МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

«Реализация базы данных магазина одежды с применением технологии резервного копирования и восстановления»

Выполнил студент Яшный Никита Сергеевич

(Ф.И.О.)

Руководитель работы асс. Нистюк Ольга Александровна

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

И.о. зав. кафедрой ст. преп. Блинова Е.А.

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

Курсовая работа защищена с оценкой

Минск 2023

**Содержание**

[Введение 3](#_Toc153886922)

[1 Постановка задачи 5](#_Toc153886923)

[1.1 Аналитический обзор аналогов 5](#_Toc153886924)

[1.1.1 Zara 5](#_Toc153886925)

[1.1.2 Bershka 6](#_Toc153886926)

[1.1.3 Pull & Bear 6](#_Toc153886927)

[1.1.4 Lamoda 7](#_Toc153886928)

[1.2Разработка функциональных требований, определение вариантов использования 8](#_Toc153886929)

[1.3 Вывод 8](#_Toc153886930)

[2 Проектирование базы данных 10](#_Toc153886931)

[2.1 Обобщенная структура создания базы данных 10](#_Toc153886932)

[2.2 Диаграмма базы данных. 10](#_Toc153886933)

[2.3 Описание информационных объектов 11](#_Toc153886934)

[2.4 Вывод 11](#_Toc153886935)

[3 Разработка объектов базы данных 12](#_Toc153886936)

[3.1 Создание необходимых объектов 12](#_Toc153886937)

[3.2 Создание таблиц 12](#_Toc153886938)

[3.3 Создание ролей для разграничения 13](#_Toc153886939)

[3.4 Создание процедур для базы данных 13](#_Toc153886940)

[3.4.1 Описание процедур роли гостя 15](#_Toc153886941)

[3.4.2 Описание процедур роли пользователя 15](#_Toc153886942)

[3.4.3 Описание процедур роли администратора 16](#_Toc153886943)

[3.4.4 Добавление в таблицу 100000 строк 16](#_Toc153886944)

[3.5 Представления базы данных 17](#_Toc153886945)

[3.6 Вывод 17](#_Toc153886946)

[4 Описание процедур импорта и экспорта 18](#_Toc153886947)

[5 Тестирование производительности 20](#_Toc153886948)

[6 Описание технологии и ее применение в базе данных 21](#_Toc153886949)

[7 Руководство пользователя 22](#_Toc153886950)

[7.1 Сценарий использования базы данных 22](#_Toc153886951)

[7.2 Вывод 22](#_Toc153886952)

[Заключение 23](#_Toc153886953)

[Список литературы 24](#_Toc153886954)

[Приложение А 25](#_Toc153886955)

# Введение

В современном мире многие люди постоянно совершают покупки одежды через интернет. И если с точки зрения покупателей существует множество сайтов и программных средств для поиска товара, то с точки зрения продавцов выбор программных средств для централизованной последовательной записи и размещения предложения сравнительно невелик.

Целью данной работы являлась разработка реляционной базы данных для магазина одежды. Эта база данных составлялась для обеспечения покупателям доступа к функциям заказа товара от поставщиков, а для самих поставщиков — размещения позиций и манипулирования ими.

База данных — это организованная структура, предназначенная для хранения 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. В качестве СУБД для базы данных была выбрана Oracle, в связи с ее высокой производительностью и надежностью.

В основной части будут затронуты все аспекты разработки проекта и обоснованы некоторые технические приёмы, к которым приходилось прибегнуть для реализации работы базы данных.

При создании базы данных использовалась технология резервного копирования и восстановления.

Основные требования к приложению:

* Определение ролей.
* Управление товарами.
* Управление категориями товара.
* Анализ продукции.
* Обеспечение резервирования товаров.
* Возможность оформления заказа.
* Возможность добавления товара в корзину.

В пояснительной записке вы сможете найти краткую информацию о похожих продуктах, архитектуре, реализации проекта, руководстве пользователя.

# 1 Постановка задачи

## 1.1 Аналитический обзор аналогов

Были проанализированы цели и задачи, поставленные в данном курсовом проекте, а также рассмотрены аналогичные примеры их решений. На основании анализа всех достоинств и недостатков данных альтернативных решений были сформулированы требования к данному программному средству.

# Zara

Одним из самых популярных альтернативных решений является интернет-ресурс «Zara», выступающая известным брендом молодёжной одежды, собравшим огромную клиентскую базу во всём мире. У сайта широкий функционал, доступный любому пользователю: удобный поиск товара, наличие сортировки по категории, полу, размеру. Данное программное средство предоставляет пользователю возможность легко найти нужный товар, добавить его в корзину и совершить покупку.

Интерфейс интернет-ресурса представлен на рисунке 1.1.

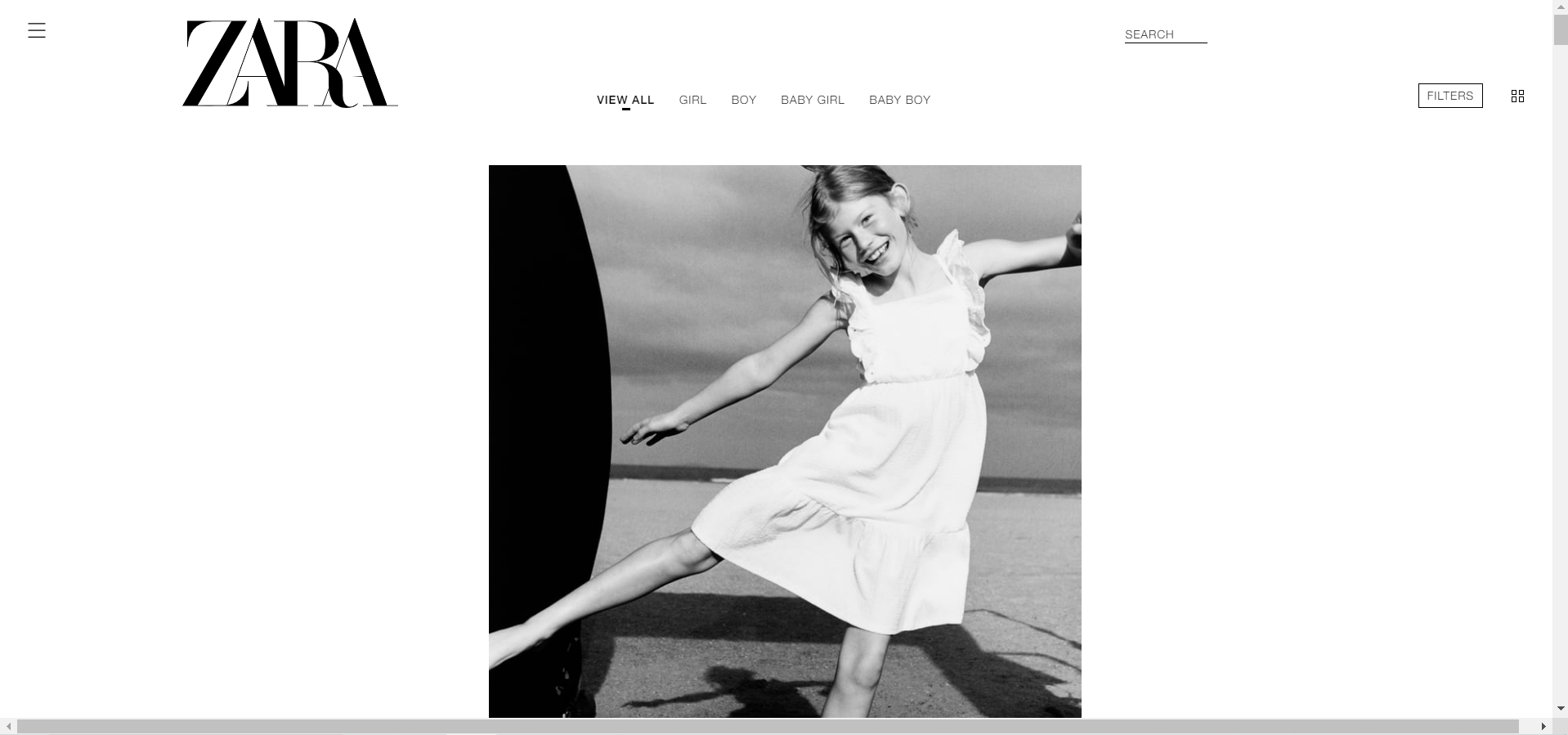


Рисунок 1.1 – Интернет-ресурс «Zara»

Из выявленных недостатков данного ресурса можно выделить отсутствие возможности зарезервировать товар на какое-то время, что гарантирует пользователю, что товар не исчезнет после попадания в корзину. Также на сайте нет возможности просмотреть популярные товары и количество продаж за определённые периоды.

# Bershka

Ещё одним альтернативным решением задач, поставленных в заданном курсовом проекте, является интернет-ресурс «Bershka». С помощью данного программного средства пользователь может ознакомиться с каталогом данного бренда одежды, выбрать для себя подходящий по вкусу товар, а также оформить доставку, введя данные своей банковской карты. Интерфейс выглядит привлекательным и удобным.

Интерфейс интернет-ресурса представлен на рисунке 1.2.

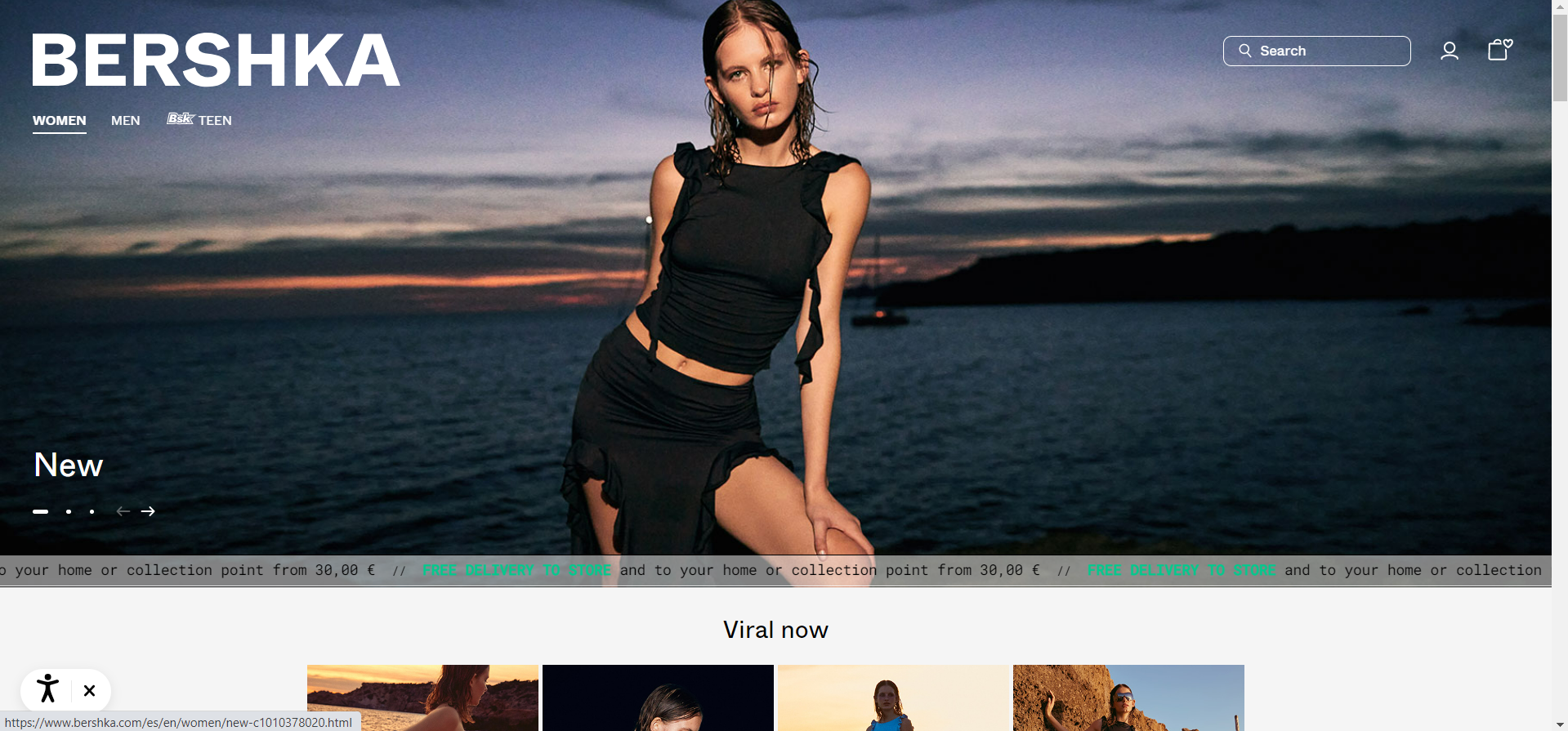


Рисунок 1.2 – Интернет-ресурс «Bershka»

Одним из главных недостатков такой альтернативы является отсутствие возможности посмотреть количество оставшихся на складе товаров, а также невозможность получить отзыв от других пользователей. В данном интернет-ресурсе тоже отсутствует возможность зарезервировать понравившийся товар на какое-то время.

# Pull & Bear

Ещё одной альтернативой решения поставленных задач является интернет-ресурс «Pull&Bear». С помощью данного программного средства пользователь может ознакомиться со списком доступных для покупки товаров, выбрать доступную для любого бюджета одежду. Немаловажным является возможность просмотреть статус заказа. На сайте крайне удобное разделение товара по категориям.

Интерфейс интернет-ресурса представлен на рисунке 1.3.

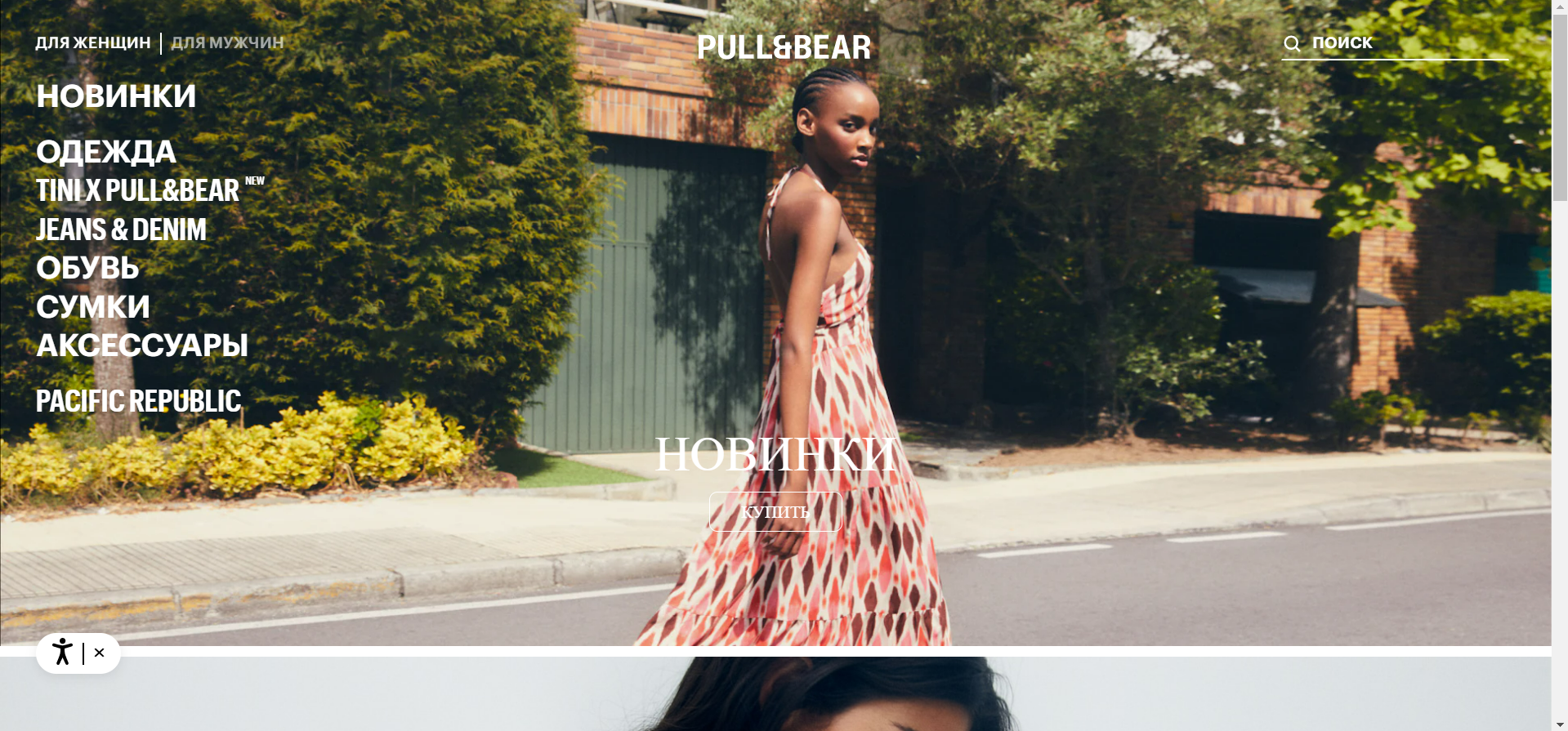


Рисунок 1.3 – Интернет-ресурс «Pull&Bear»

Однако, недостатком данной альтернативы являются отсутствие качественной оптимизации, из-за чего даже при наличии высокоскоростного интернета многие разделы долго загружаются. Также, как и у приведённых ранее примеров альтернатив, на сайте отсутствует возможность резервирования и просмотра популярных позиций.

# Lamoda

В список альтернатив, рассматриваемых для решения поставленных задач, также можно включить интернет-ресурс «Lamoda». Данное программное средство позволяет ознакомиться с предложениями от различных брендов в рамках одного маркетплейса. Приложение позволяет выбирать товар, сортируя не только по классическим категориям вроде пола и размера, но и по брендам и популярности, что является крайне удобным при поиске необходимого товара.

Интерфейс интернет-ресурса представлен на рисунке 1.4.

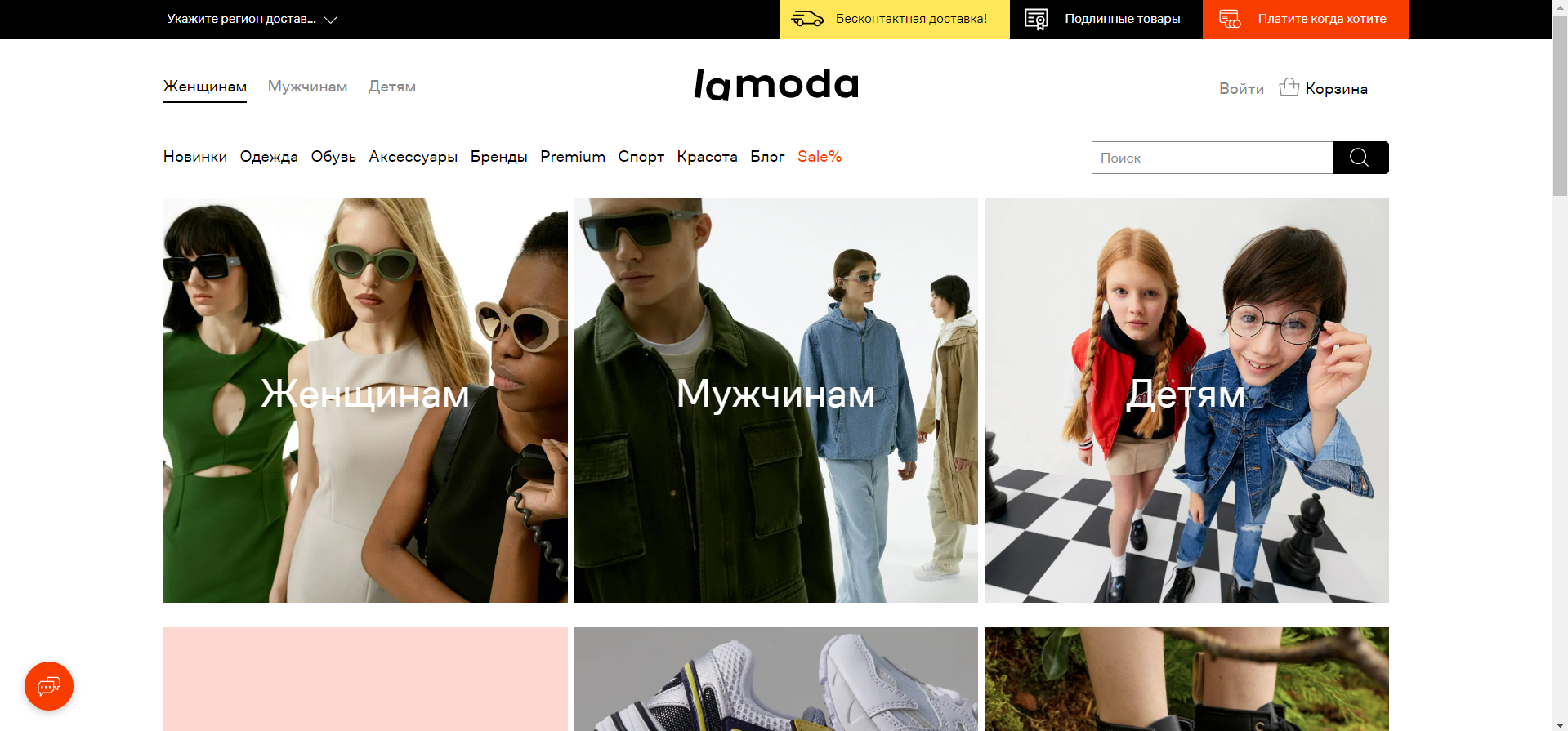


Рисунок 1.4 – Интернет-ресурс «Lamoda»

Недостатком данного программного средства является отсутствие возможности получения прямой связи с производителем, что может быть неудобным при возникновении у клиента вопросов о товаре. Нет возможности легко отменить свой заказ прямо с сайта без связи с продавцом и так же, как у конкурентов отсутствует возможность резервирования.

## 1.2 Разработка функциональных требований, определение вариантов использования

Функциональные требования базы данных определяют, как база данных должна обрабатывать данные и предоставлять пользовательскому интерфейсу необходимую функциональность. Это может включать в себя описание того, как данные должны храниться и организовываться, как происходит поиск и выборка данных, каким образом обновляются данные и какие механизмы используются для защиты данных.

Помимо функциональных требований, важно также определить роли пользователей и их варианты использования системы. Это помогает определить, какие функции должны быть доступны для каждой роли, какие данные должны быть доступны для каждой роли.

Роли пользователя – это набор прав, которые пользователь может получить в системе. В зависимости от роли пользователя, он может иметь доступ к различным функциям системы. В данном проекте роли пользователей будут следующими:

* GUEST\_DEFUALT;
* USER\_DEFAULT;
* ADMIN\_DEFAULT;
* Главный администратор.

На основе предоставленного списка ролей была построена uml-диаграмма вариантов использования. Диаграмма использования изображена на рисунке 1.5.

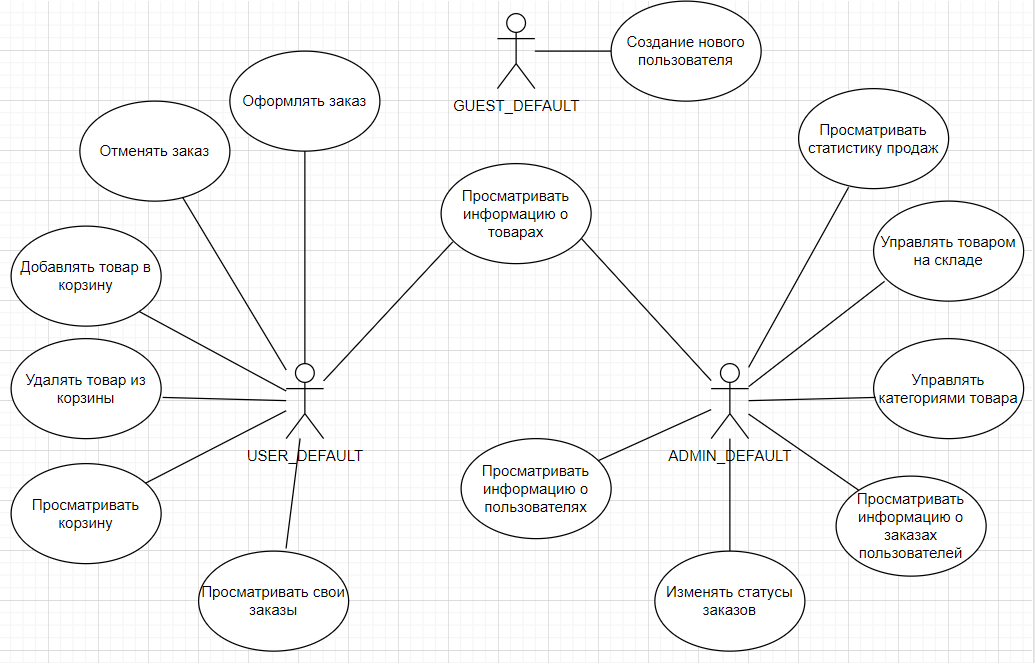


Рисунок 1.5 – Диаграмма использования

Роль GUEST\_DEFAULT позволяет только создать новый аккаунт. После создания аккаунта пользователь получает роль USER\_DEFAULT.

Роль USER\_DEFAULT дает возможность просматривать каталог, оформлять заказы, добавлять товары в корзину, резервировать товары.

Роль ADMIN\_DEFAULT также может просматривать информацию о товарах, но для него есть процедуры для более подробному анализа продукции. Он может просматривать информацию о заказах пользователей и менять их статус. Также он может манипулировать товарами на складе и их категориями.

Роль главного администратора все возможные права.

Приложение обладает четырьмя ролями, на которые разложены все основные функции для корректного использования.

## 1.3 Вывод

Итого, был проведен аналитический обзор аналогов, которые уже существуют на рынке. Этот обзор позволил определить основные характеристики и функциональные возможности, которые необходимо предусмотреть в разрабатываемой системе. Также были определены функциональные требования базы данных, а также роли пользователей и варианты использования системы в зависимости от этих ролей.

# 2 Проектирование базы данных

## 2.1 Обобщенная структура создания базы данных

Процесс проектирования базы данных состоит из следующих этапов:

1) сбор информации;

2) определение сущностей;

3) определение атрибутов для каждой сущности;

4) определение связей между сущностями;

5) нормализация.

На этапе сбора информации вам необходимо точно определить, как будет использоваться база данных, и какая информация будет в ней храниться.

Далее следует этап определения сущностей и на нем определяются сущности, из которых будет состоять база данных.

## 2.2 Диаграмма базы данных.

Диаграмма базы данных таблиц (Database Table Diagram) – это визуальное представление структуры базы данных и отношений между таблицами, которые хранятся в этой базе данных. Диаграмма базы данных представлена на рисунке 2.1.

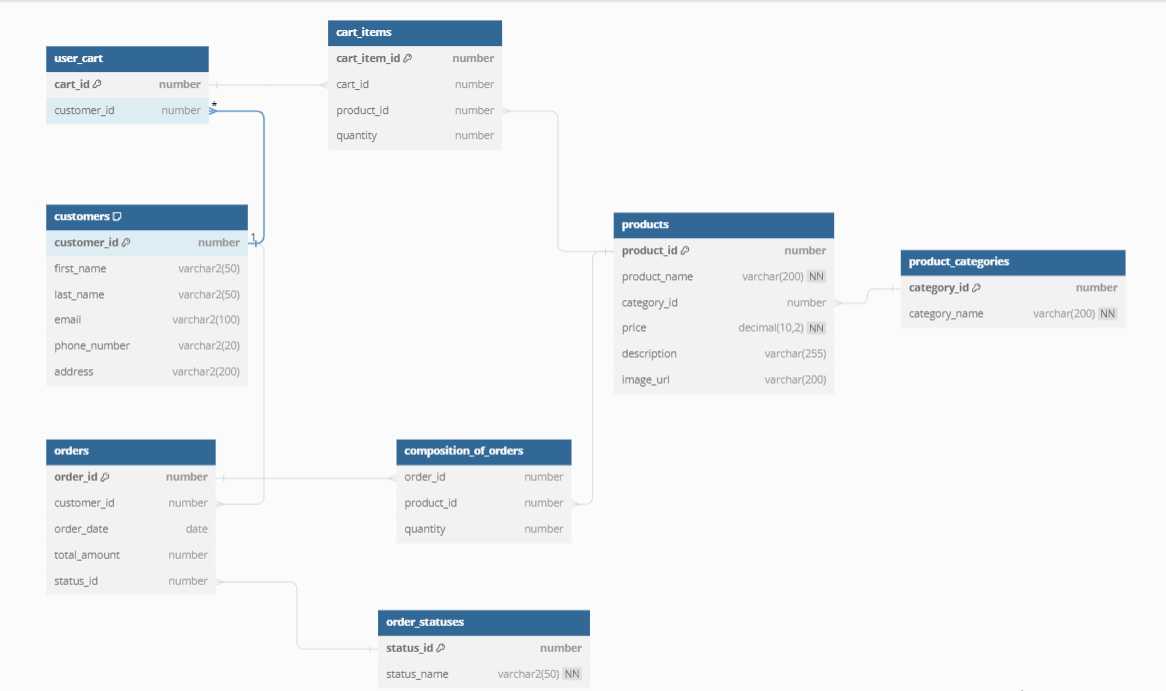


Рисунок 2.1 – Диаграмма базы данных

Таким образом, диаграмма показывает связи между таблицами и полями, а также отношения между ними, такие как связи "один-ко-одному", "один-ко-многим", "многие-ко-многим".

## 2.3 Описание информационных объектов

База данных состоит из 8 таблиц, связанных между собой внешними ключами.

Таблица «PRODUCT\_CATEGORIES» хранит информацию о существующих категориях товара.

Таблица «PRODUCTS» хранит информацию о названии, категории, цене, описании, количестве товаров. Связана внешним ключом с таблицей «PRODUCT\_CATEGORIES».

Таблица «CUSTOMERS» хранит информацию о пользователях базы данных: их имя, фамилию и контактную информацию.

Таблица «ORDERS» хранит информацию о заказах. В ней хранится идентификатор заказа, идентификатор покупателя, дату и цену. Таблица связана внешним ключом с таблицей «CUSTOMERS» и «ORDER\_STATUSES».

Таблица «COMPOSITION\_OF\_ORDERS» хранит в себе информацию о составе заказа, такую как идентификатор заказа, идентификатор товара, количество. Связана внешним ключом с таблицей «ORDERS» и «PRODUCTS».

Таблица «ORDER\_STATUSES» хранит информацию о возможных статусах заказа.

Таблица «USER\_CART» хранит информацию о корзине пользователя. Связана внешним ключом с таблицей «USERS».

Таблица «CART\_ITEMS» хранит информацию о содержимом корзины покупателя. Связана внешним ключом с таблицами «USER\_CART» и «PRODUCTS».

## 2.4 Вывод

Разработка архитектуры проекта необходима для определения структуры и функциональности базы данных.

Описание информационных объектов является важной частью архитектуры проекта, так как это помогает понять, какие данные будут использоваться в приложении, и как они будут храниться и обрабатываться.

# 3 Разработка объектов базы данных

## 3.1 Создание необходимых объектов

Для создания базы должны быть реализованы таблицы упомянутые в предыдущем разделе. Для более удобной работы с ними должны быть созданы индексы, функции, процедуры и представления. Скрипты создания таблиц представлены в приложении А.

## 3.2 Создание таблиц

Для базы данных создадим восемь основных таблиц: PRODUCT\_CATEGORIES, PRODUCTS, CUSTOMERS, ORDERS, COMPOSITION\_OF\_ORDERS, ORDER\_STATUSES, USER\_CART, CART\_ITEMS.

Каждая таблица будет содержать свои поля (столбцы) и ограничения (constraints), которые определяют правила для хранения и изменения данных. Например, ограничение FOREIGN KEY определяет связь между двумя таблицами, а ограничение PRIMARY KEY определяет уникальный идентификатор для каждой записи в таблице.

Кроме того, в базе данных будут присутствовать связи между таблицами. Одна из основных связей — это связь "один ко многим" (one-to-many), которая определяет отношение одной записи в таблице к нескольким записям в другой таблице. Например, у каждого пользователя может быть множество заказов, которые он оформил. Для этого мы добавим в таблицу ORDERS внешний ключ (FOREIGN KEY) на таблицу CUSTOMERS, который будет указывать на идентификатор пользователя.

Скрипт создание таблицы ORDERS будет представлен на листинге 3.1.

|  |
| --- |
| create table orders  (  order\_id number primary key,  customer\_id number,  order\_date date,  total\_amount number,  foreign key (customer\_id) references customers(customer\_id)  );  alter table orders  add status\_id number;  alter table orders  add constraint fk\_order\_statuses foreign key (status\_id) references order\_statuses(status\_id); |

Листинг 3.1 – Cкрипт создание таблицы ORDERS

Таким образом были созданы восемь таблиц в которых будут храниться соответствующие данные.

## 3.3 Создание ролей для разграничения

В этом разделе создаются роли для ограничения доступа к базе данных. Создание ролей позволяет установить границы доступа к различным функциям базы данных и предотвратить несанкционированный доступ к конфиденциальной информации.

## 3.4 Создание процедур для базы данных

Для управления данными пользователи и администраторы используют хранимые процедуры. Хранимая процедура представляет собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

Написанные в ходе разработки курсового проекта процедуры и функции можно разбить на несколько категорий:

1. Процедуры доступные гостю.
2. Процедуры доступные пользователю.
3. Процедуры доступные администратору.
4. Заполнение таблиц 100 000 строк.

В зависимости от того, какую задачу необходимо выполнить, следует использовать хранимую процедуру. Хранимые процедуры могут использоваться для выполнения сложных операций над данными, таких как массовые изменения в таблицах, а также для оптимизации производительности приложения.

Будут созданы три роли для разграничения доступа к базе данных: GUEST\_DEFAULT, USER\_DEFAULT, ADMIN\_DEFAULT. Описание этих ролей будет идти далее.

Роль GUEST\_DEFAULT имеет доступ лишь к процедуре, которая позволяет добавить нового пользователя. То есть ей доступно выполнение процедуры create\_new\_user. Реализация данной роли показана на листинге 3.2.

|  |
| --- |
| create role GUEST\_DEFAULT;  grant create session to GUEST\_DEFAULT;  grant execute on create\_new\_user to GUEST\_DEFAULT; |

Листинг 3.2 – Реализация роли GUEST\_DEFAULT

Роль USER\_DEFAULT имеет право на выполнение всех операций, связанных с объектами, доступными пользователю. Выданные привилегии роли USER\_DEFAULT можно увидеть на листинге 3.3.

|  |
| --- |
| create role USER\_DEFAULT;  grant create session to USER\_DEFAULT;  grant alter session to USER\_DEFAULT;  grant execute on make\_order to USER\_DEFAULT;  grant execute on undo\_order to USER\_DEFAULT;  grant execute on undo\_last\_order to USER\_DEFAULT;  grant execute on add\_product\_to\_cart to USER\_DEFAULT;  grant execute on remove\_from\_cart to USER\_DEFAULT;  grant execute on get\_cart to USER\_DEFAULT;  grant execute on view\_sales\_by\_period to USER\_DEFAULT;  grant execute on display\_popular\_products to USER\_DEFAULT;  grant execute on display\_total\_amount\_of\_products to USER\_DEFAULT;  grant execute on display\_product\_categories to USER\_DEFAULT;  grant execute on GetCustomerOrders to USER\_DEFAULT;  grant select any dictionary to USER\_DEFAULT;  grant select on products to USER\_DEFAULT;  grant select on customers to USER\_DEFAULT;  grant insert on orders to USER\_DEFAULT;  grant insert on composition\_of\_orders to USER\_DEFAULT; |

Листинг 3.3 – Привилегии, выданные роли USER\_DEFAULT

Роль ADMIN\_DEFAULT имеет все те же права, что и предыдущая роль, но плюс ко всему ей доступны процедуры манипуляции товарами и анализирования продаж. Выданные привилегии роли ADMIN\_DEFAULT можно увидеть на листинге 3.4.

|  |
| --- |
| create role ADMIN\_DEFAULT;  grant create session to ADMIN\_DEFAULT;  grant execute on make\_order to ADMIN\_DEFAULT;  grant execute on undo\_order to ADMIN\_DEFAULT;  grant execute on undo\_last\_order to ADMIN\_DEFAULT;  grant execute on add\_product\_to\_cart to ADMIN\_DEFAULT;  grant execute on remove\_from\_cart to ADMIN\_DEFAULT;  grant execute on add\_category to ADMIN\_DEFAULT;  grant execute on delete\_category to ADMIN\_DEFAULT;  grant execute on add\_product to ADMIN\_DEFAULT;  grant execute on delete\_product to ADMIN\_DEFAULT;  grant execute on update\_product to ADMIN\_DEFAULT;  grant execute on add\_quantity\_product to ADMIN\_DEFAULT;  grant execute on remove\_quantity\_product to ADMIN\_DEFAULT;  grant execute on view\_sales\_by\_period to ADMIN\_DEFAULT;  grant execute on display\_total\_amount\_of\_products to ADMIN\_DEFAULT;  grant execute on display\_popular\_products to ADMIN\_DEFAULT;  grant execute on display\_product\_categories to ADMIN\_DEFAULT;  grant execute on display\_order\_information to ADMIN\_DEFAULT;  grant execute on display\_customer\_information to ADMIN\_DEFAULT;  grant execute on GetCustomerOrders to ADMIN\_DEFAULT;  grant execute on change\_order\_status to ADMIN\_DEFAULT; |

Листинг 3.4 – Привилегии, выданные роли ADMIN\_DEFAULT

Роль Главного администратора выполнять все возможные процедуры и действия в базе данных.

### 3.4.1 Описание процедур роли гостя

Гостю доступна только процедура create\_new\_user, которая позволяет создать нового пользователя.

Листинг процедуры create\_new\_user представлен в листинге 3.5

|  |
| --- |
| create or replace procedure create\_new\_user (  p\_username in varchar,  p\_usersurname in varchar,  p\_email in varchar,  p\_phone\_number in varchar,  p\_address in varchar,  p\_password in number  ) as  p\_id number;  begin  IF p\_email IS NOT NULL THEN  IF REGEXP\_LIKE(p\_email, '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$') = FALSE THEN  dbms\_output.put\_line('Неправильный формат электронной почты');return;END IF;END IF;  IF REGEXP\_LIKE(p\_phone\_number, '^\+[0-9]{1,3}[0-9]{9}$') = FALSE THEN  dbms\_output.put\_line('Неправильный формат телефона');return;end if;  select NVL(max(customer\_id),0)  into p\_id  from customers;  insert into customers values (p\_id + 1, p\_username, p\_usersurname, p\_email, p\_phone\_number, p\_address);  execute immediate 'create user ' || p\_username || ' identified by ' || p\_password;  execute immediate 'grant USER\_DEFAULT to ' || p\_username;  commit;  exception  when others then  dbms\_output.put\_line('Ошибка при выполнении процедуры');  rollback;  raise;end; |

Листинг 3.5 – Функция create\_new\_user

### 3.4.2 Описание процедур роли пользователя

Для пользователя в базе данных доступны следующие процедуры:

* авторизация (осуществляется как подключение к базе данных);
* оформление заказа (make\_order);
* отмена заказа (undo\_order);
* отмена последнего заказа (undo\_last\_order);
* добавление товара в корзину (add\_to\_cart);
* удаление товара из корзины (remove\_from\_cart);
* резервирование товара (при добавлении товара в корзину);
* просмотр корзины (get\_cart);
* просмотр продаж магазина по периодам (view\_sales\_by\_period);
* просмотр популярных позиций (display\_popular\_products);
* просмотр категорий товаров (display\_product\_categories);
* просмотр своих заказов (getcustomerorders).

### 3.4.3 Описание процедур роли администратора

Для администратора в базе данных доступны следующие процедуры:

* добавление новой категории товаров (add\_new\_category);
* удаление категории товара (remove\_category);
* добавление нового товара (add\_product);
* удаление товара (remove\_product);
* обновление товара (update\_product);
* привоз товара (add\_quantity\_product);
* списание товара (remove\_quantity\_product);
* просмотр продаж магазина по периодам (view\_sales\_by\_period);
* просмотр популярных позиций (display\_popular\_products);
* просмотр категорий товаров (display\_product\_categories);
* просмотр истории заказов пользователей (display\_order\_information);
* просмотр информации о пользователях (display\_customer\_information);
* изменение статуса заказа (change\_order\_status).

### 3.4.4 Добавление в таблицу 100000 строк

Для заполнения таблицы products была разработан код, который вставляет 100000 строк в таблицу. Процедура представлена на листинге 3.6.

|  |
| --- |
| declare  v\_category\_id number;  begin  for i in 19..100019 loop  insert into products  values (i, 'Product' || i, 1, 130, 'Description ' || i, 'Image URL ' || i, 1);  if mod(i, 1000) = 0 thencommit;  end if;end loop;commit;end; |

Листинг 3.6 –заполнения таблицы products

Данная блок использует цикл while, который с каждой итерацией увеличивает значение переменной i на 1 и делает новую запись в таблицу products.

## 3.5 Представления базы данных

Представление в базе данных представляет собой виртуальную таблицу, которая создается на основе запроса к одной или нескольким таблицам в базе данных. Представления позволяют обращаться к данным из нескольких таблиц одновременно, при этом не изменяя структуру этих таблиц.

В данном проекте были созданы три материализованных представления:

* sales\_by\_period\_view, которое позволяет посмотреть проданные товары за периоды времени;
* popular\_products, которое позволяет посмотреть популярные товары за всё время.
* total\_amount\_of\_product, которое позволяет посмотреть общее количество товара на складе.

Представление sales\_by\_period\_view представлено на листинге 3.7.

|  |
| --- |
| create materialized view sales\_by\_period\_view  refresh complete on commit  as  select to\_char(o.order\_date, 'YYYY-MM') as month, p.product\_name, sum(co.quantity) as total\_quantity  from orders o  join composition\_of\_orders co on o.order\_id = co.order\_id  join products p on co.product\_id = p.product\_id  group by to\_char(o.order\_date, 'YYYY-MM'), p.product\_name  order by month desc; |

Листинг 3.7 – Представление returned\_users

Данные представления помогают работать с таблицами более гибко.

## 3.6 Вывод

В данном разделе была рассмотрена разработка объектов базы данных, где можно просмотреть общую архитектуру проекта.

# 4 Описание процедур импорта и экспорта

База данных имеет возможность импортировать данные для таблицы products в формат JSON и экспортировать данные из JSON в таблицу products. Это может быть полезно в случае необходимости переноса данных на другой сервер или резервного копирования данных.

Для экспорта была разработана процедура ExportToJSON. Она позволяет экспортировать данные из таблицы products в json. Данная процедура представлена в листинге 4.1.

|  |
| --- |
| create or replace procedure ExportToJSON  as  json\_data CLOB;  file\_handle utl\_file.file\_type;  cursor c\_products is  SELECT json\_object('product\_id' VALUE p.product\_id,  'product\_name' VALUE p.product\_name,  'category\_id' VALUE p.category\_id,  'price' VALUE p.price,  'description' VALUE p.description,  'image\_url' VALUE p.image\_url,  'quantity' VALUE p.quantity) as json\_row  FROM products p;  BEGIN  dbms\_lob.createtemporary(json\_data, TRUE);  file\_handle := utl\_file.fopen('ORACLE\_BASE', 'output.json', 'w', 32767);  for product\_row in c\_products loop  json\_data := json\_data || product\_row.json\_row || ',';  end loop;  json\_data := rtrim(json\_data, ',');  utl\_file.put\_raw(file\_handle, utl\_raw.cast\_to\_raw(json\_data));  utl\_file.fclose(file\_handle);  dbms\_lob.freetemporary(json\_data);  dbms\_output.put\_line('Exporting data to JSON file completed.');  EXCEPTION  WHEN OTHERS THEN  IF utl\_file.is\_open(file\_handle) THEN  utl\_file.fclose(file\_handle);  END IF;  dbms\_lob.freetemporary(json\_data);  RAISE;  END; |

Листинг 4.1 – Функция messageToJson

Для импорта в формат JSON была разработана программа import.js на языке javascript, работающая на nodejs и использующая модуль oracledb. Её запуск осуществляется через bat-файл.

Данная программа представлена в листинге 4.2.

|  |
| --- |
| const fs = require('fs');  const oracledb = require('oracledb');  const config = {      user: 'SYSTEM',      password: '123',      connectString: '//localhost:1521/CP\_YNS\_PDB'};  async function exportDataFromJson() {    try {      const jsonData = fs.readFileSync('input.json', 'utf8');      const products = JSON.parse(jsonData);      const connection = await oracledb.getConnection(config);      const value = await connection.execute('SELECT max(product\_id) FROM products');      let id = value.rows[0][0];      for (const product of products) {         const { product\_name, category\_id, price, description, image\_url, quantity } = product;          id += 1;         const insertQuery = `           INSERT INTO products           VALUES (:product\_id, :product\_name, :category\_id, :price, :description, :image\_url, :quantity)         `;        const bindParams = {          product\_id: id,          product\_name: product\_name,          category\_id: category\_id,          price: price,          description: description,          image\_url: image\_url,          quantity: quantity,        };        await connection.execute(insertQuery, bindParams);}      await connection.execute("commit");      console.log('Exporting data to the Oracle database completed.');      await connection.close();    } catch (error) {      console.error('Error exporting data:', error);    }  }  exportDataFromJson(); |

Листинг 4.2 – программа import.js

Данные функции могут пригодится в будущем для создания архивов, резервного копирования данных из БД и во многом другом.

# 5 Тестирование производительности

Тестирование производительности является важным этапом разработки, поскольку позволяет определить, насколько хорошо база данных может обрабатывать запросы и как быстро она может возвращать результаты.

Для тестирования производительности базы данных была выбрана таблица products, содержащая больше всего данных.

Результаты выполнения запроса к таблице указывают на значительные затраты времени и ресурсов, особенно при сканировании всей таблицы и применении фильтра. Время выполнения запроса составило 0.033 мс. Результаты запроса будут представлены на рисунке 5.1.

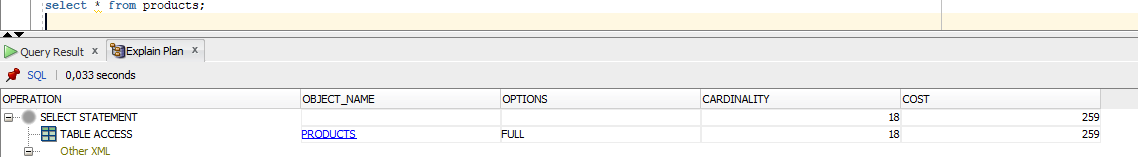


Рисунок 5.1 – Результат выполнения запроса

Результаты тестирования говорят о том, что структура таблицы оптимальна.

# 6 Описание технологии и ее применение в базе данных

В данном подразделе будут описаны технологии, использованные в процессе разработки приложения. В программном средстве была реализована технология резервного копирования и восстановления.

Резервное копирование и восстановление в Oracle — это процессы, связанные с созданием резервных копий данных и их восстановлением в случае потери или повреждения.

Резервное копирование в Oracle включает создание точных копий данных, включая таблицы, индексы, представления и другие объекты базы данных. Это обеспечивает защиту от возможной потери данных в случае сбоя, ошибки, атаки или других непредвиденных событий.

Были использованы сразу 2 варианта резервного копирования базы данных. В первом случае, была использована утилита rman. Первое что необходимо сделать для создания копии базы данных, это включить архивирование. Затем в утилите rman прописать «backup database;» для создания копии базы данных. Теперь при неисправностях работы базы данных мы всегда можем вернуть всё к состоянию, в котором находилась база данных на момент копирования.

Восстановление в Oracle происходит, когда необходимо восстановить базу данных из резервной копии. Этот процесс включает в себя восстановление файлов данных, журналов транзакций и других резервных копий с целью восстановления данных до последнего сохраненного состояния. После восстановления базы данных, можно приступить к применению журналов транзакций, чтобы восстановить данные до момента сбоя.

Во втором случае были использованы утилиты expdb и impdp. Первая позволяет создать резервную копию выбранных элементов или целую базу данных. В нашем случае мы создавали копию схемы в файл dmp. После чего её можно восстановить в другую пустую подключаемую базу данных.

# 7 Руководство пользователя

## 7.1 Сценарий использования базы данных

В данном разделе приведён пример тестового сценария для использования базы данных.

Если мы заходим в приложение впервые, то у нас роль гостя. Гость может только зарегистрировать новый аккаунт. После регистрации мы можем войти как пользователь.

Пользователь входит по своему имени и паролю. Он может просмотреть каталог товаров с помощью процедуры «display\_total\_amount\_of\_products». Понравившееся положить в корзину с помощью процедуры «add\_to\_cart». Это действие зарезервирует товар для него. Далее можно оформить заказ с помощью процедуры «make\_order».

Администратор также входит в систему по своему имени и паролю. Он может просмотреть статистику продаж с помощью таких процедур, как «view\_sales\_by\_period», «display\_popular\_products» и «display\_order\_information». В случае привоза товара на склад он проведёт пополнение в базе данных процедурой «add\_quantity\_product». А если появилась новая позиция товара добавит её процедурой «add\_product».

Это лишь одни из многих вариантов использования базы данных.

## 7.2 Вывод

В данном разделе были рассмотрены функциональные возможности базы данных, а также права доступа для пользователей с различными ролями. Было показано, что база данных позволяет пользователям оформлять заказы, искать товары, а также добавлять товары в корзину.

Кроме того, администратор имеет доступ к управлению товарами, категориями и заказами.

# Заключение

База данных является ключевым элементом любой современной организации, обеспечивая надежное хранение и управление информацией. В данной работе была поставлена задача разработки базы данных для магазина одежды с использованием технологии резервного копирования и восстановления в СУБД Oracle.

В процессе выполнения работы были использованы различные объекты, включая таблицы, процедуры, расписание и представления чтобы обеспечить структурированное хранение данных и своевременный доступ к ним. В результате, цель работы была успешно достигнута, и база данных готова к использованию. Были разработаны роли для управления доступом к данным и обеспечения безопасности.

Тестирование базы данных с использованием большого объема данных было положительным. Был реализован импорта и экспорт данных в формате JSON.

Одной из ключевых особенностей разработанной базы данных является технология резервного копирования и восстановления, что позволяет не волноваться за сохранность данных.

Таким образом, была успешно выполнена задача по разработке базы данных для магазина одежды на основе СУБД Oracle. Разработанная база данных позволяет хранить и управлять большим объемом данных, обеспечивает доступ к ним и предоставляет возможность импорта и экспорта данных в различных форматах.

# Список используемых источников

1. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com – Дата доступа: 15.10.2023.

2. DBMS\_XMLDOM [Электронный ресурс] / Режим доступа: https://docs.oracle.com/cd/B1930601/appdev.102/b14258/dxmldom.htm#i1076719 – Дата доступа: 20.11.2023.

3. Developing and Using Stored Procedures [Электронный ресурс] / Режим доступа: https://docs.oracle.com/cd/B2835901/appdev.111/b28843/tdddgprocedures.htm – Дата доступа: 26.11.2023.

4. Stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com – Дата доступа: 08.12.2023

5. Блинова, Е.А. Курс лекций по Базам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с.

# Приложение А

|  |
| --- |
| create table product\_categories (  category\_id number primary key,  category\_name varchar(200) not null);  create table products(  product\_id number primary key,  product\_name varchar(200) not null,  category\_id number,  price decimal(10,2) not null,  description varchar(255),  image\_url varchar(200),  foreign key (category\_id) references product\_categories(category\_id));  alter table products  add quantity number default 0;  alter table products modify price number(10,2);  create table customers (  customer\_id number primary key,  first\_name varchar2(50),  last\_name varchar2(50),  email varchar2(100),  phone\_number varchar2(20),  address varchar2(200));  alter table customers  add constraint uk\_first\_name unique (first\_name);  create table orders (  order\_id number primary key,  customer\_id number,  order\_date date,  total\_amount number,  foreign key (customer\_id) references customers(customer\_id));  alter table orders  add status\_id number;  alter table orders  add constraint fk\_order\_statuses foreign key (status\_id) references order\_statuses(status\_id);  create table composition\_of\_orders(  order\_id number,  product\_id number,  quantity number,  foreign key (order\_id) references orders(order\_id),  foreign key (product\_id) references products(product\_id));  alter table composition\_of\_orders  add composition\_id number primary key;  create table order\_statuses (  status\_id number primary key,  status\_name varchar2(50) not null);  create table user\_cart(  cart\_id number primary key,  customer\_id number,  foreign key (customer\_id) references customers(customer\_id));  create table cart\_items(  cart\_item\_id number primary key,  cart\_id number,  product\_id number,  quantity number,  foreign key (cart\_id) references user\_cart(cart\_id),  foreign key (product\_id) references products(product\_id)); |

Листинг 1 – Скрипты создания таблиц

|  |
| --- |
| create role USER\_DEFAULT;  grant create session to USER\_DEFAULT;  grant alter session to USER\_DEFAULT;  grant execute on make\_order to USER\_DEFAULT;  grant execute on undo\_order to USER\_DEFAULT;  grant execute on undo\_last\_order to USER\_DEFAULT;  grant execute on add\_product\_to\_cart to USER\_DEFAULT;  grant execute on remove\_from\_cart to USER\_DEFAULT;  grant execute on get\_cart to USER\_DEFAULT;  grant execute on view\_sales\_by\_period to USER\_DEFAULT;  grant execute on display\_popular\_products to USER\_DEFAULT;  grant execute on display\_total\_amount\_of\_products to USER\_DEFAULT;  grant execute on display\_product\_categories to USER\_DEFAULT;  grant execute on GetCustomerOrders to USER\_DEFAULT;  create user CP\_YNS\_USER\_1 identified by 123;  create user CP\_YNS\_USER\_2 identified by 123;  grant USER\_DEFAULT to CP\_YNS\_USER\_1;  grant USER\_DEFAULT to CP\_YNS\_USER\_2;  create role ADMIN\_DEFAULT;  grant create session to ADMIN\_DEFAULT;  grant alter session to ADMIN\_DEFAULT;  grant execute on make\_order to ADMIN\_DEFAULT;  grant execute on undo\_order to ADMIN\_DEFAULT;  grant execute on undo\_last\_order to ADMIN\_DEFAULT;  grant execute on add\_product\_to\_cart to ADMIN\_DEFAULT;  grant execute on remove\_from\_cart to ADMIN\_DEFAULT;  grant execute on add\_category to ADMIN\_DEFAULT;  grant execute on delete\_category to ADMIN\_DEFAULT;  grant execute on add\_product to ADMIN\_DEFAULT;  grant execute on delete\_product to ADMIN\_DEFAULT;  grant execute on update\_product to ADMIN\_DEFAULT;  grant execute on add\_quantity\_product to ADMIN\_DEFAULT;  grant execute on remove\_quantity\_product to ADMIN\_DEFAULT;  grant execute on view\_sales\_by\_period to ADMIN\_DEFAULT;  grant execute on display\_total\_amount\_of\_products to ADMIN\_DEFAULT;  grant execute on display\_popular\_products to ADMIN\_DEFAULT;  grant execute on display\_product\_categories to ADMIN\_DEFAULT;  grant execute on display\_order\_information to ADMIN\_DEFAULT;  grant execute on display\_customer\_information to ADMIN\_DEFAULT;  grant execute on GetCustomerOrders to ADMIN\_DEFAULT;  grant execute on change\_order\_status to ADMIN\_DEFAULT;  create user CP\_YNS\_ADMIN\_1 identified by 123;  grant ADMIN\_DEFAULT to CP\_YNS\_ADMIN\_1;  create role GUEST\_DEFAULT;  grant create session to GUEST\_DEFAULT;  grant execute on create\_new\_user to GUEST\_DEFAULT;  create user CP\_YNS\_GUEST identified by 123;  grant GUEST\_DEFAULT to CP\_YNS\_GUEST; |

Листинг 2 – Скрипты создание ролей и пользователей

|  |
| --- |
| create or replace procedure add\_category (p\_category\_name invarchar2) as  p\_category\_id number;  begin  select max(category\_id)  into p\_category\_id  from product\_categories;  insert into product\_categories (category\_id, category\_name)  values (p\_category\_id + 1, p\_category\_name);  commit;  exception  when no\_data\_found then  insert into product\_categories (category\_id, category\_name)  values (1, p\_category\_name);  commit;  when others then  dbms\_output.put\_line('Ошибка при выполнении процедуры');  rollback;  raise;  end;  create or replace procedure delete\_category (  p\_category\_name in varchar2  ) as  begin  delete from product\_categories  where category\_name = p\_category\_name;  commit;  exception  when no\_data\_found then  dbms\_output.put\_line('Категория ' || p\_category\_name || ' не найдена.');  rollback;  when others then  dbms\_output.put\_line('Ошибка при выполнении процедуры');  rollback;  raise;  end; |

Листинг 3 – Скрипты создания процедур управления категориями

|  |
| --- |
| create or replace procedure add\_order (  p\_customer\_id in number,  p\_total\_amount in number,  p\_status\_id in number  ) as  p\_order\_id number;  begin  select max(order\_id)  into p\_order\_id  from orders;  insert into orders  values (p\_order\_id + 1, p\_customer\_id, sysdate, p\_total\_amount, p\_status\_id);  commit;  exception  when no\_data\_found then  insert into orders  values (1, p\_customer\_id, sysdate, p\_total\_amount, p\_status\_id);  commit;  when others then  dbms\_output.put\_line('Ошибка при добавлении заказа.');  rollback;  raise;  end;  create or replace procedure delete\_order (  p\_order\_id in number  ) as  begin  delete from orders  where order\_id = p\_order\_id;  commit;  exception  when others then  dbms\_output.put\_line('Заказ ' || p\_order\_id || ' не найден.');  rollback;  raise;  end;  create or replace procedure add\_order\_composition (  p\_order\_id in number,  p\_product\_id in number,  p\_quantity in number  ) as  p\_composition\_id number;  begin  select max(composition\_id)  into p\_composition\_id  from composition\_of\_orders;  insert into composition\_of\_orders  values (p\_order\_id, p\_product\_id, p\_quantity, p\_composition\_id + 1);  commit;  update products  set quantity = quantity - p\_quantity  where product\_id = p\_product\_id;  commit;  exception  when no\_data\_found then  insert into composition\_of\_orders  values (p\_order\_id, p\_product\_id, p\_quantity, 1);  commit;  when others then  dbms\_output.put\_line('Ошибка при добавлении композиции.');  rollback;  raise;  end;  create or replace procedure delete\_order\_composition (  p\_order\_id in number  ) as  p\_quantity number;  p\_product\_id number;  begin  select quantity  into p\_quantity  from composition\_of\_orders  where order\_id = p\_order\_id;  select product\_id  into p\_product\_id  from composition\_of\_orders  where order\_id = p\_order\_id;  update products  set quantity = quantity + p\_quantity  where product\_id = p\_product\_id;  commit;  delete from composition\_of\_orders  where order\_id = p\_order\_id;  commit;  exception  when others then  dbms\_output.put\_line('Композиция ' || p\_order\_id || ' не найден.');  rollback;  raise;  end; |

Листинг 4 – Скрипты создания процедур управления заказами

|  |
| --- |
| create or replace procedure add\_product (  p\_product\_name in varchar,  p\_category\_id in number,  p\_price in number,  p\_description in varchar,  p\_image\_url in varchar,  p\_quantity in number  ) as  p\_product\_id number;  begin  select max(product\_id)  into p\_product\_id  from products;  insert into products  values (p\_product\_id + 1, p\_product\_name, p\_category\_id, p\_price, p\_description, p\_image\_url, p\_quantity);  commit;  exception  when no\_data\_found then  insert into products  values (1, p\_product\_name, p\_category\_id, p\_price, p\_description, p\_image\_url, p\_quantity);  commit;  when others then  dbms\_output.put\_line('Неверные данные');  rollback;  raise;  end;  create or replace procedure delete\_product (  p\_product\_name in varchar2  ) as  begin  delete from products  where product\_name = p\_product\_name;  commit;  exception  when others then  dbms\_output.put\_line('Товар ' || p\_product\_name || ' не найден.');  rollback;  raise;  end;  create or replace procedure update\_product (  p\_product\_name in varchar,  p\_price in number,  p\_description in varchar,  p\_image\_url in varchar,  p\_quantity in number  ) as  begin  update products  set price = p\_price,  description = p\_description,  image\_url = p\_image\_url,  quantity = p\_quantity  where product\_name = p\_product\_name;  commit;  exception  when others then  dbms\_output.put\_line('Продукт ' || p\_product\_name || ' не найден.');  rollback;  raise;  end;  create or replace procedure add\_quantity\_product (  p\_product\_name in varchar,  p\_quantity in number  ) as  begin  update products  set quantity = quantity + p\_quantity  where product\_name = p\_product\_name;  commit;  exception  when others then  dbms\_output.put\_line('Продукт ' || p\_product\_name || ' не найден.');  rollback;  raise;  end;  create or replace procedure remove\_quantity\_product (  p\_product\_name in varchar,  p\_quantity in number  ) as  current\_quantity number;  begin  select quantity  into current\_quantity  from products  where product\_name = p\_product\_name;  if (p\_quantity > current\_quantity) then  dbms\_output.put\_line('Количество не может превышать имеющееся');  return;  end if;  update products  set quantity = quantity - p\_quantity  where product\_name = p\_product\_name;  commit;  exception  when others then  dbms\_output.put\_line('Продукт ' || p\_product\_name || ' не найден.');  rollback;  raise;  end; |

Листинг 5 – Скрипты создания процедур управления товарами

|  |
| --- |
| create materialized view sales\_by\_period\_view  refresh complete on commit  as  select to\_char(o.order\_date, 'YYYY-MM') as month, p.product\_name, sum(co.quantity) as total\_quantity  from orders o  join composition\_of\_orders co on o.order\_id = co.order\_id  join products p on co.product\_id = p.product\_id  group by to\_char(o.order\_date, 'YYYY-MM'), p.product\_name  order by month desc;  create materialized view popular\_products  refresh complete on commit  as  select p.product\_name, sum(co.quantity) as total\_quantity  from composition\_of\_orders co  join products p on co.product\_id = p.product\_id  group by p.product\_name  order by total\_quantity desc;  create materialized view total\_amount\_of\_products  refresh complete on commit  as  select product\_name, quantity  from products; |

Листинг 6 – Скрипты создания представлений

|  |
| --- |
| create or replace procedure add\_product\_to\_cart (  p\_product\_name in varchar,  p\_quantity in number  ) as  p\_current\_quantity number;  p\_customer\_name varchar(200);  p\_cart\_item\_id number;  p\_cart\_id number;  p\_product\_id number;  p\_customer\_id number;  p\_count number;  p\_quantity\_in\_cart number;  begin  select quantity  into p\_current\_quantity  from products  where product\_name = p\_product\_name;  if (p\_quantity > p\_current\_quantity) then  dbms\_output.put\_line('Количество не может превышать имеющееся');  return;  end if;  select user into p\_customer\_name from dual;  select customer\_id  into p\_customer\_id  from customers  where first\_name = p\_customer\_name;  select max(cart\_item\_id)  into p\_cart\_item\_id  from cart\_items;  select cart\_id  into p\_cart\_id  from user\_cart  where customer\_id = p\_customer\_id;  select product\_id  into p\_product\_id  from products  where product\_name = p\_product\_name;  select count(\*)  into p\_count  from cart\_items  where product\_id = p\_product\_id and cart\_id = p\_cart\_id;  if (p\_count > 0) then  select quantity  into p\_quantity\_in\_cart  from cart\_items  where product\_id = p\_product\_id and cart\_id = p\_cart\_id;  update cart\_items  set quantity = quantity + p\_quantity  where product\_id = p\_product\_id and cart\_id = p\_cart\_id;  DBMS\_SCHEDULER.SET\_ATTRIBUTE(  name => 'remove\_from\_cart\_' || p\_product\_id || '\_' || p\_cart\_id,  attribute => 'job\_action',  value => 'BEGIN remove\_from\_cart(''' || p\_product\_name || ''', ' || (p\_quantity\_in\_cart + p\_quantity) || '); END;'  );  else  insert into cart\_items values (p\_cart\_item\_id + 1, p\_cart\_id, p\_product\_id, p\_quantity);  DBMS\_SCHEDULER.CREATE\_JOB(  job\_name => 'remove\_from\_cart\_' || p\_product\_id || '\_' || p\_cart\_id,  job\_type => 'PLSQL\_BLOCK',  job\_action => 'BEGIN remove\_from\_cart(''' || p\_product\_name || ''', ' || p\_quantity || '); END;',  start\_date => SYSTIMESTAMP + INTERVAL '1' HOUR,  repeat\_interval => '',  enabled => TRUE,  auto\_drop => TRUE  );  end if;  update products  set quantity = quantity - p\_quantity  where product\_id = p\_product\_id;  commit;  exception  when others then  dbms\_output.put\_line('Ошибка при оформлении заказа.');  rollback;  raise;  end;  create or replace procedure remove\_from\_cart (  p\_product\_name in varchar,  p\_quantity in number  ) as  p\_quantity\_in\_cart number;  p\_product\_id number;  p\_cart\_id number;  p\_customer\_name varchar(200);  begin  select user into p\_customer\_name from dual;  select customer\_id  into p\_cart\_id  from customers  where first\_name = p\_customer\_name;  select product\_id  into p\_product\_id  from products  where product\_name = p\_product\_name;  select quantity  into p\_quantity\_in\_cart  from cart\_items  where cart\_id = p\_cart\_id and product\_id = p\_product\_id;  if (p\_quantity > p\_quantity\_in\_cart) then  dbms\_output.put\_line('Количество не может превышать имеющееся');  return;  end if;  if (p\_quantity = p\_quantity\_in\_cart) then  delete from cart\_items  where cart\_id = p\_cart\_id and product\_id = p\_product\_id;  commit;    update products  set quantity = quantity + p\_quantity  where product\_id = p\_product\_id;  commit;  DBMS\_SCHEDULER.DROP\_JOB(  job\_name => 'remove\_from\_cart\_' || p\_product\_id || '\_' || p\_cart\_id  );  return;  end if;  update cart\_items  set quantity = quantity - p\_quantity  where cart\_id = p\_cart\_id and product\_id = p\_product\_id;  commit;  update products  set quantity = quantity + p\_quantity  where product\_id = p\_product\_id;  commit;  DBMS\_SCHEDULER.SET\_ATTRIBUTE(  name => 'remove\_from\_cart\_' || p\_product\_id || '\_' || p\_cart\_id,  attribute => 'job\_action',  value => 'BEGIN remove\_from\_cart(''' || p\_product\_name || ''', ' || (p\_quantity\_in\_cart - p\_quantity) || '); END;'  );  exception  when no\_data\_found then  rollback;  when others then  rollback;  raise;  end;  create or replace procedure get\_cart  as  p\_customer\_id number;  p\_customer\_name varchar(200);  begin  select user into p\_customer\_name from dual;  select customer\_id  into p\_customer\_id  from customers  where first\_name = p\_customer\_name;  for item in  (  select p.product\_name, ci.quantity, p.price, (ci.quantity \* p.price) as total\_price  from cart\_items ci  inner join products p on ci.product\_id = p.product\_id  where ci.cart\_id = (select cart\_id from user\_cart where customer\_id = p\_customer\_id)  ) loop  dbms\_output.put\_line('Product Name: ' || item.product\_name);  dbms\_output.put\_line('Quantity: ' || item.quantity);  dbms\_output.put\_line('Price: ' || item.price);  dbms\_output.put\_line('Total Price: ' || item.total\_price);  dbms\_output.put\_line('-----------------------');  end loop;  exception  when others then  dbms\_output.put\_line('Ошибка при выполнении процедуры');  rollback;  raise;  end; |

Листинг 7– Скрипты создания процедур работы с корзиной

|  |
| --- |
| create or replace procedure undo\_last\_order  as  p\_customer\_id varchar(200);  p\_order\_id number;  p\_customer\_name varchar(200);  begin  select user  into p\_customer\_name  from dual;  select customer\_id  into p\_customer\_id  from customers  where first\_name = p\_customer\_name;  select max(order\_id)  into p\_order\_id  from orders  where customer\_id = p\_customer\_id;  delete\_order\_composition(p\_order\_id);  delete\_order(p\_order\_id);  exception  when others then  rollback;  raise;  end;  create public synonym undo\_last\_order for SYSTEM.undo\_last\_order;  create public synonym change\_order\_status for SYSTEM.change\_order\_status;  -- изменение статуса заказа  CREATE OR REPLACE PROCEDURE change\_order\_status(  p\_order\_id IN orders.order\_id%TYPE,  p\_status\_name IN order\_statuses.status\_name%TYPE  )  IS  v\_status\_id order\_statuses.status\_id%TYPE;  BEGIN  SELECT status\_id INTO v\_status\_id  FROM order\_statuses  WHERE status\_name = p\_status\_name;  IF v\_status\_id IS NULL THEN  DBMS\_OUTPUT.PUT\_LINE('Указанный статус не найден.');  RETURN;  END IF;  IF p\_order\_id IS NULL THEN  DBMS\_OUTPUT.PUT\_LINE('Указанный заказ не найден.');  RETURN;  END IF;  UPDATE orders  SET status\_id = v\_status\_id  WHERE order\_id = p\_order\_id;  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Статус заказа успешно изменен.');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('указанный статус не найден.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Произошла ошибка. Изменение статуса заказа не выполнено.');  END;  create public synonym GetCustomerOrders for SYSTEM.GetCustomerOrders;  CREATE OR REPLACE PROCEDURE GetCustomerOrders  AS  v\_order\_id NUMBER;  v\_order\_date DATE;  v\_total\_amount NUMBER;  v\_status\_name VARCHAR2(50);  v\_product\_name VARCHAR2(200);  v\_product\_qty NUMBER;  p\_customer\_id number;  p\_customer\_name varchar2(200);  BEGIN  select user  into p\_customer\_name  from dual;  select customer\_id  into p\_customer\_id  from customers  where first\_name = p\_customer\_name;  FOR order\_rec IN (  SELECT o.order\_id, o.order\_date, o.total\_amount, os.status\_name, p.product\_name, co.quantity  FROM orders o  INNER JOIN order\_statuses os ON o.status\_id = os.status\_id  INNER JOIN composition\_of\_orders co ON o.order\_id = co.order\_id  INNER JOIN products p ON co.product\_id = p.product\_id  WHERE o.customer\_id = p\_customer\_id  ) LOOP  v\_order\_id := order\_rec.order\_id;  v\_order\_date := order\_rec.order\_date;  v\_total\_amount := order\_rec.total\_amount;  v\_status\_name := order\_rec.status\_name;  v\_product\_name := order\_rec.product\_name;  v\_product\_qty := order\_rec.quantity;  DBMS\_OUTPUT.PUT\_LINE('Order ID: ' || v\_order\_id || ', Order Date: ' || v\_order\_date || ', Total Amount: ' || v\_total\_amount || ', Status: ' || v\_status\_name || ', Product: ' || v\_product\_name || ', Quantity: ' || v\_product\_qty);  END LOOP;  END; |

Листинг 8 – Процедуры для работы с заказами пользователем

|  |
| --- |
| create or replace procedure create\_new\_user (  p\_username in varchar,  p\_usersurname in varchar,  p\_email in varchar,  p\_phone\_number in varchar,  p\_address in varchar,  p\_password in number  ) as  p\_id number;  begin  IF p\_email IS NOT NULL THEN  IF REGEXP\_LIKE(p\_email, '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$') = FALSE THEN  dbms\_output.put\_line('Неправильный формат электронной почты');  return;END IF;  END IF;  IF REGEXP\_LIKE(p\_phone\_number, '^\+[0-9]{1,3}[0-9]{9}$') = FALSE THEN  dbms\_output.put\_line('Неправильный формат телефона');return;  end if;  select NVL(max(customer\_id),0)  into p\_id  from customers;  insert into customers values (p\_id + 1, p\_username, p\_usersurname, p\_email, p\_phone\_number, p\_address);  execute immediate 'create user ' || p\_username || ' identified by ' || p\_password;  execute immediate 'grant USER\_DEFAULT to ' || p\_username;  commit;  exception  when others then  dbms\_output.put\_line('Ошибка при выполнении процедуры');  rollback;  raise;  end; |

Листинг 9 – Процедура создания нового пользователя

|  |
| --- |
| create or replace procedure ExportToJSON  as  json\_data CLOB;  file\_handle utl\_file.file\_type;  cursor c\_products is  SELECT json\_object('product\_id' VALUE p.product\_id,'product\_name' VALUE p.product\_name,'category\_id' VALUE p.category\_id,'price' VALUE p.price,'description' VALUE p.description,'image\_url' VALUE p.image\_url,'quantity' VALUE p.quantity) as json\_row  FROM products p;  BEGIN  dbms\_lob.createtemporary(json\_data, TRUE);  file\_handle := utl\_file.fopen('ORACLE\_BASE', 'output.json', 'w', 32767);  for product\_row in c\_products loop  json\_data := json\_data || product\_row.json\_row || ',';  end loop;  json\_data := rtrim(json\_data, ',');  utl\_file.put\_raw(file\_handle, utl\_raw.cast\_to\_raw(json\_data));  utl\_file.fclose(file\_handle);  dbms\_lob.freetemporary(json\_data);  dbms\_output.put\_line('Exporting data to JSON file completed.');  EXCEPTION  WHEN OTHERS THEN  IF utl\_file.is\_open(file\_handle) THEN  utl\_file.fclose(file\_handle);  END IF;  dbms\_lob.freetemporary(json\_data);  RAISE;  END; |

Листинг 10 – Процедура экспорта в JSON