а также обеспечивают связность между элементами данных с помощью внешних ключей.

Перед созданием таблиц необходимо выделить наиболее важные сущности, которые отражают основную информацию, хранимую в базе данных. Основные таблицы в базе данных это:

* Заказы (Orders)
* Клиенты (Customers)
* Продукты (Products)

Это таблицы, которые прямо участвуют в процессе заказа. Остальные таблицы, такие как Категории (Categories), Корзина (Cart), Продажи (Sales), Сотрудники (Employees), Должности (Positions) и Отзывы (Reviews), являются вспомогательными таблицами, которые предоставляют дополнительную информацию для основных таблиц. В листинге 3.1 представлен код создание одной из основных таблицы «Продукты». Приложение А представлен код для создание всех таблицу бд.

CREATE TABLE "Products" (

"product\_id" NUMBER NOT NULL,

"product\_name" VARCHAR2(255) NOT NULL,

"category" VARCHAR2(255) NOT NULL,

"price" NUMBER NOT NULL,

"quantity" NUMBER NOT NULL,

"storage\_location" VARCHAR2(255) NOT NULL,

"image" BLOB NOT NULL,

constraint PRODUCTS\_PK PRIMARY KEY ("product\_id")

);

Листинг 3.1 – Создание таблицы Products

3.2 Процедуры

Для обеспечения взаимодействия пользователя с данными, хранящимися в базе данных, используются хранимые процедуры на языке T-SQL. Операции, выполняемые процедурами, делятся на CRUD-операции и операции, специфичные для конкретной предметной области базы данных. В том числе процедуры позволяют пользователю проходить тесты, регистрироваться и авторизовываться. Процедуры были разработаны для обработки сложных отношений между данными в базе и решения задачи соблюдения связей между данными и обработки ситуаций, связанных с этими связями. Они прошли успешное тестирование, и в приложении к работе готовы. Для импорта данных используются процедура importXml. Код основных процедур представлен в приложение Б. Большинство процедур представляют собой CRUD-операции.

Процедуры для создания, изменения, удаления:

* ADDCATEGORY – добавляет новую категорию товаров в базу данных.
* ADDCUSTOMER – добавляет нового клиента в базу данных.
* ADDEMPLOYEE – добавляет нового сотрудника в базу данных.
* ADDORDER – добавляет новый заказ в базу данных.
* ADDPOSITION – добавляет новую позицию в заказ в базу данных.
* ADDPRODUCT – добавляет новый продукт в базу данных.
* ADDREVIEW – добавляет новый отзыв о продукте в базу данных.
* ADDTOCART – добавляет товар в корзину клиента.
* COMPLETEORDER – завершает заказ и переносит его из корзины в раздел "Завершенные заказы".
* DELETECARTITEMBYID – удаляет товар из корзины по заданному ID.
* DELETECATEGORY – удаляет категорию товаров из базы данных.
* DELETECUSTOMER – удаляет клиента из базы данных.
* DELETEEMPLOYEE – удаляет сотрудника из базы данных.
* DELETEFROMCART – удаляет все товары из корзины клиента.
* DELETEORDERBYID – удаляет заказ из базы данных по заданному ID.
* DELETEPOSITION – удаляет позицию в заказе из базы данных.
* DELETEPRODUCTBYID – удаляет продукт из базы данных по заданному ID.
* DELETEREVIEWBYID – удаляет отзыв о продукте из базы данных по заданному ID.
* EDITCART – изменяет количество товаров в корзине клиента.
* EDITCATEGORYNAME – изменяет название категории товаров.
* GETALLCATEGORIES – получает список всех категорий товаров из базы данных.
* GETALLCUSTOMERS – получает список всех клиентов из базы данных.
* GETALLEMPLOYEES – получает список всех сотрудников из базы данных.
* GETALLORDERS – получает список всех заказов из базы данных.
* GETALLPOSITIONS – получает список всех позиций в заказах из базы данных.
* GETALLPRODUCTS – получает список всех продуктов из базы данных.
* GETCARTINFOBYCUSTOMERID – получает информацию о корзине клиента по заданному ID клиента.
* GETCATEGORYBYID – получает категорию товаров из базы данных по заданному ID.
* GETCUSTOMER – получает информацию о клиенте по заданному ID.
* GETORDERINFOBYORDERID – получает информацию о заказе по заданному ID заказа.
* GETORDERSBYCUSTOMERID – получает список заказов клиента по заданному ID клиента.
* GETPOSITIONBYID – получает позицию в заказе по заданному ID.
* GETPRODUCTBYID – получает продукт из базы данных по заданному ID.
* GETREVIEWSBYCUSTOMERID – получает список отзывов о продуктах, оставленных клиентом по заданному ID клиента.
* GETREVIEWSBYPRODUCTID – получает список отзывов о продукте по заданному ID продукта.
* GETSALESBYCUSTOMERID – получает все заказы и продажи клиента по заданному ID клиента.
* GETSALESBYPRODUCTID – получает все заказы и продажи продукта по заданному ID продукта.
* INSERTPRODUCT – вставляет новый продукт в базу данных.
* MOVECARTTOORDERS – перемещает товары из корзины в список заказов.
* UPDATECUSTOMER – обновляет информацию о клиенте в базе данных.
* UPDATEEMPLOYEEBYID – обновляет информацию о сотруднике в базе данных по заданному ID.
* UPDATEORDERSTATUS – обновляет статус заказа в базе данных.
* UPDATEPOSITION – обновляет информацию о позиции в заказе в базе данных.
* UPDATEPRODUCT – обновляет информацию о продукте в базе данных.

1. Описание процедур импорта и экспорта

Один из этапов разработки инфраструктуры для базы данных был связан с созданием процедур для экспорта и импорта данных в формате XML. Чтобы импортировать данные из XML-файла, нужно выполнить SQL-запрос, представленный в листинге 4.1. Данная процедура «importXml» импортирует данные из XML-файла в таблицу «Categories» в схеме «PIA».

Процедура использует цикл «for p in (...) loop», который позволяет перебрать каждую извлеченную строку XML-данных по порядку. После этого, используя «ExtractValue», процедура извлекает значение «category\_id» и «category\_name» для каждой строки данных.

Затем используется оператор INSERT INTO, чтобы добавить полученные значения «id» и «name» в таблицу «Categories».

create or replace NONEDITIONABLE PROCEDURE importXml IS

BEGIN

for p in (SELECT ExtractValue(VALUE(row\_xml), '//category\_id') AS id, ExtractValue(VALUE(row\_xml), '//category\_name') AS

name

FROM TABLE(XMLSequence(EXTRACT(XMLTYPE(bfilename('IMAGE\_BASE', 'test.xml'),

nls\_charset\_id('UTF-8')),'/ROWSET/ROW'))) row\_xml)

loop

INSERT INTO PIA."Categories" ("category\_id", "category\_name") VALUES (p.id, p.name);

end loop;

commit;

END;

Листинг 4.1 – Процедура импорта данных из xml

Чтобы экспортировать данные из базы данных, можно использовать встроенное средство "Export Wizard", доступное в пакете SQL Developer. Для этого нужно выбрать таблицу, которую необходимо экспортировать, указать путь к месту сохранения файла и формат XML. Пример экспорта данных показан на рисунке 4.1.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 4.1 – Export Wizzard

Так же для экспорта можно воспользоваться процедурой «exportXml». Указав в функции «WRITETOFILE» путь сохранения и xml файл созданный из таблицы.

create or replace NONEDITIONABLE PROCEDURE exportXml

IS

rc sys\_refcursor;

doc DBMS\_XMLDOM.DOMDocument;

BEGIN

OPEN rc FOR SELECT "category\_id", "category\_name" FROM PIA."Categories";

doc := DBMS\_XMLDOM.NewDOMDocument(XMLTYPE(rc));

DBMS\_XMLDOM.WRITETOFILE(doc, 'XT\_DIR/test.xml');

END;

Листинг 4.2 – Процедура экспорта данных в xml

Экспорт и импорт данных в базе данных Oracle являются важной частью её управления. Они позволяют сохранить данные в базе данных или переместить их в другую базу данных. Процедуры экспорта и импорта, обеспечивают эффективный способ выполнения этих операций. Они могут быть повторно использованы для автоматизации процесса и поддержания консистентности данных.

1. Тестирование производительности

Производительность базы данных – это важный аспект приложений, работающих на базах данных. Хорошо спроектированная база данных способствует более быстрой и эффективной работе приложений.

В базах данных Oracle существуют различные способы улучшения производительности. Некоторые из них могут включать создание индексов, проектирование правильной структуры данных, оптимизацию процедур запроса и конфигурирование параметров базы данных.

Один из способов улучшения производительности базы данных Oracle — это создание индексов. Индексы позволяют быстро находить и выбирать данные определенных значений в столбце. Более эффективно всего использование индексов на столбцах, которые широко используются в таблице.

Для проведения тестирования производительности я заполнил таблицу «Categories» с помощью процедуры «fill\_categories\_table», которая содержится в листинге 5.1. Данная процедура добавляет 100 000 записей в таблицу. Это было сделано с целью проверить производительность базы данных и ее способность обрабатывать большой объем данных.

create or replace NONEDITIONABLE PROCEDURE fill\_categories\_table

AS BEGIN FOR i IN 1..100000 LOOP

INSERT INTO PIA."Categories" ("category\_name","category\_id") VALUES ('Category ' || i, i);

END LOOP; COMMIT; END;

Листинг 5.1 – Процедура заполнения таблицы Categories 100 000 записей

Для анализа запросов в SQL Developer можно использовать модуль «Explain Plan». Этот модуль показывает и оценивает план выполнения запроса. В данном случае запрос фильтрует строки таблицы по содержанию столбца «category\_name» и четности значений столбца «category\_id». Он представлен на рисунке 5.1.

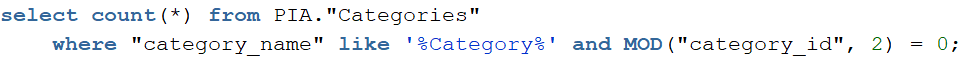


Рисунок 5.1 – Оцениваемый запрос

Исходя из плана выполнения запроса (рис. 5.2), выбираются все строки из таблицы *Categories*, где имя категории содержит слово «Category» и ID категории является четным числом.

Данный запрос выбирает все строки из таблицы «Categories», где имя категории содержит слово «Category» и "ID" категории является четным числом.

В плане выполнения запроса использовано полное сканирование двух индексов (fast full scan index) и хэш-соединение (hash join) между результатами поиска по каждому индексу. Затем выполняется сортировка (sort) и подсчет общего количества строк (aggregate) в результате запроса.

Важно отметить, что Oracle создает индексы автоматически, чтобы ускорить выполнение запросов, и может использовать существующие индексы для выполнения запроса. В данном случае использование индексов позволило оптимизировать выполнение запроса.

Из стоимости каждой операции видно, что сканирование индексов было наиболее дорогостоящим этапом. В целом запрос был оптимизирован и выполнен эффективно, но возможно есть пути его дальнейшей оптимизации. Например, можно оптимизировать запрос, чтобы использовать только один индекс для поиска строк, удовлетворяющих обоим условиям, вместо двух индексов. Также можно использовать параметр поиска для ограничения результатов только категориями, созданными в определенный временной период.

Кроме того, для улучшения производительности базы данных, необходимо регулярно выполнять анализ производительности и оптимизацию параметров базы данных и таблиц, а также возможную регулярную очистку неиспользуемых данных.

Изображение выглядит как текст, линия, число, Шрифт

Автоматически созданное описание

Рисунок 5.2 – Оценка запроса

В целом, оптимизация запроса и базы данных в целом являются важными аспектами, которые позволяют повысить эффективность работы базы данных и приложений.

1. Описание технологии
   1. Технология «Мультимедийные типы данных в БД»

Технология «Мультимедийные типы данных в БД» – это возможность хранения, обработки и управления мультимедийными данными (изображения, видео, аудио) в реляционных базах данных. Эта технология позволяет интегрировать мультимедийные данные в базу данных и использовать их для различных приложений, например, для мультимедийного хранения и обработки данных, анализа изображений и видео, автоматической идентификации или распознавания, а также для создания мультимедийных приложений.

В базах данных, которые поддерживают мультимедийные типы данных, мультимедийная информация хранится в виде объектов BLOB (двоичных крупных объектов) в таблицах базы данных. Различные типы мультимедийных данных могут использовать различные форматы хранения данных, такие как JPEG или GIF для изображений, MPEG или AVI для видео, а также различные аудиоформаты.

Одним из преимуществ использования мультимедийных типов данных в базе данных является возможность интеграции не только текстовой информации, но и мультимедийной информации в базу данных. Это расширяет возможности приложений и упрощает их разработку.

Однако, управление мультимедийными данными в базе данных может представить некоторые вызовы, такие как масштабирование хранилища данных, управление доступом и безопасностью данных. Также для хранения больших мультимедийных файлов может потребоваться большой объем памяти и вычислительных ресурсов.

В различных типах баз данных, таких как Oracle, PostgreSQL и MySQL, существуют различные API и функции, чтобы обеспечить поддержку мультимедийных типов данных. Oracle предоставляет поддержку обработки и хранения мультимедийных типов данных с помощью технологии Oracle Multimedia, которая включает в себя API для работы с изображениями, аудио и видео, а также возможности поиска по содержимому файлов, распознавание образов и многое другое.

В целом, использование мультимедийных типов данных в базах данных позволяет интегрировать в базу данных различные типы мультимедийной информации и упрощает разработку приложений, которые могут использовать эти данные. Однако, управление мультимедийными данными также представляет свои вызовы, такие как управление доступом и безопасностью данных, а также масштабирование хранилища данных.

6.2 Применение технологии

Для начала необходимо создать директорию, где будут хранится медиафайлы. Для создания директории в базе данных Oracle можно использовать оператор «CREATE DIRECTORY**»**. Этот оператор позволяет создавать директории в файловой системе и связывать их с псевдонимами в базе данных, которые можно использовать для обращения к файлам, хранящимся в этих директориях. Процесс создания директории изложен в листинге 6.1.

CREATE DIRECTORY IMAGE\_BASE AS '/media/ImagesDB;

Листинг 6.1 – Создание директории

После создания директории с медиафайлами можно работать с ними и загружать их в базу данных Oracle. В коде процедуры «AddProduct» (листинг 6.2) представлен пример загрузки медиафайла (изображения) в переменную типа «BLOB». Данный код загружает бинарный файл (изображение) в переменную типа «BLOB». Процедура принимает в качестве параметра «p\_img\_path» путь к файлу изображения. С помощью оператора «BFILENAME», созжается «v\_bfile» типа «BFILE», которая хранит путь к файлу изображения в созданной ранее директории «IMAGE\_BASE». Для открытия файла «v\_bfile» в режиме чтения используем функцию «DBMS\_LOB.FILEOPEN».

RETURN "image" INTO v\_blob;

v\_bfile := BFILENAME('IMAGE\_BASE', p\_img\_path);

DBMS\_LOB.FILEOPEN(v\_bfile,DBMS\_LOB.FILE\_READONLY);

DBMS\_LOB.LOADBLOBFROMFILE(v\_blob,v\_bfile,v\_lobMaxSize,v\_desOffset,v\_srcOffset);

DBMS\_LOB.FILECLOSE(v\_bfile);COMMIT;

Листинг 6.2 – Код для работы с картинками

Затем использя функцию «DBMS\_LOB.LOADBLOBFROMFILE», чтобы загрузить данные из файла «v\_bfile» в переменную «v\_blob». Эта функция используется для копирования данных из файла «v\_bfile» «v\_blob», начиная с позиции, заданной в параметрах функции.

Когда закончилась загрузка медиафайла, используется функцию «DBMS\_LOB.FILECLOSE», чтобы закрыть файл «v\_bfile». Это позволяет освободить ресурсы, занятые открытым файлом.

Таким образом, после выполнения процедуры «AddProduct» можно использовать переменную «v\_blob» для сохранения изображения в базе данных Oracle. Это позволяет интегрировать медиафайлы в базу данных и использовать их для различных приложений и задач, таких как отображение изображений в интерфейсе приложения или анализ мультимедийных данных.

1. Описание приложения

Предоставленное приложение является базой данных магазина электроники, и все операции с базой данных осуществляются через процедуры, указанные в разделе 3.2. Приложение имеет три типа пользователей: «Userka» – покупатель, «PIA» – администратор и «Sys» – суперпользователь.

Как покупатель, «Userka» может использовать несколько процедур базы данных, которые позволяют ему получать информацию о продуктах, заказах, отзывах и продажах, а также изменять данные своей учетной записи и оставлять отзывы о продуктах. «Userka» может воспользоваться такими процедурами, как «GetCartInfoByCustomerID», «GetCustomer», «GetOrderInfoByOrderID», «GetOrdersByCustomerID», «GetReviewsByCustomerID», «GetSalesByCustomerID», «UpdateCustomer» и «AddReview».

Как администратор, «PIA» имеет доступ ко всем процедурам, кроме «exportXml» и «importXml», которые принадлежат только «Sys».

В листинге 7.1 представлен код для выполнения процедуры «ADDREVIEW» пользователем «Userka». Данная процедура позволяет пользователю оставлять отзыв о товаре.

P\_CUSTOMER\_ID NUMBER; P\_PRODUCT\_ID NUMBER; P\_RATING NUMBER;

P\_COMMENT VARCHAR2(200); P\_REVIEW\_DATE DATE;

P\_IMAGE\_PATH VARCHAR2(200); P\_REVIEW\_ID NUMBER;

BEGIN

P\_CUSTOMER\_ID := 21; P\_PRODUCT\_ID := 2; P\_RATING := 4;

P\_COMMENT := 'фыафыафыа22';

P\_REVIEW\_DATE := to\_date('2002-02-22','yyyy-MM-dd');

P\_IMAGE\_PATH := 'test.jpg';

ADDREVIEW(

P\_CUSTOMER\_ID => P\_CUSTOMER\_ID,P\_PRODUCT\_ID => P\_PRODUCT\_ID,

P\_RATING => P\_RATING,P\_COMMENT => P\_COMMENT,

P\_REVIEW\_DATE => P\_REVIEW\_DATE, P\_IMAGE\_PATH => P\_IMAGE\_PATH,

P\_REVIEW\_ID => P\_REVIEW\_ID

);

Листинг 7.1 ­– Код вызова процедуры «ADDREVIEW»

Выдача разные разрешения разным пользователям в базе данных имеет несколько преимуществ:

* Безопасность: Разграничение прав доступа в базу данных обеспечивает безопасность данных. Установка разных уровней доступа для разных пользователей может помочь в предотвращении несанкционированного доступа к конфиденциальным данным в базе данных.
* Упрощение управления базой данных: Разграничение прав доступа упрощает управление базой данных. Например, администратор базы данных может присвоить пользователям только те права доступа, которые им необходимы, уменьшая тем самым количество ошибок и снижая потенциальные риски.
* Улучшение производительности: Доступ к различным функциям базы данных разрешается только соответствующим пользователям, которые могут использовать их правильно. Это может уменьшить нагрузку на базу данных и улучшить ее производительность.
* Минимизация рисков: Разграничение прав доступа помогает минимизировать риски, связанные с возможностью несанкционированного изменения или удаления данных в базе данных. Путем ограничения доступа только к необходимым функциям и данным пользователи могут быть защищены от случайных или злонамеренных действий, которые могут повредить целостность или доступность данных.

Выдача разных разрешений разным пользователям является важной стратегией управления базой данных, которая обеспечивает безопасность, упрощает управление и повышает производительность базы данных.

1. Руководство пользователя

Во время работы с базой данных Oracle пользователи могут использовать несколько инструментов для взаимодействия с БД. Один из таких инструментов – командная строка SQLPlus, которая позволяет выполнять базовые SQL-запросы и управлять базой данных напрямую из командной строки. Пример подключения к базе данных с помощью SQLPlus (рисунок 8.1). Где «username», «password», «hostname», «port», и «service\_name» – это соответствующие параметры, требуемые для подключения к конкретной базе данных.

Изображение выглядит как Шрифт, текст, снимок экрана, Графика

Автоматически созданное описание

Рисунок 8.1 – Подключение к базе данных через SQLPlus

Однако, для удобства работы с базой данных Oracle, рекомендуется использовать специализированные инструменты, такие как Oracle SQL Developer. SQL Developer — это бесплатное приложение, которое предоставляет графический интерфейс для работы с базой данных и используется для написания и отладки SQL-запросов, управления данными схемой и объектами базы данных, отображения и редактирования таблиц.

Oracle SQL Developer предлагает множество преимуществ по сравнению с использованием SQL\*Plus в командной строке, включая:

* Простой в использовании графический интерфейс
* Поддержка многих форматов запросов и отчетов
* Поддержка развертывания приложений в базе данных

Поэтому, для работы с базой данных Oracle рекомендуется использовать Oracle SQL Developer вместо SQL\*Plus в командной строке, чтобы повысить производительность работы и улучшить качество кода.

При входе в SQL Developer необходимо создать подключение выбрав «New Connection». Пример подключения представлен на рисунке 8.2.

Изображение выглядит как текст, снимок экрана, число, линия

Автоматически созданное описание

Рисунок 8.2 – Создание нового подключения

После подключения пользователь может создать новый файл выбрав в разделе «File» пункт «New» и создать новый файл с расширением «.sql». В данном файле пользователь может обратить к процедуре с помощью sql запроса. В листинге 8.1 представлен код для работы с процедурой «ADDPOSITION» пользователя «PIA».

DECLARE POSITION\_NAME\_IN VARCHAR2(200);

BEGIN POSITION\_NAME\_IN := 'Worker2';

ADDPOSITION(POSITION\_NAME\_IN => POSITION\_NAME\_IN); --rollback; END;

Листинг 8.1 – Вызов процедуры «ADDPOSITION»

Также, с помощью Oracle SQL Developer можно автоматически генерировать код для вызова процедуры. Для этого нужно перейти в раздел "Procedures" в панели "Connection", выбрать нужную процедуру и выполнить ее (рис. 8.3). После этого появится форма для заполнения параметров работы процедуры (рис. 8.4), которую необходимо заполнить.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 8.3 – Выбор процедуры

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 8.4 – Форма запуска процедуры

Руководство пользователя предоставляет пользователям удобный способ взаимодействия с базой данных Oracle. Описанные процедуры облегчают выполнение различных операций и позволяют пользователям получить доступ к функциональности базы данных без необходимости вводить сложные SQL-запросы или команды.

Заключение

В рамках выполнения работы был разработан дизайн базы данных магазина электроники с учетом разнообразных связей и хранимых параметров.

Для удобства осуществления взаимодействия с таблицами были использованы процедуры. Было проведено тестирование производительности, в котором анализ времени выполнения показал, что база данных работает довольно быстро.

Также в ходе работы была применена технология использования мультимедийных данных в базе данных. Это помогло сохранить информацию о продуктах с изображениями и видео, значительно увеличив использование базы данных магазина и обеспечивая более реалистичное представление о товарах при просмотре для потенциальных покупателей.

В целом, результаты выполнения работы позволят эффективно управлять базой данных магазина электроники и быстро получать необходимую информацию о продуктах, заказах и пользователях. Использование мультимедийных данных в базе данных поможет продвигать продукты и привлекать новых покупателей.

Таким образом, применение технологии использования мультимедийных данных в базе данных позволяет сохранять большой объем информации о продуктах, повышать качество базы данных, снижать время выполнения запросов и обеспечивать лучшее взаимодействие магазина с клиентами.

Список используемой литературы

1. Свободная энциклопедия Википедия [Электронный ресурс] /CRUD – Режим доступа: https://ru.wikipedia.org/wiki/CRUD – Дата доступа: 10.05.2022.
2. Oracle Documentation Library [Электронный ресурс]  – Режим доступа: https://docs.oracle.com/– Дата доступа: 10.05.2022.
3. Oracle Learning Library [Электронный ресурс]  – Режим доступа: https://www.oracle.com/education/– Дата доступа: 10.05.2022.
4. MySQL [Электронный ресурс]  / MySQL 5.7 – Режим доступа: https://dev.mysql.com/doc/refman/5.7/en/ – Дата доступа: 07.05.2022.
5. Quick database diagrams [Электронный ресурс] /DB Diagrams – Режим доступа: https://app.quickdatabasediagrams.com/ – Дата доступа: 09.05.2022.

Приложение А

CREATE TABLE "Orders" (

"ordorder\_id" NUMBER NOT NULL,

"order\_number" NUMBER NOT NULL,

"order\_date" DATE NOT NULL,

"status" VARCHAR2(255) NOT NULL,

"customer\_id" NUMBER NOT NULL,

"product\_id" NUMBER NOT NULL,

constraint ORDERS\_PK PRIMARY KEY ("ordorder\_id"));

/

CREATE TABLE "Customers" (

"customer\_id" NUMBER NOT NULL,

"first\_name" VARCHAR2(255) NOT NULL,

"last\_name" VARCHAR2(255) NOT NULL,

"email" VARCHAR2(255) NOT NULL,

"phone\_number" VARCHAR2(255) NOT NULL,

constraint CUSTOMERS\_PK PRIMARY KEY ("customer\_id"));

/

CREATE TABLE "Products" (

"product\_id" NUMBER NOT NULL,

"product\_name" VARCHAR2(255) NOT NULL,

"category" VARCHAR2(255) NOT NULL,

"price" NUMBER NOT NULL,

"quantity" NUMBER NOT NULL,

"storage\_location" VARCHAR2(255) NOT NULL,

"image" BLOB NOT NULL,

constraint PRODUCTS\_PK PRIMARY KEY ("product\_id"));

/

CREATE TABLE "Categories" (

"category\_id" NUMBER NOT NULL,

"category\_name" VARCHAR2(255) UNIQUE NOT NULL,

constraint CATEGORIES\_PK PRIMARY KEY ("category\_id"));

CREATE sequence "CATEGORIES\_CATEGORY\_ID\_SEQ";

CREATE trigger "BI\_CATEGORIES\_CATEGORY\_ID"

before insert on "Categories"

for each row

begin

select "CATEGORIES\_CATEGORY\_ID\_SEQ".nextval into :NEW."category\_id" from dual;

end;

/

CREATE TABLE "Cart" (

"cart\_id" NUMBER NOT NULL,

"customer\_id" NUMBER NOT NULL,

"product\_id" NUMBER NOT NULL,

"quantity" NUMBER NOT NULL,

constraint CART\_PK PRIMARY KEY ("cart\_id"));

/

CREATE TABLE "Sales" (

"sale\_id" NUMBER NOT NULL,

"quantity" NUMBER NOT NULL,

"sale\_date" DATE NOT NULL,

"customer\_id" NUMBER NOT NULL,

"product\_id" NUMBER NOT NULL,

"salesperson\_id" NUMBER NOT NULL,

constraint SALES\_PK PRIMARY KEY ("sale\_id"));

/

CREATE TABLE "Employees" (

"employee\_id" NUMBER NOT NULL,

"first\_name" VARCHAR2(255) NOT NULL,

"last\_name" VARCHAR2(255) NOT NULL,

"middle\_name" VARCHAR2(255),

"email" VARCHAR2(255) NOT NULL,

"phone\_number" VARCHAR2(255) NOT NULL,

"position" VARCHAR2(255) NOT NULL,

"salary" NUMBER NOT NULL,

constraint EMPLOYEES\_PK PRIMARY KEY ("employee\_id","first\_name","last\_name","middle\_name","email","phone\_number","position","salary"));

/

CREATE TABLE "Positions" (

"position\_id" NUMBER NOT NULL,

"position\_name" VARCHAR2(255) UNIQUE NOT NULL,

constraint POSITIONS\_PK PRIMARY KEY ("position\_id"));

/

CREATE TABLE "Reviews" (

"review\_id" NUMBER NOT NULL,

"customer\_id" NUMBER NOT NULL,

"product\_id" NUMBER NOT NULL,

"rating" NUMBER NOT NULL,

"comment" VARCHAR2(255),

"review\_date" DATE NOT NULL,

"image" BLOB NOT NULL,

constraint REVIEWS\_PK PRIMARY KEY ("review\_id"));

/

ALTER TABLE "Orders" ADD CONSTRAINT "Orders\_fk0" FOREIGN KEY ("customer\_id") REFERENCES "Customers"("customer\_id");

ALTER TABLE "Orders" ADD CONSTRAINT "Orders\_fk1" FOREIGN KEY ("product\_id") REFERENCES "Products"("product\_id");

ALTER TABLE "Products" ADD CONSTRAINT "Products\_fk0" FOREIGN KEY ("category") REFERENCES "Categories"("category\_id");

ALTER TABLE "Cart" ADD CONSTRAINT "Cart\_fk0" FOREIGN KEY ("customer\_id") REFERENCES "Customers"("customer\_id");

ALTER TABLE "Cart" ADD CONSTRAINT "Cart\_fk1" FOREIGN KEY "product\_id") REFERENCES "Products"("product\_id");

ALTER TABLE "Sales" ADD CONSTRAINT "Sales\_fk0" FOREIGN KEY ("customer\_id") REFERENCES "Customers"("customer\_id");

ALTER TABLE "Sales" ADD CONSTRAINT "Sales\_fk1" FOREIGN KEY ("product\_id") REFERENCES "Products"("product\_id");

ALTER TABLE "Sales" ADD CONSTRAINT "Sales\_fk2" FOREIGN KEY ("salesperson\_id") REFERENCES "Employees"("employee\_id");

ALTER TABLE "Employees" ADD CONSTRAINT "Employees\_fk0" FOREIGN KEY ("position") REFERENCES "Positions"("position\_id");

ALTER TABLE "Reviews" ADD CONSTRAINT "Reviews\_fk0" FOREIGN KEY ("customer\_id") REFERENCES "Customers"("customer\_id");

ALTER TABLE "Reviews" ADD CONSTRAINT "Reviews\_fk1" FOREIGN KEY ("product\_id") REFERENCES "Products"("product\_id");

Приложение Б

create or replace NONEDITIONABLE PROCEDURE AddProduct (

p\_product\_name VARCHAR2,

p\_category\_id NUMBER,

p\_price NUMBER,

p\_quantity NUMBER,

p\_storage\_location VARCHAR2,

p\_img\_path VARCHAR2

)

AS

v\_blob blob;

v\_bfile bfile;

v\_desOffset INTEGER := 1;

v\_srcOffset INTEGER := 1;

v\_lobMaxSize CONSTANT INT := DBMS\_LOB.LOBMAXSIZE;

BEGIN

INSERT INTO "Products" ("product\_id", "product\_name", "category", "price", "quantity", "storage\_location", "image")

VALUES (Products\_seq.NEXTVAL, p\_product\_name, p\_category\_id, p\_price, p\_quantity, p\_storage\_location, empty\_blob ())

RETURN "image" INTO v\_blob;

v\_bfile := BFILENAME('IMAGE\_BASE', p\_img\_path);

DBMS\_LOB.FILEOPEN(v\_bfile,DBMS\_LOB.FILE\_READONLY);

DBMS\_LOB.LOADBLOBFROMFILE(v\_blob,v\_bfile,v\_lobMaxSize,v\_desOffset,v\_srcOffset);

DBMS\_LOB.FILECLOSE(v\_bfile);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

END;

create or replace NONEDITIONABLE PROCEDURE AddOrder(

p\_order\_date DATE,

p\_status VARCHAR2,

p\_customer\_id NUMBER,

p\_product\_id NUMBER,

p\_quantity NUMBER

)

AS

BEGIN

INSERT INTO "Orders" ("order\_id", "order\_date", "status", "customer\_id", "product\_id", "quantity")

VALUES (Orders\_seq.NEXTVAL, p\_order\_date, p\_status, p\_customer\_id, p\_product\_id, p\_quantity);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Order has been added successfully.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Customer or Product with provided IDs does not exist.');

WHEN OTHERS THEN

ROLLBACK;

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

END;

create or replace NONEDITIONABLE PROCEDURE UpdatePosition(

id\_in NUMBER,

position\_name\_in VARCHAR2

)

AS

BEGIN

UPDATE "Positions"

SET "position\_name"=position\_name\_in

WHERE "position\_id"=id\_in;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Position with ID ' || id\_in || ' has been updated.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Position with ID ' || id\_in || ' does not exist.');

END;

create or replace NONEDITIONABLE PROCEDURE UpdatePosition(

id\_in NUMBER,

position\_name\_in VARCHAR2

)

AS

BEGIN

UPDATE "Positions"

SET "position\_name"=position\_name\_in

WHERE "position\_id"=id\_in;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Position with ID ' || id\_in || ' has been updated.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Position with ID ' || id\_in || ' does not exist.');

END;

create or replace NONEDITIONABLE PROCEDURE MoveCartToOrders (

p\_customer\_id IN NUMBER

)

IS

v\_order\_id NUMBER;

BEGIN

-- Получение идентификатора для нового заказа

SELECT ORDERS\_SEQ.NEXTVAL INTO v\_order\_id FROM dual;

-- Вставка записей в таблицу Orders

INSERT INTO "Orders" ("order\_id", "order\_date", "status", "customer\_id", "product\_id", "quantity")

SELECT ORDERS\_SEQ.NEXTVAL, SYSDATE, 'начато', p\_customer\_id, "product\_id", "quantity"

FROM "Cart"

WHERE "customer\_id" = p\_customer\_id;

DELETE FROM "Cart" WHERE "customer\_id" = p\_customer\_id;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Записи в таблице Cart для клиента с id ' || p\_customer\_id || ' не найдены');

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Ошибка: ' || SQLCODE || ' ' || SQLERRM);

END;

create or replace NONEDITIONABLE procedure GetOrderInfoByOrderID (p\_order\_id NUMBER) as

cursor order\_info\_cursor is

select "order\_id", "order\_date", "status", "customer\_id", "product\_id", "quantity"

from "Orders"

where "order\_id" = p\_order\_id;

begin

dbms\_output.put\_line('INFO FOR ORDER ID ' || p\_order\_id || ':');

dbms\_output.put\_line('---------------------------');

for ord in order\_info\_cursor loop

dbms\_output.put\_line(

'Order ID: ' || ord."order\_id" ||

' | Order Date: ' || to\_char(ord."order\_date", 'MM/DD/YYYY') ||

' | Status: ' || ord."status" ||

' | Customer ID: ' || ord."customer\_id" ||

' | Product ID: ' || ord."product\_id" ||

' | Quantity: ' || ord."quantity"

);

end loop;

dbms\_output.put\_line('---------------------------');

END;