**Содержание**

[Введение 4](#_Toc166531108)

[1. Анализ предметной области 5](#_Toc166531109)

[1.1 Описание предметной области 5](#_Toc166531110)

[1.2 Постановка задачи 9](#_Toc166531111)

[2. Концептуальное и логическое проектирование 10](#_Toc166531112)

[2.1 Сущности предметной области 10](#_Toc166531113)

[2.2 Логическое проектирование 13](#_Toc166531114)

[3. Физическое проектирование 16](#_Toc166531115)

[4. Создание базы данных 18](#_Toc166531116)

[5. Заполнение базы данных 19](#_Toc166531117)

[6. Основные запросы к базе данных 21](#_Toc166531118)

[Заключение 23](#_Toc166531119)

[Список использованной литературы 24](#_Toc166531120)

[Приложение 1 25](#_Toc166531121)

[Приложение 2 28](#_Toc166531122)

# 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.

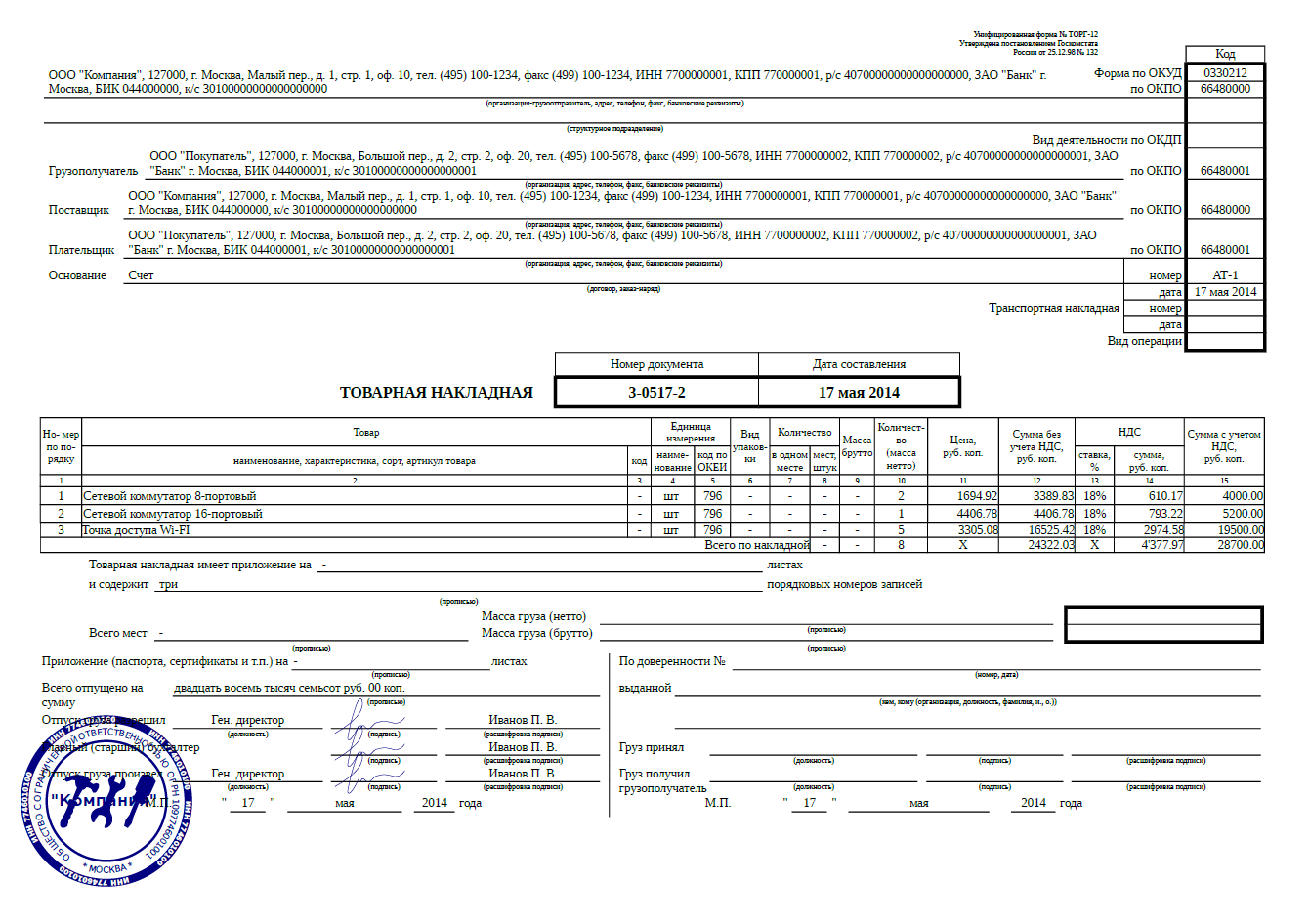


Рисунок 1 – пример заполненной накладной ТОРГ-12

В данной накладной учитываются следующие данные:

— Поставщик

— Получатель

— Принявший поставку сотрудник

— 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, фамилия кассира.

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.

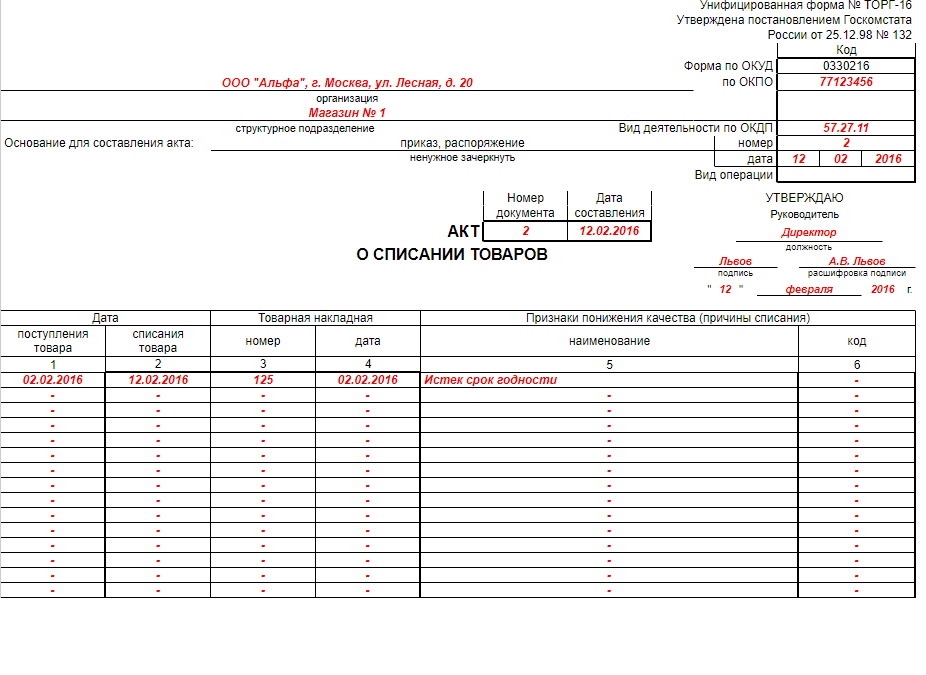


Рисунок 2 – пример заполненного акта о списании товаров ТОРГ-16. Лицевая сторона

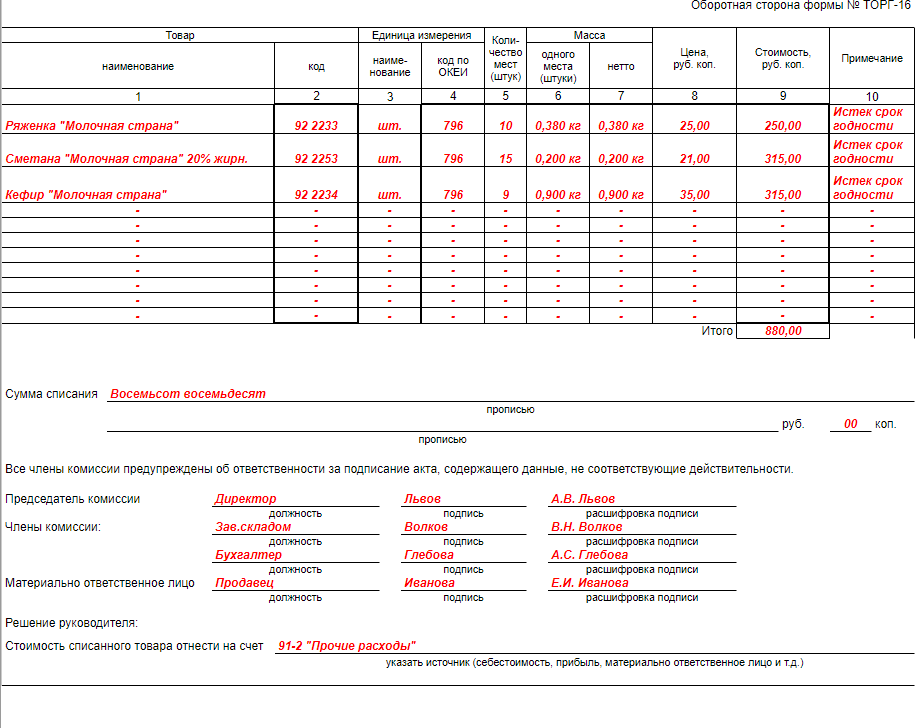


Рисунок 3 – пример заполненного акта о списании товаров ТОРГ-16. Оборотная сторона

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

— Ответственный за списание сотрудник

## 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

— Накладная

— Работник

— 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, ответственного за прием товара сотрудника, дату поставки.

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

Накладная характеризуется поступившими товарами, их количеством, закупочной стоимостью.

В сущности «Списание товара» свойствами будут являться ответственный за списание сотрудник, дата списания и количество списанного товара. Для получения данных о товарах и о датах их поставки она должна быть связана как с сущностью «Накладная».

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).

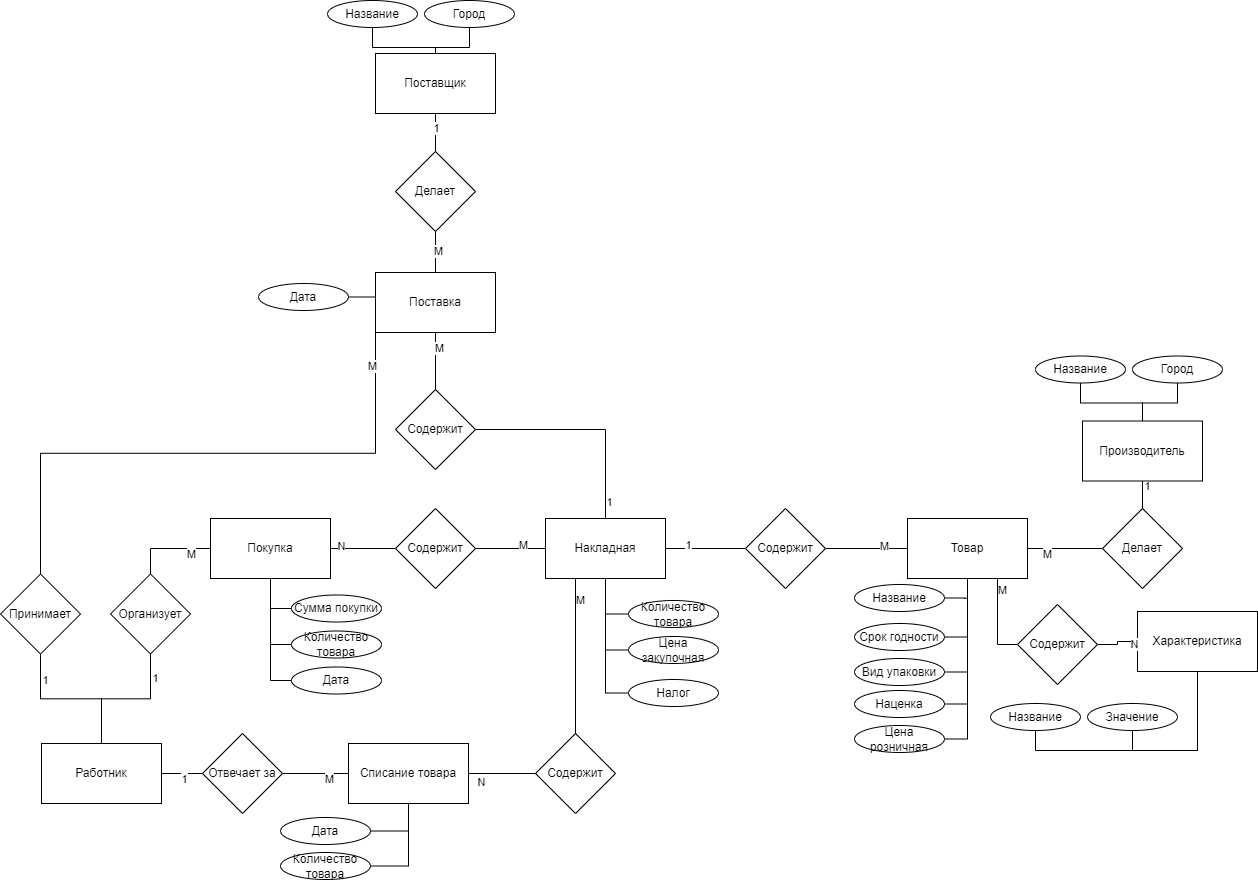


Рисунок 4 – ER-Диаграмма в нотации Чена.

## 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. Исключением станет сущность «Накладная», для нее можно выделить составной первичный ключ из внешних ключей сущностей «Товар» и «Поставка».

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 покупке». Она будет связана с сущностями «Накладная» 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).

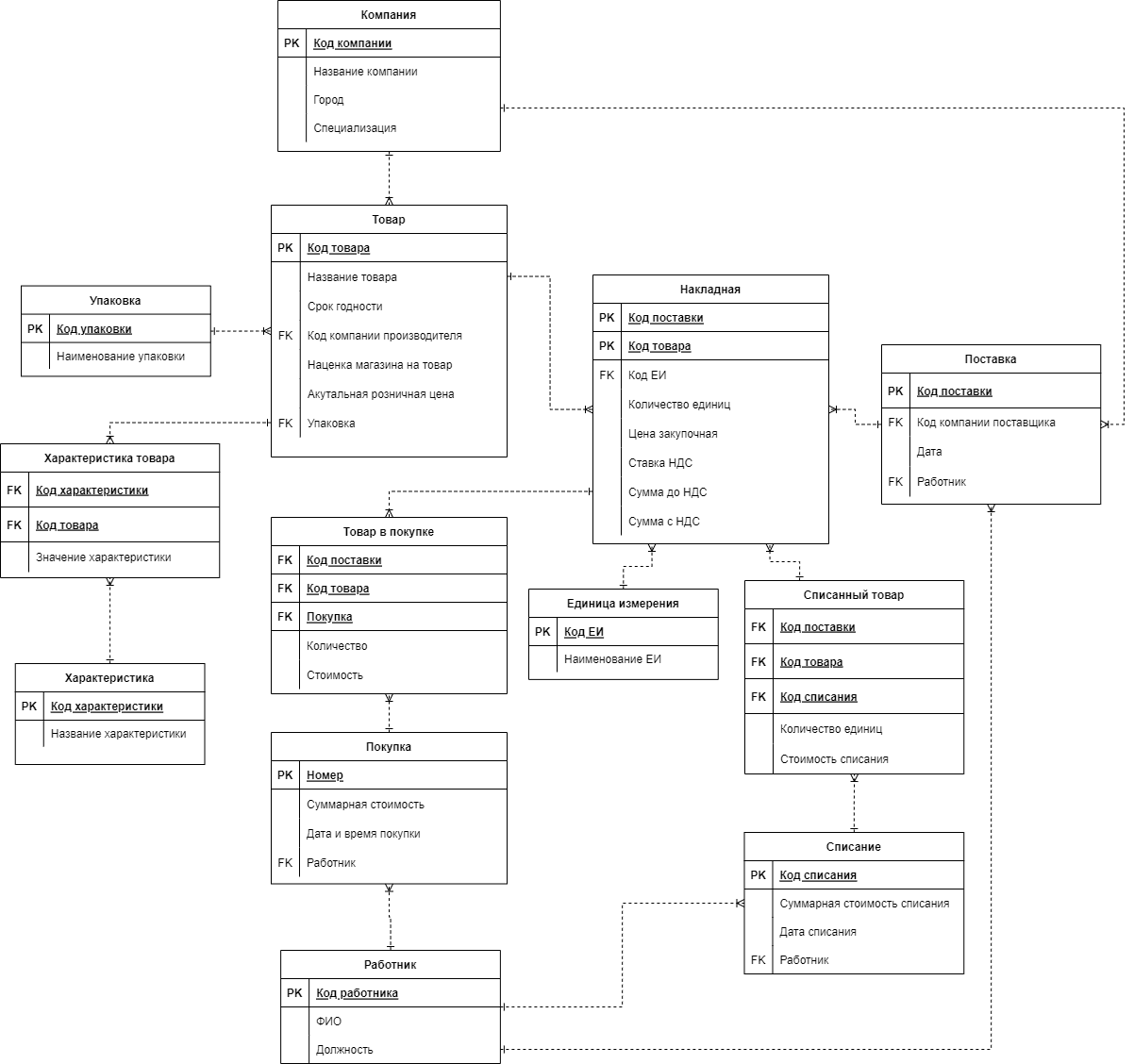


Рисунок 5 – Логическая схема базы данных в нотации Мартина

# 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.

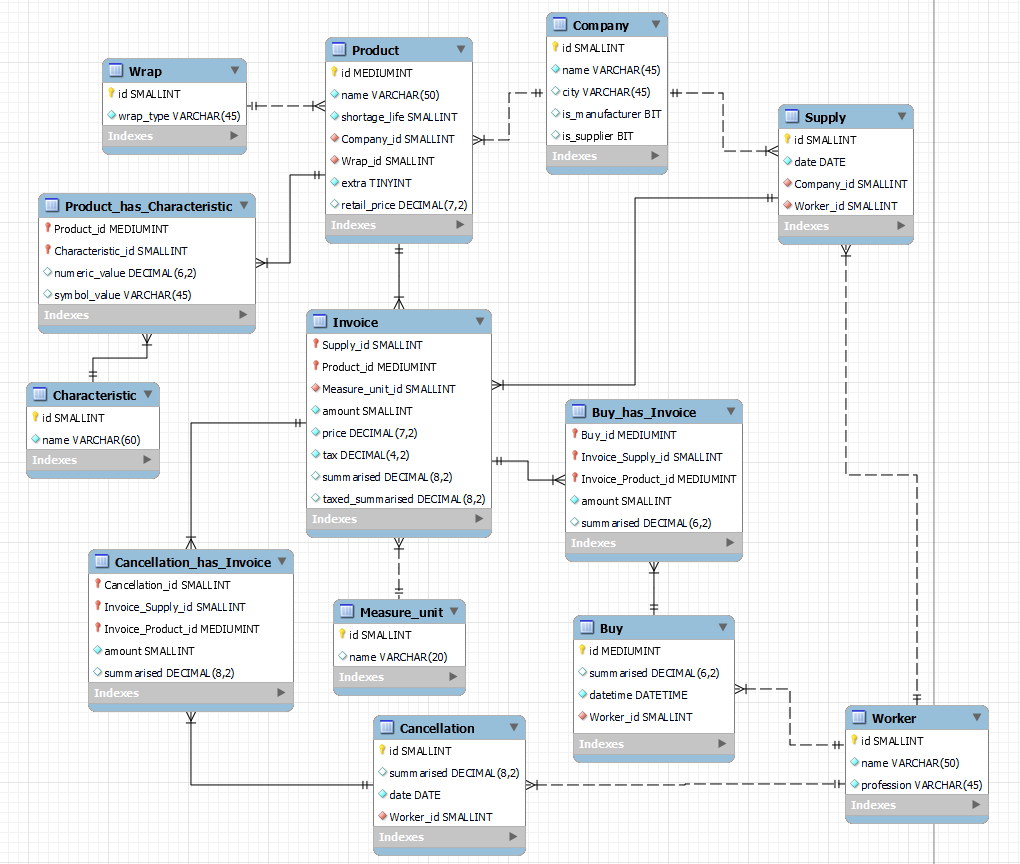
В физической схеме необходимо учесть некоторые особенности СУБД. Так, необходимо определиться, как будут выглядеть атрибуты «Специализация» сущности «Компания» и «Значение» сущности «Характеристика». Чтобы хранить информацию о том, является компания поставщиком, производителем либо всем сразу, целесообразно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

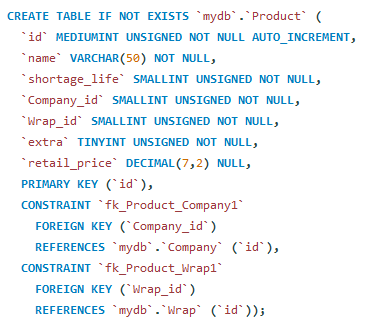


Рисунок 7 – пример создания таблицы в MySQL

Каждый столбец таблицы должен иметь уникальное в рамках этой таблицы имя, а также набор свойств, определяющих работу с этим столбцом.

В эти свойства входят:

— Тип данных

— Может ли столбец быть не заполнен

— Имеет ли столбец значение по умолчанию

# 5. Заполнение базы данных

В MySQL для заполнения таблиц используется команда INSERT. После ключевого слова INSERT INTO указываются имя таблицы и список столбцов через запятую, а затем в скобках перечисляются добавляемые значения для каждого столбца. Если для столбца определён атрибут AUTO\_INCREMENT, то его значение будет автоматически генерироваться. Также можно опускать столбцы, поддерживающие значение NULL или имеющие значение по умолчанию (Рисунок 8).

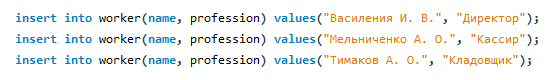


Рисунок 8 – Пример заполнения таблицы в базе данных

При заполнении базы данных можно воспользоваться вложенным запросом. Вложенный запрос в MySQL — это запрос, который выполняется внутри другого запроса. Вложенные запросы используются для извлечения данных из одной или нескольких таблиц на основе определённых условий (Рисунок 9).

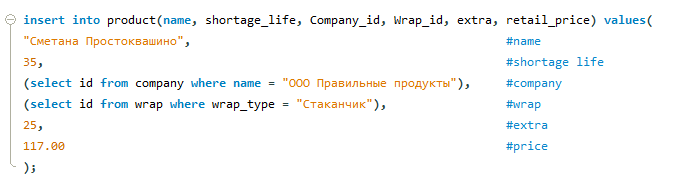


Рисунок 9 – Пример заполнения таблицы с использованием вложенного запроса

Для проектируемой базы данных будет целесообразно использование триггеров. Триггеры в MySQL — это хранимые SQL-процедуры, которые автоматически выполняются при наступлении определённых событий, таких как вставка, удаление или обновление записей в таблице. Они обеспечивают целостность данных и реализуют сложную бизнес-логику. Триггеры запускаются сервером автоматически при изменении данных в связанной таблице и выполняют соответствующие действия. Триггер создается командой CREATE TRIGGER, с указанием события для срабатывания, названия триггера и таблицы, на которой он действует.

В проектируемой базе данных в таблице «Product» есть столбец «retail\_price», отвечающий за текущую розничную цену товара. При поступлении в поставке данного товара с закупочной ценой выше, чем в прошлой поставке, розничная цена товара должна обновиться. Удобно будет осуществлять это автоматически с помощью триггера на таблице «Накладная», который будет оценивать, следует ли увеличить розничную цену, и делать это при необходимости. Создадим описанный триггер (Рисунок 10).

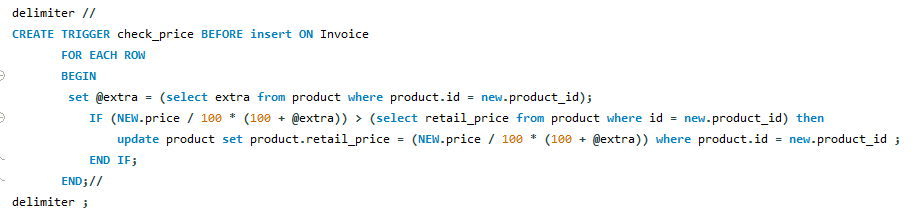


Рисунок 10 – Создание триггера для обновления розничной цены товара

Так же для удобства работы с базой данных реализуем триггеры, автоматические подсчитывающие суммарные стоимости в таблицах «Invoice», «Buy\_has\_Invoice» и «Cancellation\_has\_invoice» (Рисунок 11).

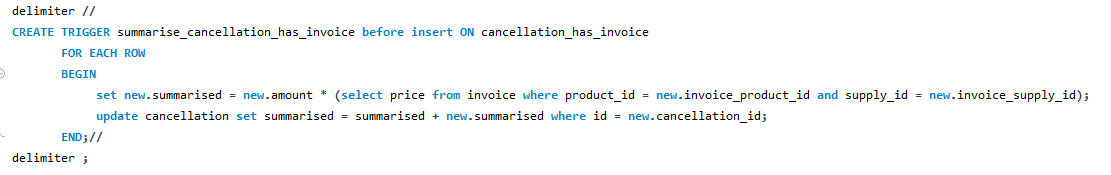


Рисунок 11 – Пример создания триггера для подсчета суммарной стоимости

Заполним базу данных используя команду INSERT.

# 6. Основные запросы к базе данных

Рассмотрим основные запросы к спроектированной и реализованной базе данных.

Выборка данных в SQL производится оператором SELECT с указанием имен таблиц, столбцов и условий выборки.

В продуктовом магазине в конце каждой смены целесообразно проверять, есть ли в магазине просроченный товар. Для этого создадим поставку, пришедшую за 6 недель до текущей даты, а товаром в этой поставке будет являться товар со сроком годности 35 дней. Создадим запрос, сравнивающий дату поставки товаров с текущей датой и сроком годности товара, и выводящий информацию о просроченных товарах (Рисунок 12).

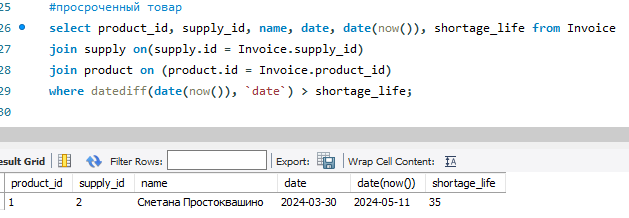


Рисунок 12 – Запрос, показывающий просроченные товары в магазине

Также, каждый раз на кассе, при пробивании товара, будет совершаться запрос к базе данных, который получает информацию о коде товара и номере его поставки. Напишем запрос, проверяющий, истек ли срок годности этого товара, и выводящий его розничную цену (Рисунок 13).

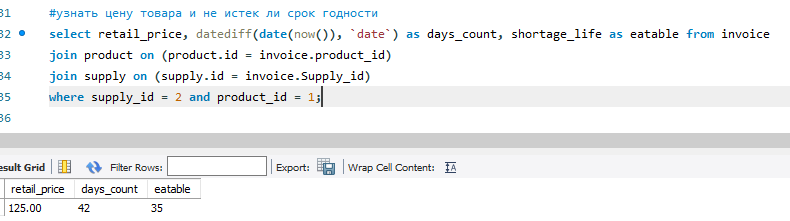


Рисунок 13 – Запрос, проверяющий количество дней с поставки товара и выводящий его розничную цену

# Заключение

В ходе выполнения курсовой работы было проведено проектирование и реализация базы данных для продуктового магазина. Были изучены теоретические основы проектирования и реализации баз данных, а также рассмотрены особенности работы продуктовых магазинов.

В процессе работы была создана база данных, включающая информацию о товарах, поставщиках, покупателях и покупках.

Для реализации базы данных были использованы современные технологии и инструменты, такие как MySQL Workbench, MySQL Server. Это позволило создать гибкую и масштабируемую систему, способную обрабатывать большие объёмы данных.

В ходе работы были проведены тестирование и отладка базы данных, что позволило выявить и устранить возможные ошибки и недочёты.

Таким образом, в результате выполнения курсовой работы была успешно спроектирована и реализована база данных для продуктового магазина, отвечающая современным требованиям и обеспечивающая эффективное функционирование предприятия.

# Список использованной литературы

1. Trigger Syntax and Examples | mysql.com [Электронный ресурс]. Режим доступа: [https://habr.com/ru/articles/717572](https://habr.com/ru/articles/717572/) (Дата обращения: 02.05.2024)

2. Adding an EER Diagram | mysql.com [Электронный ресурс]. Режим доступа: https://dev.mysql.com/doc/workbench/en/wb-creating-eer-diagram.html (Дата обращения: 03.05.2024)

3. Forward Engineering Using an SQL Script | mysql.com [Электронный ресурс]. Режим доступа: https://dev.mysql.com/doc/workbench/en/wb-forward-engineering-sql-scripts.html (Дата обращения: 03.05.2024)

4. Reverse Engineering Using a Create Script | mysql.com [Электронный ресурс]. Режим доступа: https://dev.mysql.com/doc/workbench/en/wb-reverse-engineer-create-script.html (Дата обращения: 03.05.2024)

# Приложение 1

**Скрипт создания базы данных  
(обязательное)**

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

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`.`Worker` (

`id` SMALLINT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(50) NOT NULL,

`profession` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`));

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

`id` SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

`summarised` DECIMAL(8,2) NULL DEFAULT 0,

`date` DATE NOT NULL,

`Worker\_id` SMALLINT NOT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `fk\_Cancellation\_Worker1`

FOREIGN KEY (`Worker\_id`)

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

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

`id` MEDIUMINT UNSIGNED NOT NULL AUTO\_INCREMENT,

`summarised` DECIMAL(6,2) NULL DEFAULT 0,

`datetime` DATETIME NOT NULL,

`Worker\_id` SMALLINT NOT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `fk\_Buy\_Worker1`

FOREIGN KEY (`Worker\_id`)

REFERENCES `mydb`.`Worker` (`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,

`Worker\_id` SMALLINT NOT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `fk\_Supply\_Company`

FOREIGN KEY (`Company\_id`)

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

CONSTRAINT `fk\_Supply\_Worker1`

FOREIGN KEY (`Worker\_id`)

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

**Приложение 1   
(продолжение)**

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`.`Invoice` (

`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(4,2) NOT NULL,

`summarised` DECIMAL(8,2) NULL DEFAULT 0,

`taxed\_summarised` DECIMAL(8,2) NULL DEFAULT 0,

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\_Invoice` (

`Cancellation\_id` SMALLINT UNSIGNED NOT NULL,

`Invoice\_Supply\_id` SMALLINT UNSIGNED NOT NULL,

`Invoice\_Product\_id` MEDIUMINT UNSIGNED NOT NULL,

`amount` SMALLINT UNSIGNED NOT NULL,

`summarised` DECIMAL(8,2) NULL DEFAULT 0,

PRIMARY KEY (`Cancellation\_id`, `Invoice\_Supply\_id`, `Invoice\_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 (`Invoice\_Supply\_id` , `Invoice\_Product\_id`)

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

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

`Buy\_id` MEDIUMINT UNSIGNED NOT NULL,

`Invoice\_Supply\_id` SMALLINT UNSIGNED NOT NULL,

`Invoice\_Product\_id` MEDIUMINT UNSIGNED NOT NULL,

`amount` SMALLINT UNSIGNED NOT NULL,

`summarised` DECIMAL(6,2) NULL DEFAULT 0,

PRIMARY KEY (`Buy\_id`, `Invoice\_Supply\_id`, `Invoice\_Product\_id`),

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

FOREIGN KEY (`Buy\_id`)

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

CONSTRAINT `fk\_Buy\_has\_Product\_Invoice1`

FOREIGN KEY (`Invoice\_Supply\_id` , `Invoice\_Product\_id`)

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

**Приложение 1   
(продолжение)**

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

`id` SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(60) NOT NULL,

PRIMARY KEY (`id`));

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

`Product\_id` MEDIUMINT UNSIGNED NOT NULL,

`Characteristic\_id` SMALLINT UNSIGNED NOT NULL,

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

`symbol\_value` VARCHAR(45) NULL,

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

CONSTRAINT `fk\_Product\_has\_Characteristic\_Product1`

FOREIGN KEY (`Product\_id`)

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

CONSTRAINT `fk\_Product\_has\_Characteristic\_Characteristic1`

FOREIGN KEY (`Characteristic\_id`)

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

delimiter //

CREATE TRIGGER summarise\_cancellation\_has\_invoice before insert ON cancellation\_has\_invoice

FOR EACH ROW

BEGIN

set new.summarised = new.amount \* (select price from invoice where product\_id = new.invoice\_product\_id and supply\_id = new.invoice\_supply\_id);

update cancellation set summarised = summarised + new.summarised where id = new.cancellation\_id;

END;//

delimiter ;

delimiter //

CREATE TRIGGER summarise\_buy\_has\_invoice before insert ON buy\_has\_invoice

FOR EACH ROW

BEGIN

set new.summarised = new.amount \* (select retail\_price from product where id = new.invoice\_product\_id);

update buy set summarised = summarised + new.summarised where id = new.buy\_id;

END;//

delimiter ;

delimiter //

CREATE TRIGGER summarise\_invoice before insert ON Invoice

FOR EACH ROW

BEGIN

set new.summarised = new.amount \* new.price;

set new.taxed\_summarised = new.summarised \* (100 + new.tax) / 100;

END;//

delimiter ;

delimiter //

CREATE TRIGGER check\_price BEFORE insert ON Invoice

FOR EACH ROW

BEGIN

set @extra = (select extra from product where product.id = new.product\_id);

IF (NEW.price / 100 \* (100 + @extra)) > (select retail\_price from product where id = new.product\_id) then

update product set product.retail\_price = (NEW.price / 100 \* (100 + @extra)) where product.id = new.product\_id ;

END IF;

END;//

delimiter ;

# Приложение 2

**Скрипт заполнения базы данных  
(обязательное)**

insert into worker(name, profession) values("Василения И. В.", "Директор");

insert into worker(name, profession) values("Мельниченко А. О.", "Кассир");

insert into worker(name, profession) values("Тимаков А. О.", "Кладовщик");

insert into wrap(wrap\_type) values ("Коробка");

insert into wrap(wrap\_type) values ("Пакет");

insert into wrap(wrap\_type) values ("Контейнер");

insert into wrap(wrap\_type) values ("Стаканчик");

insert into wrap(wrap\_type) values ("Бутылка");

insert into wrap(wrap\_type) values ("Банка");

insert into measure\_unit(name) values("Киллограм");

insert into measure\_unit(name) values("Грамм");

insert into measure\_unit(name) values("Штука");

insert into measure\_unit(name) values("Литр");

insert into measure\_unit(name) values("Миллилитр");

insert into company(name, city, is\_manufacturer, is\_supplier) values("ИП Харламов М.В.", "Орел", false, true);

insert into company(name, city, is\_manufacturer, is\_supplier) values("Логистический партнёр", "Орел", false, true);

insert into company(name, city, is\_manufacturer, is\_supplier) values("Опт-Торг", "Орел", false, true);

insert into company(name, city, is\_manufacturer, is\_supplier) values("Сельскохозяйственная Марёвская Корпорация", "Марево", true, false);

insert into company(name, city, is\_manufacturer, is\_supplier) values("ООО МЯСОПРОДУКТ", "Москва", true, false);

insert into company(name, city, is\_manufacturer, is\_supplier) values("ООО Правильные продукты", "Абакан", true, false);

insert into company(name, city, is\_manufacturer, is\_supplier) values("ООО Хэппи-Фиш", "Астрахань", true, true);

insert into characteristic(name) values("Вес");

insert into characteristic(name) values("Процент жирности");

insert into characteristic(name) values("Наличие заменителя молочного жира");

insert into characteristic(name) values("Вид приготовления");

insert into product(name, shortage\_life, Company\_id, Wrap\_id, extra, retail\_price) values(

"Сметана Простоквашино",

35,

(select id from company where name = "ООО Правильные продукты"),

(select id from wrap where wrap\_type = "Стаканчик"),

25,

117.00

);

insert into Product\_has\_Characteristic(product\_id, characteristic\_id, numeric\_value, symbol\_value) values(1,1,300,NULL);

insert into Product\_has\_Characteristic(product\_id, characteristic\_id, numeric\_value, symbol\_value) values(1,2,20,NULL);

insert into Product\_has\_Characteristic(product\_id, characteristic\_id, numeric\_value, symbol\_value) values(1,3,NULL,"да");

insert into product(name, shortage\_life, Company\_id, Wrap\_id, extra, retail\_price) values(

"Сметана Простоквашино",

35,

(select id from company where name = "ООО Правильные продукты"),

(select id from wrap where wrap\_type = "Стаканчик"),

25,

87.00

);

insert into Product\_has\_Characteristic(product\_id, characteristic\_id, numeric\_value, symbol\_value) values(2,1,180,NULL);

insert into Product\_has\_Characteristic(product\_id, characteristic\_id, numeric\_value, symbol\_value) values(2,2,15,NULL);

**Приложение 2   
(продолжение)**

insert into Product\_has\_Characteristic(product\_id, characteristic\_id, numeric\_value, symbol\_value) values(2,3,NULL,"да");

insert into product(name, shortage\_life, Company\_id, Wrap\_id, extra, retail\_price) values(

"Сметана Простоквашино",

35,

(select id from company where name = "ООО Правильные продукты"),

(select id from wrap where wrap\_type = "Стаканчик"),

25,

95.00

);

insert into Product\_has\_Characteristic(product\_id, characteristic\_id, numeric\_value, symbol\_value) values(3,1,180,NULL);

insert into Product\_has\_Characteristic(product\_id, characteristic\_id, numeric\_value, symbol\_value) values(3,2,15,NULL);

insert into Product\_has\_Characteristic(product\_id, characteristic\_id, numeric\_value, symbol\_value) values(3,3,NULL,"нет");

insert into supply(date, company\_id, worker\_id) values(DATE\_SUB(date(now()), INTERVAL 1 WEEK),2,3);

insert into Invoice(supply\_id, product\_id, measure\_unit\_id, amount, price, tax) values(1,1,(select id from measure\_unit where `name` = "Штука"),40,93.6,18);

insert into supply(date, company\_id, worker\_id) values(DATE\_SUB(date(now()), INTERVAL 6 WEEK),2,3);

insert into Invoice(supply\_id, product\_id, measure\_unit\_id, amount, price, tax) values(2,1,(select id from measure\_unit where `name` = "Штука"),30,100,18);

insert into buy(worker\_id, `datetime`) values(2,now());

insert into buy\_has\_invoice(buy\_id, invoice\_product\_id, invoice\_supply\_id, amount) values(1,1,2,4);

insert into cancellation(date, worker\_id) values(now(), 3);

insert into cancellation\_has\_invoice(cancellation\_id, invoice\_supply\_id, invoice\_product\_id, amount) values(1,2,1,30);