Министерство науки и высшего образования Российской Федерации

Федеральное автономное образовательное учреждение

высшего образования

**Пермский национальный исследовательский**

**политехнический университет**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

**Отчет по курсовой работе**по дисциплине «Базы данных»

**Тема:**«Информационная система магазина автозапчастей» (вариант №70)

Выполнил

студент группы ИВТ-21-2б

Болотов Марк

Проверил

Доцент кафедры ИТАС

Петренко А.А.

Пермь, 2023

Реферат

Отчет 27 с., 10 рис., 1 табл., 4 источн., 3 прил.

БАЗА ДАННЫХ, МАГАЗИН АВТОЗАПЧАСТЕЙ, ВЕБ-ИНТЕРФЕЙС.

Объектом исследования является магазин автозапчастей.

Предметом исследования является система хранения и ведения информации о магазине автозапчастей.

Цель работы – проектирование базы данных информационной системы магазина автозапчастейи разработка интерфейса.

Задачи:

1) Анализ предметной области;

2) Проектирование базы данных;

3) Разработка web интерфейса;

4) Проведение тестов.

В ходе проведения работы были применены научные методы: анализ и моделирование.

По итогу работы была разработана реляционная база данных и веб приложение для взаимодействия с системой.

Результат проделанной работы обладает практической значимостью, так как может применяться для учёта и ведения хозяйства предприятия, а именно магазина автозапчастей.

**Оглавление**

[**Перечень сокращений и обозначений** 4](#_Toc135871565)

[**Введение** 5](#_Toc135871566)

[**1.** **Анализ предметной области** 6](#_Toc135871567)

[**1.1** **Заданная предметная область** 6](#_Toc135871568)

[**1.2** **Анализ** 7](#_Toc135871569)

[**2.** **Проектирование базы данных** 9](#_Toc135871570)

[**2.1** **Концептуальная модель** 9](#_Toc135871571)

[**2.2** **Логическая модель** 10](#_Toc135871572)

[**2.3** **Физическая модель** 13](#_Toc135871573)

[**2.4** **Разграничение прав доступа** 15](#_Toc135871574)

[**3.** **Разработка и реализация интерфейса** 16](#_Toc135871575)

[**4.** **Эксперименты** 22](#_Toc135871576)

[**Заключение** 26](#_Toc135871577)

[**Список использованных источников** 27](#_Toc135871578)

[**Приложение 1** 28](#_Toc135871579)

[**Приложение 2** 29](#_Toc135871580)

[**Приложение 3** 30](#_Toc135871581)

# **Перечень сокращений и обозначений**

В настоящем отчете применяют следующие сокращения и обозначения

База данных (БД) – совокупность данных, хранимых в соответствии c схемой данных, манипулирование которыми выполняют в соответствии с правилами

средств моделирования данных

Система управления базами данных (СУБД) – это комплекс программно-языковых средств, позволяющих создать базы данных и управлять ими.

Внешний ключ (FK, Foreign Key) – атрибут или группа атрибутов, однозначно идентифицирующих кортеж в отношении, на которое производится ссылка (внешнем отношении).

Первичный ключ (PK, Primary Key) – информация, уникально идентифицирующая строку таблицы.

RLS (Row Level Security) – механизм разграничения прав доступа использующийся в PostgreSQL.

# **Введение**

Базы данных являются неотъемлемой частью современного информационного общества. Они служат основой для хранения, организации и управления большим объемом данных, которые важны для различных организаций и предприятий. Вместе с тем, с постоянным увеличением объемов информации, возникают сложности в эффективном управлении и использовании этих данных. Поэтому понимание баз данных и умение работать с ними становятся все более востребованными навыками.

Цель работы: проектирование базы данных магазина автозапчастейи разработка интерфейса.

Задачи:

1) Анализ предметной области;

2) Проектирование базы данных;

3) Разработка web интерфейса;

4) Проведение тестов.

# **Анализ предметной области**

## **Заданная предметная область**

70(3). Информационная система магазина автозапчастей

Магазин розничной торговли осуществляет заказ запчастей в различных странах. Ведется статистика продаж, отражающая спрос на те или иные детали, и потребность магазина в них (сколько единиц, на какую сумму, какого товара продано за последнее время) и на ее основе составляются заказы (магазин) на требуемые товары. Выбор поставщика на каждый конкретный заказ осуществляют менеджеры магазина. В заказах (магазин) перечисляется наименование товара, количество. Если указанное наименование товара ранее не поставлялось, оно пополняет справочник номенклатуры товаров.

Поставщики бывают различных категорий: фирмы, непосредственно производящие детали, дилеры, небольшие производства, мелкие поставщики и магазины. В результате поставщики различных категорий имеют различающийся набор атрибутов. Фирмы и дилеры – это самые надежные партнеры, они могут предложить полный пакет документов, скидки, а главное – гарантию, чего не может сделать небольшое производство или мелкий магазин. У них же (фирмы и дилеры) закупается большой объем продукции. Небольшое производство – это низкие цены, но никакой гарантии качества. В мелких магазинах можно выгодно купить небольшое количество простых деталей, на которых сразу виден брак. Фирмы и дилеры поставляют детали на основе договоров, чего не делается для небольшого производства и мелкого магазина. В ходе маркетинговых работ изучается рынок поставщиков, в результате чего могут появляться новые поставщики и исчезать старые.

Когда ожидаются новые поставки, магазин собирает заявки от покупателей на свои товары. Груз приходит, производится его таможенное оформление, оплата пошлин, после чего он доставляется на склад в магазин. В первую очередь удовлетворяются заявки покупателей, а оставшийся товар продается в розницу.

В любой момент можно получить любую информацию о деталях, находящихся на складе, либо о поставляемых деталях. Детали хранятся на складе в определенных ячейках. Все ячейки пронумерованы. Касса занимается приемом денег от покупателей за товар, а также производит возврат денег за брак. Брак, если возможно, возвращается поставщику, который производит замену бракованной детали. Информация о браке (поставщик, фирма-производитель, деталь) фиксируется.

## **Анализ**

Информационная система магазина автозапчастей разрабатывается для одного магазина.

Магазин осуществляет продажу своих **товаров**. Товар — **запчасти**. **Товар** – это название из **справочника номенклатуры запчастей** и **поставщик,** цена продажи. **Справочник номенклатуры запчастей –** это наименование запчасти и **вид запчасти.**

**Вид запчасти: Двигательные детали, Тормозные детали, Ходовые детали, Кузовные детали, Электрические детали, Системы охлаждения, Системы выхлопа, Расходники, Системы передачи, Салонные детали…**

* *Существует «Гайка 5 мл» — это одна запись в справочнике номенклатуры запчастей. Эта же гайка, может быть, от разных поставщиков, и поэтому «Гайка 5 мл» от «Honda» и «От Михалыча» – разные товары.*

**Поставщик –** это имя поставщика, и его **категория**, статус. **Категория**: дилер, маленький магазин, фирмы и т. д. Каждая категория имеет свои **атрибуты** присущие этой категории. **Атрибуты**: наличие гарантии, скидки, пакет документов, низкие цены, наличие договора. Статус поставщика – это метка, которая показывает, доступен ли поставщик для оформления у него заказа магазина.

* *Фирмы – пакет документов, гарантия, скидки, поставка на основе договора. Небольшое производство – низкие цены, простые детали.*
* *Статус поставщика определяют в ходе «маркетинговых работ».* *Маркетинговые работы – это процесс появления новых поставщиков и исчезновение старых. Роль метки – это сохранение целостности базы данных. Если в результате «маркетинговых работ» решили, что поставщик больше не подходит (к примеру, из-за большого числа брака), то при простом удаление данных о поставщике из таблицы, нарушится взаимосвязь между другими данными (к примеру, раннее оформленными поставками или договорами).*

**Менеджер** – это тот, кто составляет поставки (заказы магазина), на основе статистики продаж.

**Поставка (заказ магазина)** – это **менеджер,** который оформил поставку, выбранный **поставщик** для поставки, запчасть, **договор** (не обязательное поле), сумма поставки, **статус поставки.**

* *Под “Товары” подразумевается 1 и более товаров.*
* *Если указанное наименование товара ранее не поставлялось, оно пополняет справочник номенклатуры запчастей до оформления поставки (заказа магазина).*
* *Договор может отсутствовать.*
* *Товары из справочника номенклатуры не могут быть от разных поставщиков. Если есть необходимость товара от разных поставщиков, то это оформляется как несколько разных поставок.*

Новые поставщики определяются в ходе неких “маркетинговых работ”. Маркетинговые работы – это процесс появления новых поставщиков и исчезновение старых. Результатами этих работ пользуются менеджеры как списком доступных поставщиков для оформления поставки (заказ магазина).

# **Проектирование базы данных**

## **Концептуальная модель**

Концептуальная модель выполняется в нотации Питере Чена. Проектирование базы данных начинается с выявления сущностей предметной области. При анализе предметной области были выделены следующие сущности:

* Товар
* Запчасть
* Вид запчасти
* Ячейка склада
* Поставщик
* Категория поставщика
* Атрибут категории
* Менеджер
* Заказ магазина (поставка)
* Договор
* Покупатель
* Заказ
* Заявка
* Брак
* Накладные расходы
* Денежная операция
* Характер денежной операции

Каждая сущность связана между собой хотя бы одной связью, кроме одной – накладные расходы. Полная картина всех взаимосвязей отражена в Приложение №1.

## **Логическая модель**

Следующий этап проектирования предполагает построение логической модели. Модель будет выполнена в нотации Мартина или как её называют по-другому – Crow’s Foot. В этой нотации необходимо из ранее выделенных сущностей, к каждой определить ключевые поля, а также показать их взаимодействие с другими полями сущностей. Но прежде всего следует определиться с СУБД. Для рассмотрения я взял следующие варианты: MySQL, MS SQL SERVER, PostgresSQL. Для удобства восприятия результаты представлены в виде таблицы.

Таблица №1 – сравнение СУБД

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | MySQL | MS SQL server | PostgreSQL |
| Исходный код | Открытый | Закрытый | Открытый |
| Функционал | Низкий | Средний | Широкий |
| Надежность | Средняя | Высокая | Высокая |
| Расширяемость | + | - | + |
| Совместимость со стандартами | Хорошее соответствие | Отклонение в пользу своих стандартов | Стремится к соответствию стандартам |

Была выбрана СУБД PostgreSQL в силу ряда причин:

* Открытый исходный код: PostgreSQL является свободной и открытой системой управления базами данных (СУБД), что означает, что можно получить доступ к полному исходному коду и вносить изменения в систему согласно своим потребностям. Это позволяет более гибко настраивать и расширять функциональность PostgreSQL.
* Богатый набор функций: PostgreSQL предлагает широкий спектр функций и возможностей, которые позволяют разработчикам создавать сложные и мощные базы данных. Он поддерживает расширенные функции, такие как географическая информационная система (GIS), полнотекстовый поиск, JSON-обработка и многое другое.
* Надежность и целостность данных: PostgreSQL славится своей надежностью и поддержкой целостности данных. Он обеспечивает транзакционную безопасность, контроль целостности и возможности восстановления после сбоев. Это важные аспекты при работе с базами данных, особенно в профессиональной среде.
* Расширяемость и поддержка: Сообщество PostgreSQL активно развивается и поддерживает эту СУБД. Можно найти множество дополнительных модулей и расширений, разработанных сообществом, которые могут значительно расширить возможности PostgreSQL.
* Совместимость со стандартами: PostgreSQL стремится к соответствию стандартам ANSI SQL и SQL:2008. Это означает, что многие запросы и операции, которые вы изучите в контексте баз данных, должны быть применимы и в PostgreSQL. Это также способствует переносимости кода между различными СУБД, если вам в будущем потребуется работать с другими системами.

Так же на этом этапе была проведена нормализация до 3 нормальной формы. Это в свою очередь означает, что проектируемая база данных имеет свойства предыдущих форм включая третью нормальную форму. А именно:

* От первой нормальной формы – все атрибуты в таблице являются атомарными, то есть они не могут быть разделены на более мелкие части.
* Вторая нормальная форма (2NF): В этой форме все атрибуты в таблице должны зависеть от ее первичного ключа. Если в таблице есть составной первичный ключ, каждый атрибут должен зависеть от всего составного ключа, а не от его частей.
* Третья нормальная форма (3NF): В этой форме все атрибуты в таблице должны зависеть только от ее первичного ключа или других атрибутов, которые уже являются ключами. Если атрибуты зависят от неключевых атрибутов, то эти неключевые атрибуты должны быть вынесены в отдельную таблицу.

Полученная логическая модель в ходе привидения к нормализации отражена в Приложение №2.

## **Физическая модель**

Физическая модель выполнена в нотации Crow’s Foot. Модель составлена исходя из выбранной СУБД. Модель использует термины, названия и типы данных PostgreSQL для описания всех таблиц и полей базы данных.

Также на этом этапе к некоторым таблицам снизу отмечены короткие описания триггеров и триггерных функций, которые связаны с таблицей.

Рассмотрим таблицу «Заказы». К этой таблице относиться триггер «add\_trans\_order» и относящаяся к нему триггерная функция «add\_trans\_order()». Этот триггер поставлен на показатель «…BEFORE INSERT…», означает, что триггерная функция выполниться раньше, вставки, то есть дословно переводится, как «…ДО ВСТАВКИ…». Для заказа необходима транзакция, отражающая на какую сумму товара находиться в заказе. Именно триггерная функция создаёт новую запись в таблице с денежными транзакциями, тем самым связывая между собой заказ и денежную транзакцию.

Абсолютно такая же ситуация происходит и с таблицей «Заявки». Триггер «add\_trans\_request» и триггерная функция «add\_trans\_request()», выполняют такой же функционал.

Немного другая картина с вставкой значения для таблицы «Брак». В таблице брак присутствует ещё одно поле «transaction\_past» для ссылки на предыдущую транзакцию, указывающий в какой покупке был обнаружен бракованный товар, а поле «transaction» указывает на денежную операцию возврата средств за некачественный товар. Поэтому функционал триггерной функции немного отличается, наличием проверки на владение предыдущей транзакцией (ваша/не ваша покупка).

Следующие виды триггеров – это триггеры для добавления товаров в таблицы «Заказ», «Заявки», «Брак». Для таблиц «Заказ», «Заявки» перед вставкой проверяется на наличие товара, а также в поля заносятся/обновляется/формируется информация о:

* Выбранное количество
* Цена в момент продажи
* Уменьшается доступное кол-во
* Пересчитывается полная сумма за заказ/заявку

\*Для «Брака» дополнительно проводиться проверка, был ли этот товар в предыдущей заявке/заказе.

Получившиеся схема физической модели отражена в Приложение №3.

## **Разграничение прав доступа**

В результате анализа предметной области были выделены следующие роли, взаимодействующие с СУБД:

* «group\_user» – покупатель. Имеет права на формирование заказов/заявок/браков. Просматривать имеет право только свои заказы/заявки/браки. Имеет доступ к просмотру всего ассортимента товаров. Имеет право редактировать информацию только о себе. Вход по паролю.
* «group\_manager» – менеджер. Имеет право на просмотр информации всех таблиц. Не имеет право создавать или как-то редактировать заказы/заявки/браки. Вход по паролю.
* «group\_admin» – администратор. Имеет доступ к просмотру, вставке, изменению по всем таблицам. Однако есть запрет на вставку и редактирование на таблицу «Денежные транзакции».

Помимо раздачи привилегий, продумана защита по столбцам и строкам. Реализована за счёт встроенного в PostgreSQL механизма «ROW LEVEL SECURITY» или по-другому «защита на уровне строк». Представляет из себя более гибкий аналог «VIEW». «VIEW» является статическим объектом базы данных и может быть определен заранее с фиксированными условиями фильтрации данных. RLS предоставляет возможность динамического управления доступом к данным на основе контекста пользователя, переменных сессии или параметров запроса, что активно используется в реализованной базе данных. Это позволяет реализовывать динамические политики безопасности и учитывать изменения в состоянии пользователя или среды выполнения.

# **Разработка и реализация интерфейса**

В качестве интерфейса был выбран web-интерфейс в силу его доступности, простоты, гибкости использования, наличие широких возможностей разработки.

Для базы web-интерфейса использован фреймворк «Flask», в силу поддержки СУБД PostgreSQL, а также легкой разработкой web-интерфейса.

Реализованный вариант интерфейса поддерживает все сценарии взаимодействия с группой роли «group-user». Реализованы следующие сценарии:

* Авторизация. Без авторизации покупатель не сможет использовать другой функционал. Рисунок №1

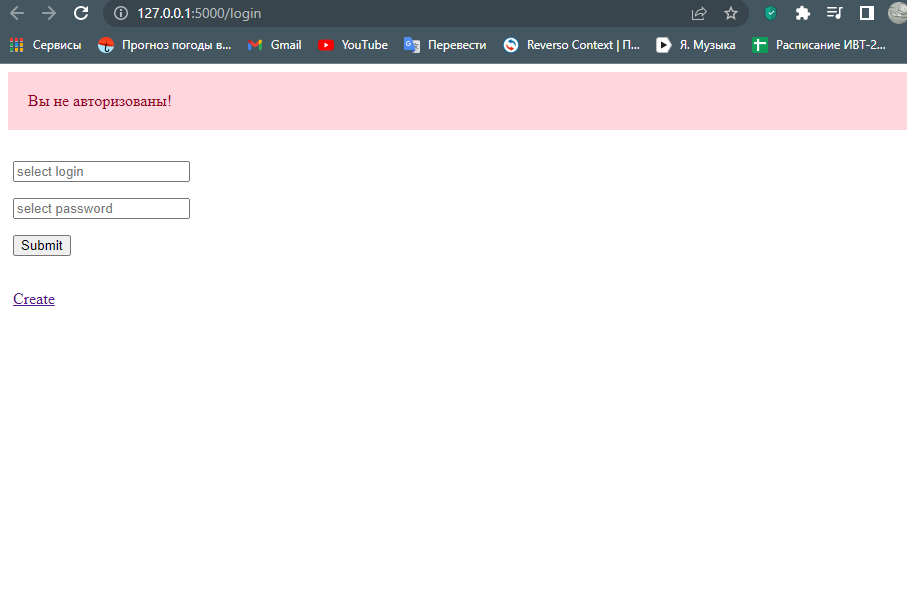


Рисунок №1 – окно авторизации пользователя

* Создание нового пользователя Рисунок №2.

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

Автоматически созданное описание

Рисунок №2 – окно регистрации

* Профиль. Реализована возможность смены информации о себе. Просмотр оформленных заказов/заявок/браков. Просмотр транзакций. Возможность выйти из системы. Рисунок №3

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

Автоматически созданное описание

Рисунок №3 – окно профиля

* Ассортимент Рисунок №4

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рисунок №4 – окно ассортимента товаров

* Просмотр информации о заказах Рисунок №5

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок №5 – окно подробной информации о заказах

* Формирование заказа Рисунок №6

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Рисунок №6 – окно добавления товаров в заказ

* Просмотр информации о заявках Рисунок 7



Рисунок №7 – окно подробной информации о заявках

* Формирование заявки Рисунок №8

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Рисунок №8 – окно формирования заявки

* Просмотр информации о браках Рисунок №9

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок №9 – окно подробной информации о браке

* Формирование брака Рисунок №10. Реализован Фильтр для более удобного поиска по прошлым заказам/заявкам, также добавлено окно прошлых транзакций не включающий транзакции за брак.

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Рисунок №10 – окно формирования брака

# **Эксперименты**

* Проверка триггерной функции на привязку транзакции к заказу

При создание нового заказа, должна быть создана и привязана новая транзакция. Таблицы «Транзакции» и «Заказы» до добавление нового заказа Рисунок №11 и Рисунок №12:

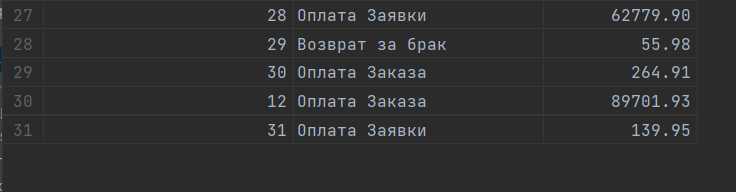


Рисунок №11 – Таблицы «Транзакции» до вставки

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Рисунок №12 – Таблица «Заказы» до вставки

Изменяем таблицу заказы, путем добавления нового заказа, через консоль командой

1. **INSERT** **INTO** orders\_customers(customer\_login) **VALUES** ("current\_user"());

Результат работы после вставки в таблице «Транзакции» и «Заказы» Рисунок №13 и Рисунок №14:

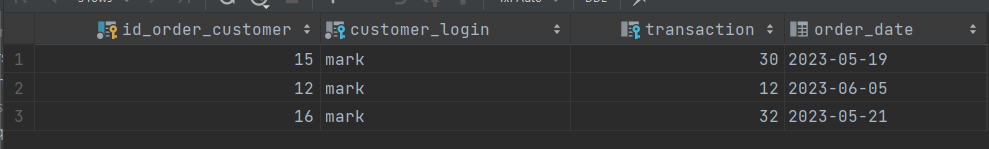


Рисунок №13 – Таблица «Заказы» после вставки

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок №14 – Таблицы «Транзакции» после вставки

На рисунках видно, что после вставки в таблицу «Заказы», была добавлена и привязана новая запись из таблицы «Транзакции».

* Проверка интерфейса

Вход выполнен под пользователем «mark». Вид информации о профиле из интерфейса и консоли Рисунок №15 и Рисунок №16:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок №15 – Вид на информацию о покупатели из интерфейса

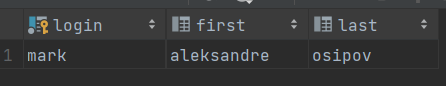


Рисунок №16 – Вид на информацию о покупатели из консоли

Имя и фамилия были изменены через страницу интерфейса. Результат после изменения в интерфейсе и консоли:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок №17 – Результат после изменения информации из интерфейса

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок №18 – Результат после изменения информации из консоли

Исходя из результатам, можно сделать вывод, что интерфейс работает корректно.

* Проверка запроса «6) Получите среднее число продаж на месяц по любому виду деталей»

**Запрос для решения:**

1. **SELECT** **EXTRACT**(**MONTH** **FROM** order\_date) **AS** sale\_month, **AVG**(count) **AS** average\_sales
2. **FROM** orders\_customers
3. **JOIN** orders\_customers\_products ocp **ON** orders\_customers.id\_order\_customer = ocp.order\_customer
4. **JOIN** products p **ON** p.id\_product = ocp.product
5. **JOIN** machine\_parts mp **ON** mp.id\_machine\_part = p.machine\_part
6. **JOIN** types\_machine\_part tmp **ON** mp.type\_machine\_part = tmp.id\_type\_machine\_part
7. **WHERE** tmp.name\_type = 'Ходовые детали'
8. **GROUP** **BY** sale\_month;

**Результат работы запроса Рисунок №19:**

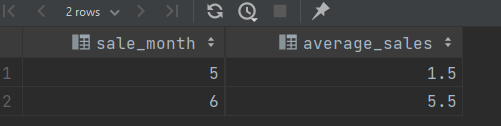


Рисунок №19 – Результат работы запроса

Для проверки правильности запроса проверяются все позиции, где фигурирует данный вид товара. Данную проверку можно выполнить следующим запросом:

1. **SELECT** count, oc.order\_date **FROM** orders\_customers\_products
2. **JOIN** products p **ON** p.id\_product = orders\_customers\_products.product
3. **JOIN** machine\_parts mp **ON** mp.id\_machine\_part = p.machine\_part
4. **JOIN** types\_machine\_part tmp **ON** tmp.id\_type\_machine\_part = mp.type\_machine\_part
5. **JOIN** orders\_customers oc **ON** oc.id\_order\_customer = orders\_customers\_products.order\_customer
6. **WHERE** tmp.name\_type = 'Ходовые детали'

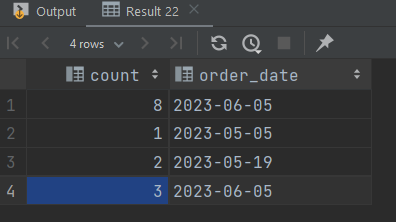


Рисунок №20 – Результат работы проверки

Исходя из проверки запрос работает правильно.

# **Заключение**

В ходе данной работы была смоделирована и разработана база данных магазина автозапчастей. Были выделены ключевые аспекты работы, спроектированы три последовательные модели базы данных, выбрана наиболее подходящая для решения данной задачи СУБД, определены разграничения прав доступа. Также было разработан web интерфейс для работы с базой данных. Вся система была протестирована.

Цель работы была достигнута. Все обозначенные ранее задачи были выполнены.

Данная работа имеет потенциал для дальнейших доработок. Можно выделить следующие векторы развития:

* Оптимизация запросов.
* Доработка web интерфейса.

# **Список использованных источников**

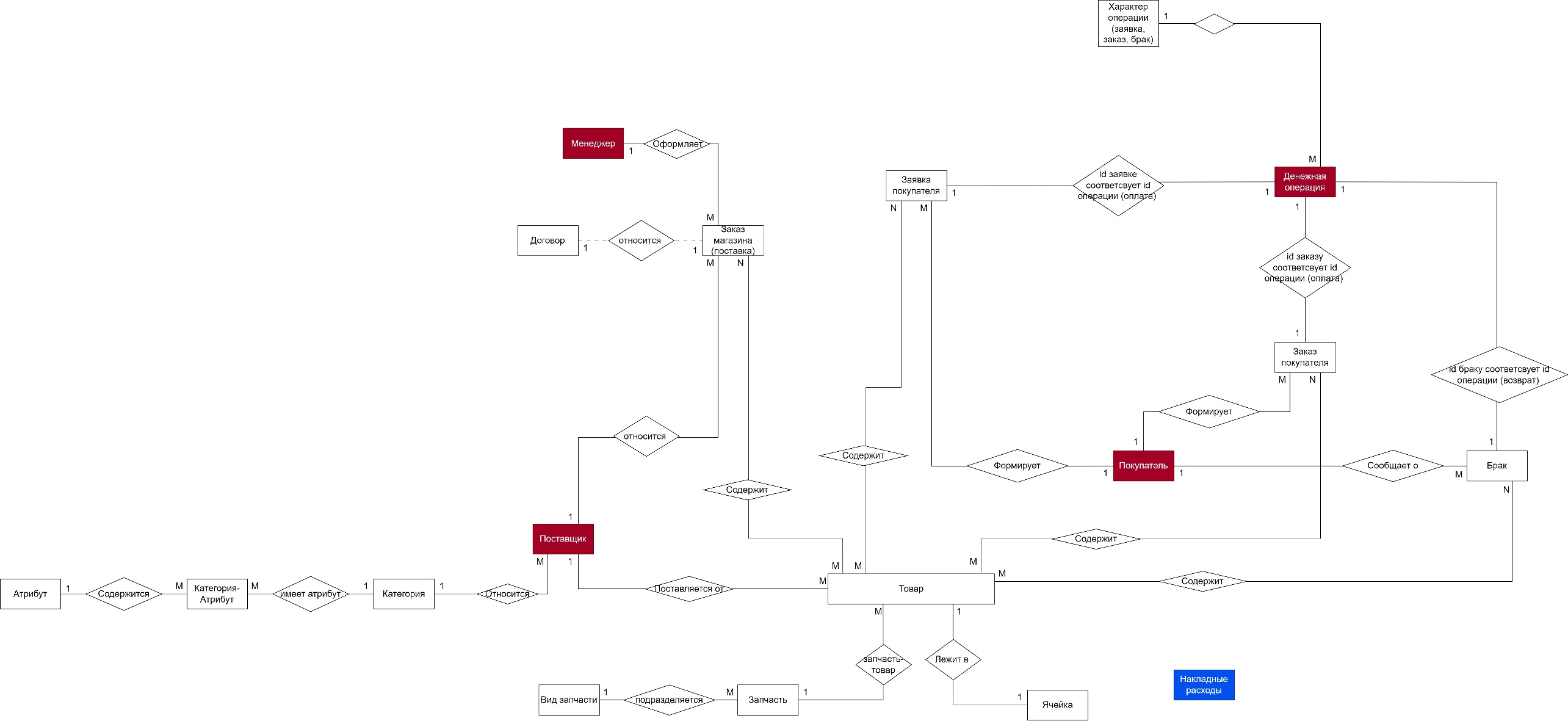
1. Чен П. Модель "сущность-связь" – шаг к единому представлению данных // СУБД. № 3. 1995.
2. Кузин, А.В. Базы данных: Учебное пособие для студ. высш. учеб. заведений / А.В. Кузин, С.В. Левонисова. - М.: ИЦ Академия, 2012. – 320 c.
3. Копырин А. С. Базы данных : практикум для студентов направления

09.03.03 «прикладная информатика». Сочи : СГУ, 2019. 106 с. URL: https://elib.pstu.ru/Record/lanRU-LAN-BOOK-147663 (дата обращения: 18.04.2023).

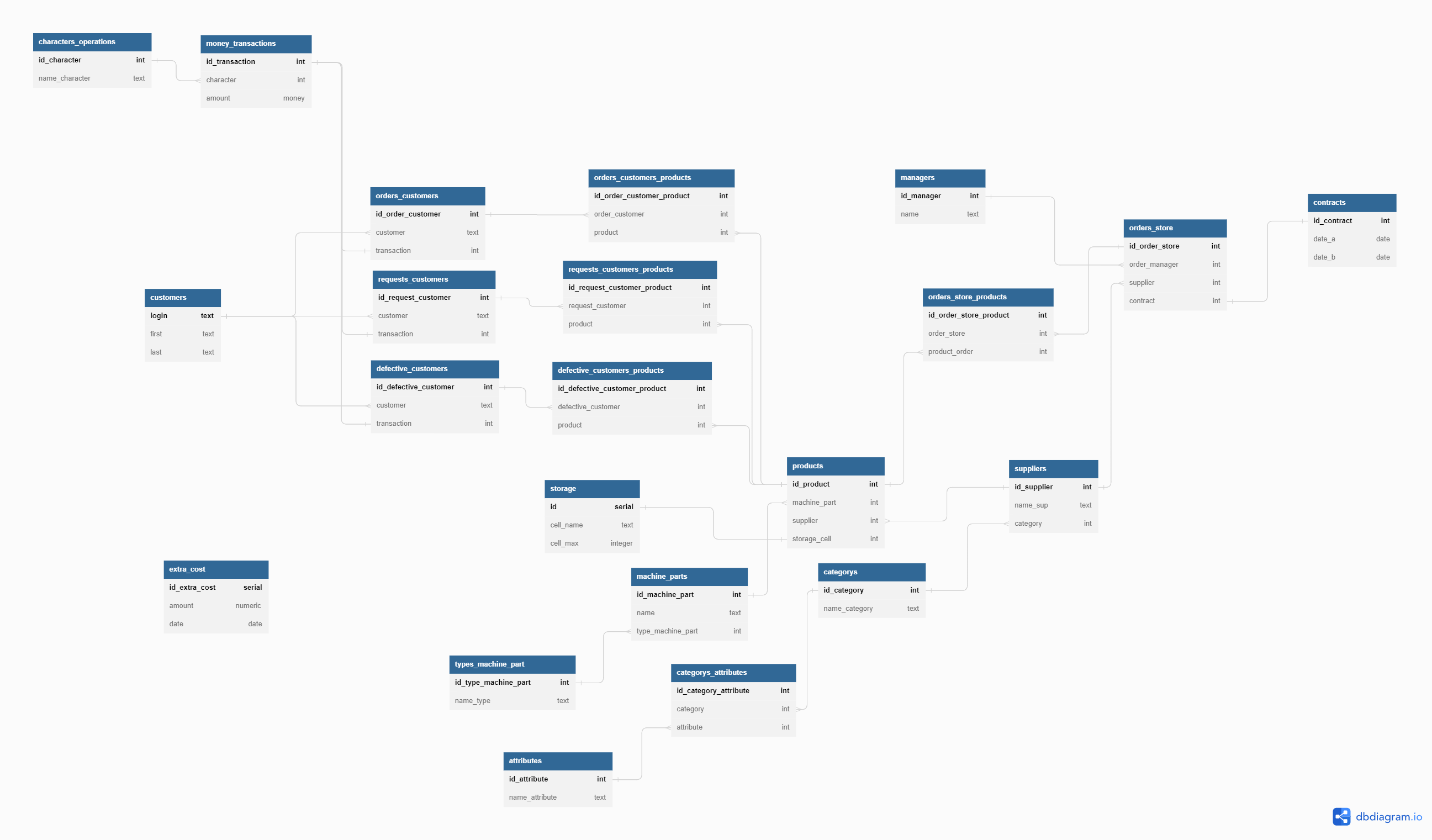
1. Дейт К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ.

— М.: Издательский дом "Вильяме", 2005. — 1328 с.: ил. — Парал. тит. англ.

# **Приложение 1**

****

# **Приложение 2**



# **Приложение 3**

