

**Содержание**

[Введение 4](#_Toc166175523)

[1. Анализ предметной области 5](#_Toc166175524)

[1.1 Описание предметной области 5](#_Toc166175525)

[1.2 Постановка задачи 9](#_Toc166175526)

[2. Концептуальное и логическое проектирование 10](#_Toc166175527)

[2.1 Сущности предметной области 10](#_Toc166175528)

[2.2 Логическое проектирование 13](#_Toc166175529)

[3. Физическое проектирование 16](#_Toc166175530)

[4. Создание базы данных 18](#_Toc166175531)

[5. Заполнение базы данных 19](#_Toc166175532)

[6. Основные запросы к базе данных 19](#_Toc166175533)

[Заключение 19](#_Toc166175534)

[Список использованной литературы 20](#_Toc166175535)

[Приложение 1 21](#_Toc166175536)

# Введение

В современном мире розничная торговля играет значительную роль в экономике. Она обеспечивает население товарами и услугами, способствуя удовлетворению повседневных потребностей. Однако для успешного функционирования розничных торговых предприятий необходимо эффективно управлять продажами товаров. В этом контексте проектирование и реализация базы данных для учёта продажи товаров становится актуальной задачей.

Целью данной курсовой работы является проектирование и реализация базы данных для учёта продажи товаров в продуктовом магазине. В ходе работы будут рассмотрены особенности розничной торговли, требования к информационным системам и методы проектирования баз данных. Также будет проведён анализ существующих подходов к проектированию и реализации баз данных, выбраны подходящие инструменты и технологии для решения поставленной задачи.

Для достижения поставленной цели необходимо решить следующие задачи:

* Изучить особенности розничной торговли и требования к информационным системам.
* Разработать модель базы данных для учёта продажи товаров в продуктовом магазине.
* Выбрать подходящие инструменты и технологии для реализации базы данных.
* Реализовать спроектированную базу данных и протестировать ее.

# 1. Анализ предметной области

## 1.1 Описание предметной области

Существуют заводы, производящие продукты, продаваемые в магазине. Продукты от производителей в магазин доставляют компании-поставщики. Под поставкой понимается партия товара, привезенная в одном грузовом автомобиле. При поставке товаров в магазин составляются товарные накладные для описания поставленных товаров.

В накладной ТОРГ-12 предусматриваются все необходимые поля для оформления поставки.

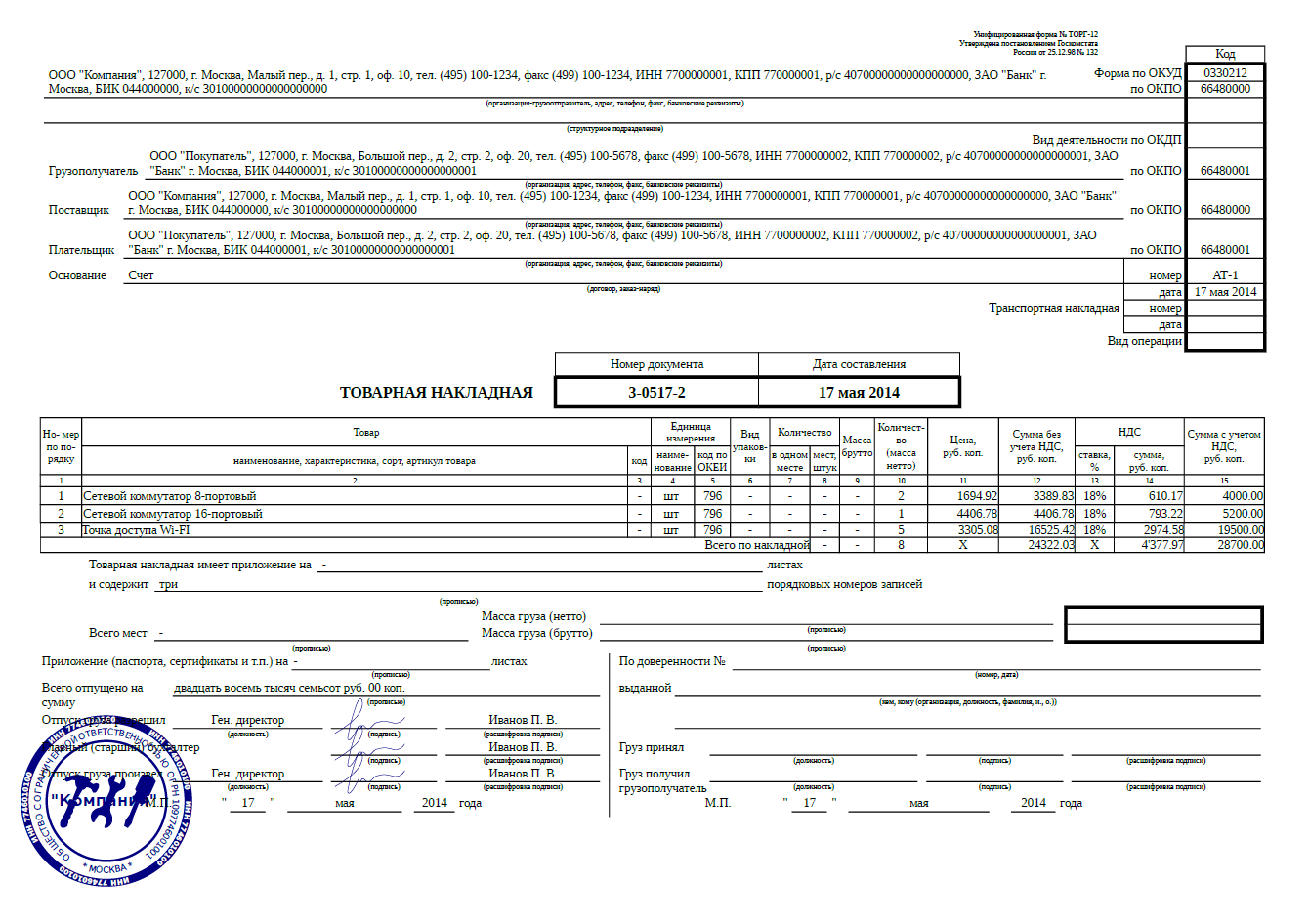


Рисунок 1 – пример заполненной накладной ТОРГ-12

В данной накладной учитываются следующие данные:

* Поставщик
* Получатель
* Дата поставки
* Наименования товаров
* Количества товара
* Цена товара до и после добавления НДС

Каждый товар имеет фиксированный срок годности и наценку.

Розничная цена товара формируется следующим образом: магазин устанавливает наценку на определенный товар, которая прибавляется к его закупочной цене. В один день нельзя продать товар с одним и тем же артикулом по разной цене, поэтому магазин устанавливает актуальную розничную цену исходя из максимальной закупочной цены товара в последних поставках и ситуации на рынке. Если в магазин поступает партия товара по меньшей цене, чем предыдущая, то этот товар начнет продаваться по меньшей розничной цене только после принудительного уменьшения наценки администрацией магазина.

При покупке товара печатается чек с уникальным номером. В чеке перечислены товары, их количества, стоимости, суммарная цена покупки.

Также необходимо вести учёт списанных товаров. Товары с истекшим сроком годности необходимо списывать.

Срок годности считается истекшим, если в дату на товаре, указанную как крайнюю для употребления, он не был куплен до закрытия магазина.

При списании товаров формируется акт о списании товара, например, ТОРГ-16.

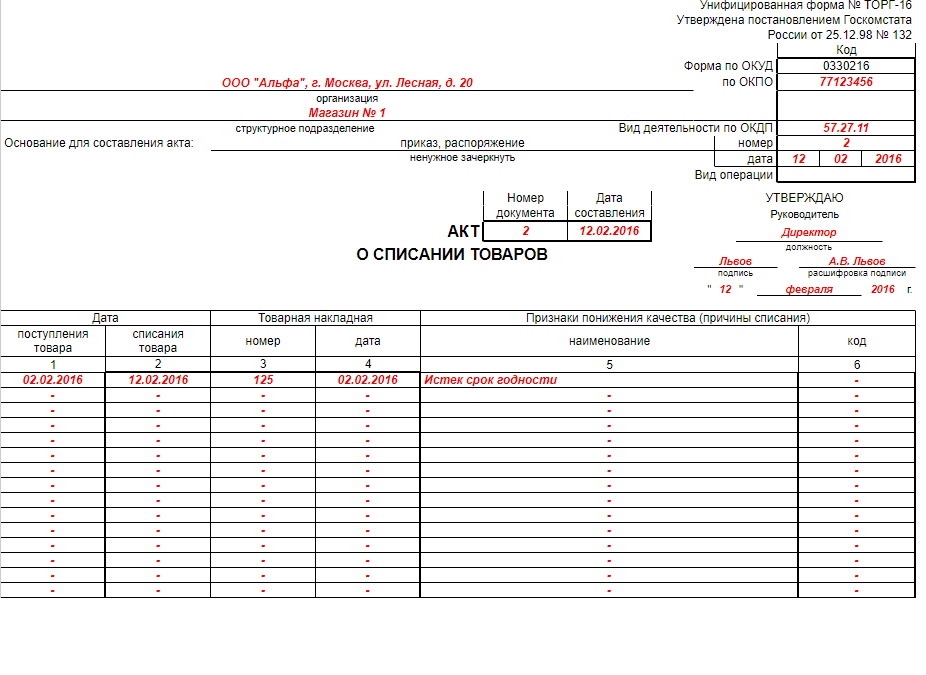


Рисунок 2 – пример заполненного акта о списании товаров ТОРГ-16. Лицевая сторона

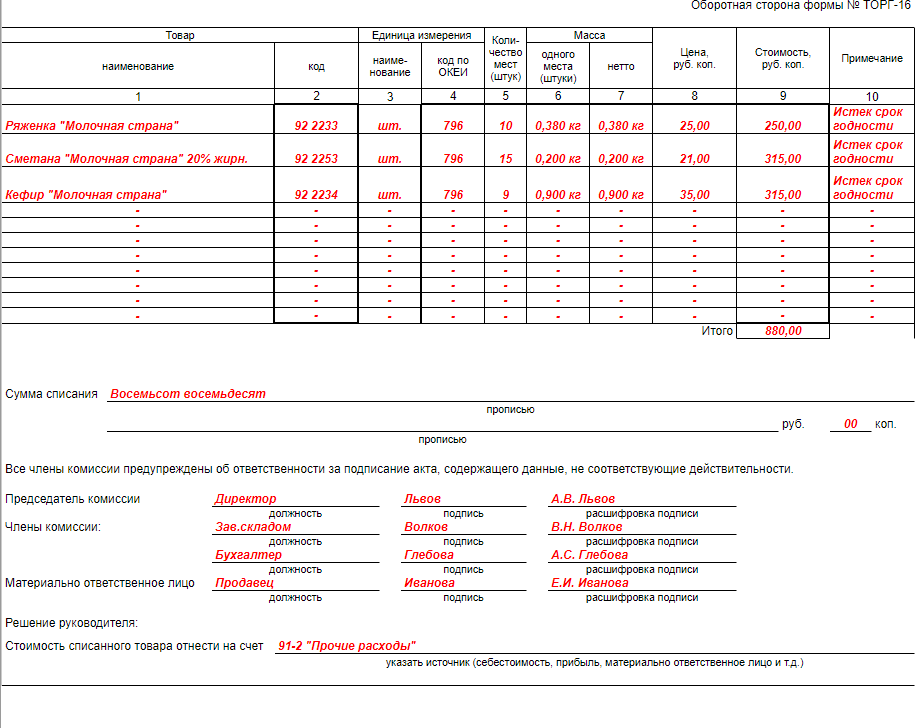


Рисунок 3 – пример заполненного акта о списании товаров ТОРГ-16. Оборотная сторона

В этой форме указаны следующие данные:

* Дата составления акта о списании товаров
* Дата поставки и списания товаров
* Товарная накладная, по которой сверялась поставка товара
* Наименования товаров
* Количество товаров
* Масса товаров
* Суммарная стоимость списанных товаров

## 1.2 Постановка задачи

Таким образом, исходя из описания предметной области, можно сформировать список требований к разрабатываемой базе данных:

* Необходимо хранить информацию обо всех поставщиках, с которыми работает магазин
* Необходимо вести учёт поставок
* При учёте поставок необходимо сохранять информацию о дате поставки, количестве и стоимости определенного товара
* Необходимо вести учёт списанных товаров
* При учёте списанных товаров необходимо сохранять информацию о поставке, с которой поступил списываемый товар, количестве и стоимости списываемого товара
* Необходимо вести учёт покупок товара
* При учёте покупок сохранять информацию о поставках, с которыми прибыли купленные товары, количестве и стоимости купленных товаров

К дополнительным требованиям можно отнести наличие в базе данных триггеров, призванных упростить ее заполнение.

# 2. Концептуальное и логическое проектирование

## 2.1 Сущности предметной области

Сущность предметной области — это абстракция реального объекта, группы однотипных объектов или концептуального понятия предметной области, характеризуемая набором существенных характеристик (данных, атрибутов), связанных с проектируемой программной системой.

В рассматриваемой предметной области можно выделить следующие сущности:

* Поставщик
* Поставка
* Товар
* Продажа
* Списание товара

Это основные объекты и абстракции реального мира, необходимые для реализации проектируемой базы данных. Каждая сущность содержит особый набор атрибутов. Определим атрибуты для каждой из перечисленных сущностей.

Поставщик характеризуется своим местоположением и наименованием компании. Остальные характеристики поставщика в рамках рассматриваемой предметной области не важны, так что именно эти характеристики и будут являться атрибутами сущности.

Товар характеризуется своим названием, массой, видом упаковки (например, банка либо бутылка для напитков), сроком годности и актуальной розничной ценой. Так же в числе атрибутов будет содержаться наценка магазина на товар, которую определяет магазин.

Поставка будет содержать информацию о количестве поставленных товаров, закупочную цену товаров, а также дату.

Покупка будет характеризоваться датой, количеством проданных товаров и суммой полученных денег.

В сущности «Списание товара» свойствами будут являться дата списания и количество списанного товара. Для получения данных о товарах и о датах их поставки она должна быть связана как с сущностью «Поставка» так и с сущностью «Товар».

Также товар может характеризоваться различными параметрами, которые уникальны для каждой категории продуктов. В накладных эти характеристики обычно записываются в колонке с названием товара (например, процент жирности молочных продуктов). Удобно хранить эти данные в отдельной сущности. Введем для этого сущность «Характеристика» с атрибутом «Название характеристики» и «Значение характеристики».

Основываясь на описаниях предметной области и сущностей построим ER-диаграмму в нотации Чена для визуализации концептуальной модели (Рисунок 4).

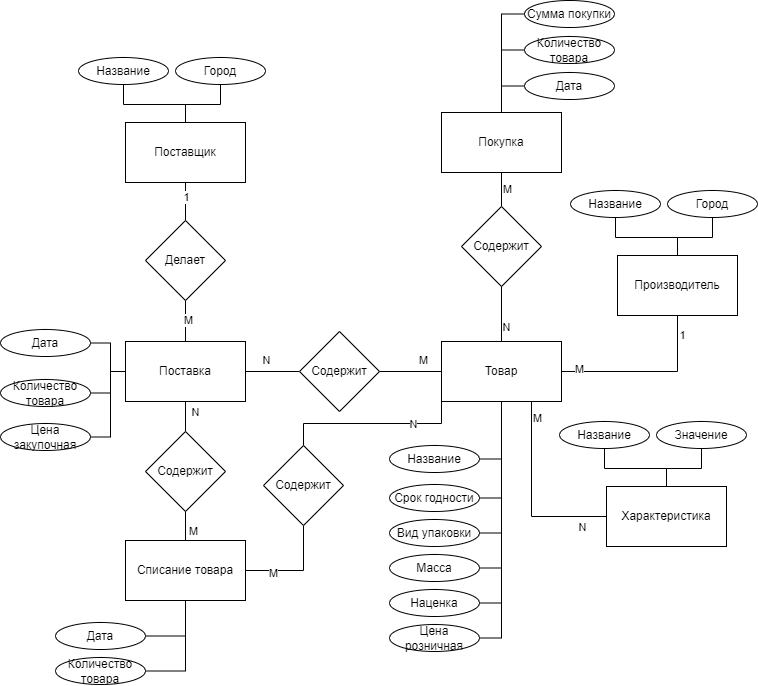


Рисунок 4 – ER-Диаграмма в нотации Чена.

## 2.2 Логическое проектирование

Первичный ключ в базе данных — это столбец или группа столбцов, которые однозначно определяют каждую запись в таблице. Он используется для обеспечения уникальности и целостности данных, а также для установления связей между таблицами.

В свою очередь, внешний ключ — это поле (или набор полей) в одной таблице, которое ссылается на первичный ключ в другой таблице. Благодаря внешним ключам реализуется связь между таблицами в реляционных базах данных.

Выделим первичные ключи для сущностей проектируемой базы данных. Про сущности проектируемой базы данных можно сказать, что в них не выходит выделить один атрибут, который бы мог стать первичным ключом, а выделение сложных первичных ключей из нескольких атрибутов создаст крайне большие и неудобные в запросах ключи. Таким образом, для этих сущностей необходимо выделить суррогатный ключ, который представляет из себя численный индекс каждой записи в таблице.

Товар в поставке может исчисляться разными единицами измерения. Например, это могут быть жидкости в литрах, миллилитрах, это могут быть овощи в килограммах, приправы в граммах и т.д. Для учёта возможных единиц измерения введем для них новую сущность «Единица измерения». Таким же образом введем новую сущность для видов упаковок товара.

Сущности «Поставщик» и «Производитель» очень похожи между собой, поэтому целесообразно будет объединить их в сущность «Компания», в которую будет добавлен атрибут «Специализация».

В концептуальной модели базы данных присутствуют связи типа «многие ко многим». Так, например, в одной поставке могут быть несколько наименований товаров, а точнее практически всегда поставка будет представлять из себя набор из разных товаров производителя. Но, в то же время, один конкретный товар может поставляться в разное время и разными поставщиками, то есть находиться в разных поставках. Для концептуальной схемы такой тип связи является допустимым, однако в логической и физической моделях базы данных такой связи быть не может, так как она не является однозначной. Для того, чтобы избавиться от связей типа «многие ко многим», в базу данных вводятся новые сущности. В проектируемую базу данных требуется добавить:

* Сущность «Товар в поставке», чтобы описать для каждого отдельного товара из поставки его количество и цену. Она будет связана с сущностями «Товар» и «Поставка». В сущности «Поставка» из свойств останутся лишь её дата и поставщик, осуществивший поставку, остальные свойства перейдут в новую сущность;
* Сущность «Товар в покупке». Она будет связана с сущностями «Товар» и «Покупка». В сущности «Покупка» таким образом останутся свойства «Суммарная стоимость» и «Дата»;
* Сущность «Списанный товар», связанная с сущностями «Товар в поставке» и «Списание». В сущности «Списание» таким образом останутся свойства «Суммарная стоимость» и «Дата».
* Сущность «Характеристика товара», связанная с сущностями «Товар» и «Характеристика». В ней будет содержаться значение характеристики.

Отразим всё вышесказанное на логической схеме базы данных в нотации Мартина (Рисунок 5).

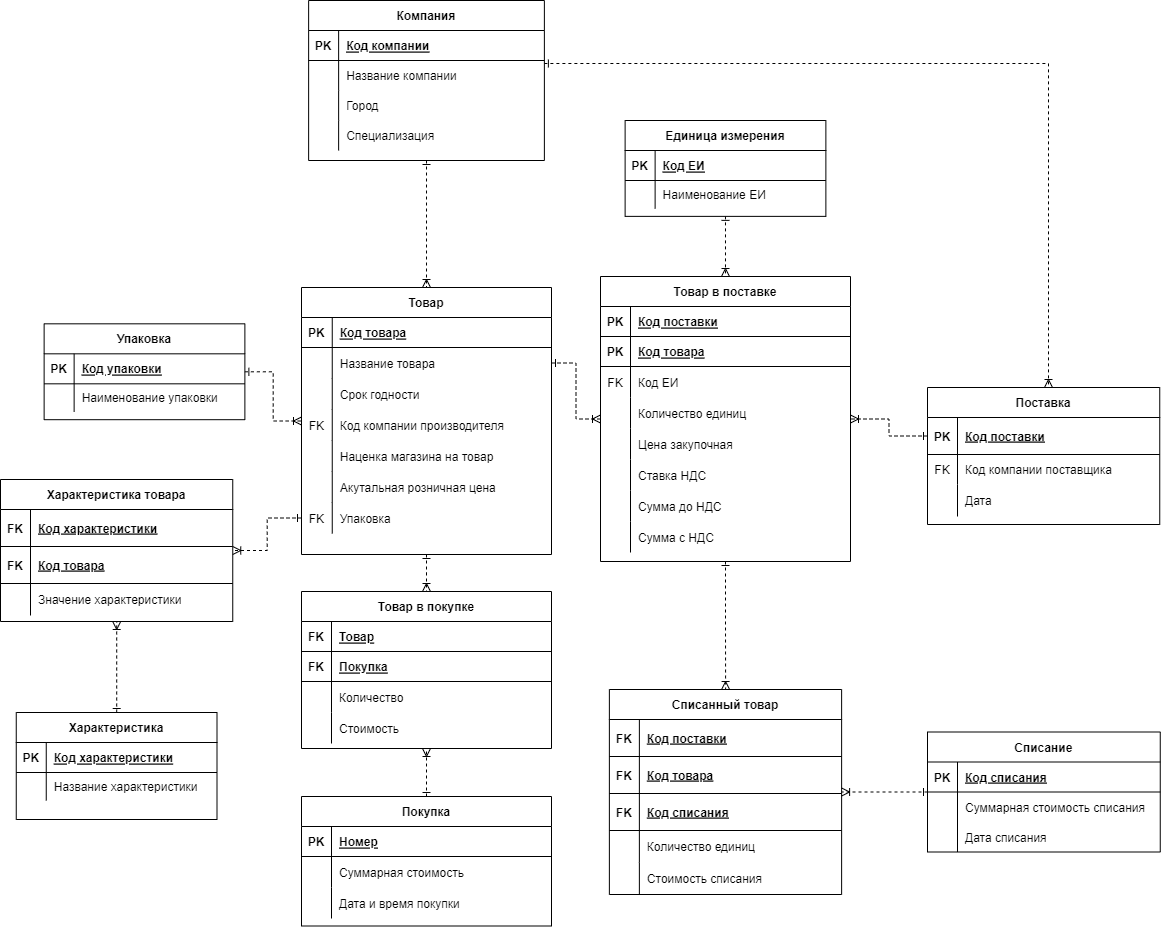


Рисунок 5 – Логическая схема базы данных в нотации Мартина

# 3. Физическое проектирование

Физическое проектирование базы данных — это создание схемы базы данных для конкретной системы управления базами данных (СУБД). Для этого необходимо выбрать СУБД и спроектировать физическую схему базы данных, с созданием таблиц и определением типов данных полей этих таблиц в нотации выбранной СУБД. Для реализации проектируемой базы данных была выбрана СУБД MySQL.

В физической схеме необходимо учесть некоторые особенности СУБД. Так, необходимо определиться, как будут выглядеть атрибуты «Специализация» сущности «Компания» и «Значение» сущности «Характеристика». Чтобы хранить информацию о том, является компания поставщиком, производителем либо всем сразу, целесообразно выделить в таблице сущности «Компания» выделить две колонки с типом значения BIT.

Для хранения информации о значениях характеристик целесообразно выделить в таблице сущности «Характеристика товара» две колонки с разными типами данных, для хранения символьного и численного представления значения, так как характеристики могут быть представлены как строка (например, цвет), так и как число (например, процент жирности).

Все численные типы данных обязаны быть беззнаковыми, так как в проектируемой базе данных не предполагается хранить отрицательные числа. Все колонки таблиц, предполагающие хранение цены, должны учитывать 2 знака после запятой.

Колонки с суммарной стоимостью целесообразно не помечать как NOT NULL, так как при заполнении базы данных может потребоваться сначала создать запись в таблице с такой колонкой, а только после этого рассчитывать суммарную стоимость.

С учётом всех вышеупомянутых замечаний построим физическую схему базы данных (Рисунок 6).

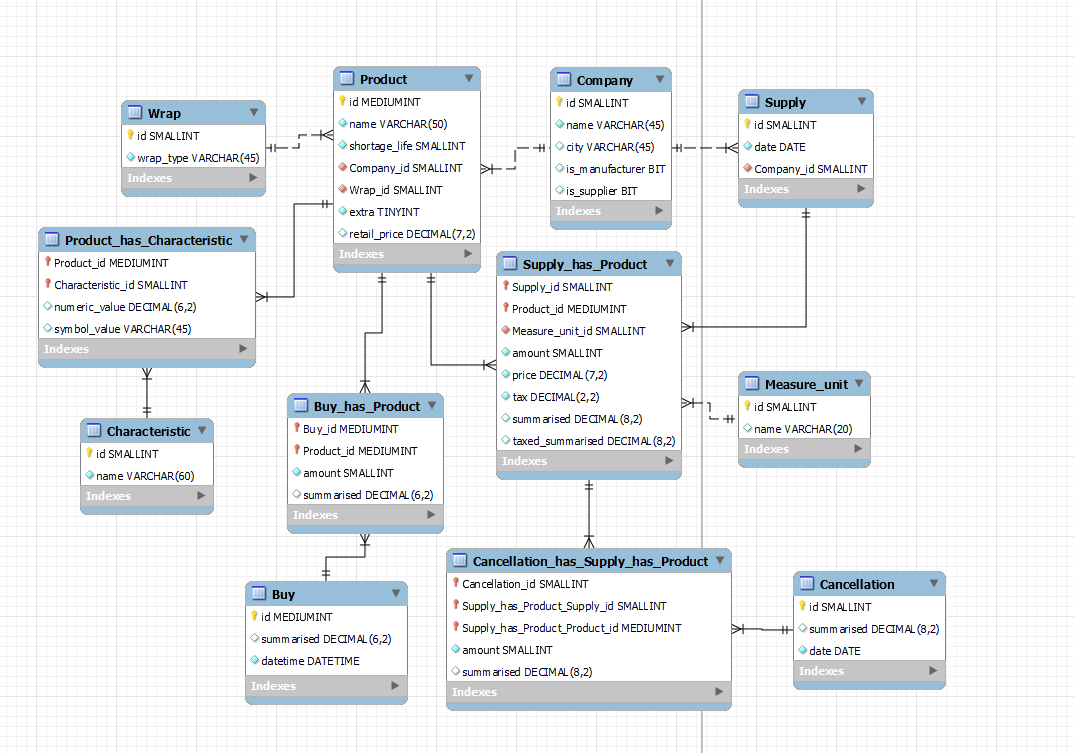


Рисунок 6 – Физическая схема базы данных MySQL

# 4. Создание базы данных

Создание базы данных в СУБД MySQL начинается с команды CREATE DATABASE <name>. Для обращения к таблицам без указания имени базы данных используется команда USE <name>. Далее командой CREATE TABLE <name> создаются таблицы с указанием имен и свойств колонок.

Для объявления первичного ключа используется конструкция PRIMARY KEY(<column\_name>). Для объявления внешнего ключа используется конструкция CONSTRAINT, реализующая ограничение на один или несколько столбцов таблицы. В ней указываются название внешнего ключа и таблица, на которую ссылается внешний ключ.

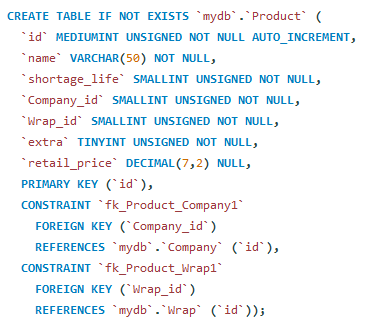


Рисунок 7 – пример создания таблицы в MySQL

# 5. Заполнение базы данных

# 6. Основные запросы к базе данных

# Заключение

В процессе разработки программы для курсовой работы мы изучили концепцию объектов и классов, а также ознакомились с методологией разработки классовых библиотек на языке C++. Мы освоили теоретические основы, которые мы применили на практике при написании программы. Таким образом, мы успешно достигли поставленной цели и выполнение задачи можно считать завершенным.

# Список использованной литературы

1. Структуры данных: связный список | habr [Электронный ресурс]. Режим доступа: [https://habr.com/ru/articles/717572](https://habr.com/ru/articles/717572/) (Дата обращения: 01.06.2023)
2. Динамическая идентификация типа данных| wikipedia [Электронный ресурс]. Режим доступа:

<https://ru.wikipedia.org/wiki/Динамическая_идентификация_типа_данных> (Дата обращения: 01.06.2023)

1. Инкапсуляция в C ++ | GeeksforGeeks [Электронный ресурс]. Режим доступа: https://www.geeksforgeeks.org/encapsulation-in-cpp/ (Дата обращения: 01.06.2023)
2. Наследование и полиморфизм | academy.yandex [Электронный ресурс]. Режим доступа: https://academy.yandex.ru/handbook/cpp/article/inheritance-and-polymorphism (Дата обращения: 01.06.2023)
3. Дружественные функции и классы | metanit.com [Электронный ресурс]. Режим доступа: https://metanit.com/cpp/tutorial/5.5.php (Дата обращения: 10.05.2023)

# Приложение 1

**Скрипт создания базы данных  
(обязательное)**

CREATE SCHEMA IF NOT EXISTS `mydb`;

USE `mydb` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Wrap` (

`id` SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

`wrap\_type` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`));

CREATE TABLE IF NOT EXISTS `mydb`.`Company` (

`id` SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`city` VARCHAR(45) NULL,

`is\_manufacturer` BIT NULL,

`is\_supplier` BIT NULL,

PRIMARY KEY (`id`));

CREATE TABLE IF NOT EXISTS `mydb`.`Measure\_unit` (

`id` SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(20) NULL,

PRIMARY KEY (`id`));

CREATE TABLE IF NOT EXISTS `mydb`.`Characteristic` (

`id` SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(60) NULL,

PRIMARY KEY (`id`));

CREATE TABLE IF NOT EXISTS `mydb`.`Cancellation` (

`id` SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

`summarised` DECIMAL(8,2) NULL,

`date` DATE NOT NULL,

PRIMARY KEY (`id`));

CREATE TABLE IF NOT EXISTS `mydb`.`Buy` (

`id` MEDIUMINT UNSIGNED NOT NULL AUTO\_INCREMENT,

`summarised` DECIMAL(6,2) NULL,

`datetime` DATETIME NOT NULL,

PRIMARY KEY (`id`));

CREATE TABLE IF NOT EXISTS `mydb`.`Supply` (

`id` SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

`date` DATE NOT NULL,

`Company\_id` SMALLINT UNSIGNED NOT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `fk\_Supply\_Company`

FOREIGN KEY (`Company\_id`)

REFERENCES `mydb`.`Company` (`id`));

CREATE TABLE IF NOT EXISTS `mydb`.`Product` (

`id` MEDIUMINT UNSIGNED NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(50) NOT NULL,

`shortage\_life` SMALLINT UNSIGNED NOT NULL,

`Company\_id` SMALLINT UNSIGNED NOT NULL,

`Wrap\_id` SMALLINT UNSIGNED NOT NULL,

`extra` TINYINT UNSIGNED NOT NULL,

`retail\_price` DECIMAL(7,2) NULL,

PRIMARY KEY (`id`),

CONSTRAINT `fk\_Product\_Company1`

FOREIGN KEY (`Company\_id`)

REFERENCES `mydb`.`Company` (`id`),

CONSTRAINT `fk\_Product\_Wrap1`

FOREIGN KEY (`Wrap\_id`)

REFERENCES `mydb`.`Wrap` (`id`));

CREATE TABLE IF NOT EXISTS `mydb`.`Product\_has\_Characteristic` (

`Characteristic\_id` SMALLINT UNSIGNED NOT NULL,

`Product\_id` MEDIUMINT UNSIGNED NOT NULL,

`numeric\_value` DECIMAL(6,2) NULL,

`symbol\_value` VARCHAR(45) NULL,

PRIMARY KEY (`Characteristic\_id`, `Product\_id`),

CONSTRAINT `fk\_Product\_has\_Characteristic1`

FOREIGN KEY (`Characteristic\_id`)

REFERENCES `mydb`.`Characteristic` (`id`),

CONSTRAINT `fk\_Product\_has\_Characteristic2`

FOREIGN KEY (`Product\_id`)

REFERENCES `mydb`.`Product` (`id`));

CREATE TABLE IF NOT EXISTS `mydb`.`Supply\_has\_Product` (

`Supply\_id` SMALLINT UNSIGNED NOT NULL,

`Product\_id` MEDIUMINT UNSIGNED NOT NULL,

`Measure\_unit\_id` SMALLINT UNSIGNED NOT NULL,

`amount` SMALLINT UNSIGNED NOT NULL,

`price` DECIMAL(7,2) NOT NULL,

`tax` DECIMAL(2,2) NOT NULL,

`summarised` DECIMAL(8,2) NULL,

`taxed\_summarised` DECIMAL(8,2) NULL,

PRIMARY KEY (`Supply\_id`, `Product\_id`),

CONSTRAINT `fk\_Supply\_has\_Product\_Supply1`

FOREIGN KEY (`Supply\_id`)

REFERENCES `mydb`.`Supply` (`id`),

CONSTRAINT `fk\_Supply\_has\_Product\_Product1`

FOREIGN KEY (`Product\_id`)

REFERENCES `mydb`.`Product` (`id`),

CONSTRAINT `fk\_Supply\_has\_Product\_Measure\_unit1`

FOREIGN KEY (`Measure\_unit\_id`)

REFERENCES `mydb`.`Measure\_unit` (`id`));

CREATE TABLE IF NOT EXISTS `mydb`.`Cancellation\_has\_Supply\_has\_Product` (

`Cancellation\_id` SMALLINT UNSIGNED NOT NULL,

`Supply\_has\_Product\_Supply\_id` SMALLINT UNSIGNED NOT NULL,

`Supply\_has\_Product\_Product\_id` MEDIUMINT UNSIGNED NOT NULL,

`amount` SMALLINT UNSIGNED NOT NULL,

`summarised` DECIMAL(8,2) NULL,

PRIMARY KEY (`Cancellation\_id`, `Supply\_has\_Product\_Supply\_id`, `Supply\_has\_Product\_Product\_id`),

CONSTRAINT `fk\_Cancellation\_has\_Supply\_has\_Product\_Cancellation1`

FOREIGN KEY (`Cancellation\_id`)

REFERENCES `mydb`.`Cancellation` (`id`),

CONSTRAINT `fk\_Cancellation\_has\_Supply\_has\_Product\_Supply\_has\_Product1`

FOREIGN KEY (`Supply\_has\_Product\_Supply\_id` , `Supply\_has\_Product\_Product\_id`)

REFERENCES `mydb`.`Supply\_has\_Product` (`Supply\_id` , `Product\_id`));

CREATE TABLE IF NOT EXISTS `mydb`.`Buy\_has\_Product` (

`Buy\_id` MEDIUMINT UNSIGNED NOT NULL,

`Product\_id` MEDIUMINT UNSIGNED NOT NULL,

`amount` SMALLINT UNSIGNED NOT NULL,

`summarised` DECIMAL(6,2) NULL,

PRIMARY KEY (`Buy\_id`, `Product\_id`),

CONSTRAINT `fk\_Buy\_has\_Product\_Buy1`

FOREIGN KEY (`Buy\_id`)

REFERENCES `mydb`.`Buy` (`id`),

CONSTRAINT `fk\_Buy\_has\_Product\_Product1`

FOREIGN KEY (`Product\_id`)

REFERENCES `mydb`.`Product` (`id`));