


МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем

Работа допущена к защите

 Руководитель

«19» 12 2019 г.

*М. Я. Яковлев*

**КУРСОВАЯ РАБОТА**

по дисциплине «Базы данных»

на тему: «Разработка и реализация базы данных по учёту продаж и поставок интернет-магазина компьютерных комплектующих»

Студент  Беликов П.Г.

Шифр 180818

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Группа 81ПГ

Руководитель  Рыженков Д.В.

Оценка: «»

Дата 26.12.2019

Орел 2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем

УТВЕРЖДАЮ:

 Зав. кафедрой

«17» сентября 2019 г.

**ЗАДАНИЕ**  
на курсовую работу

по дисциплине «Базы данных»

Студент Беликов П.Г.

— Шифр 180818

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Группа 81ПГ

1 Тема курсовой работы

«Разработка и реализация базы данных по учёту продаж и поставок интернет-магазина компьютерных комплектующих»

2 Срок сдачи студентом законченной работы «16» декабря 2019 г.

### 3 Исходные данные

Данные по учёту продаж и поставок интернет-магазина компьютерных комплектующих.

### 4 Содержание курсовой работы

Анализ и описание предметной области. Проектирование схемы базы данных по учёту продаж и поставок интернет-магазина компьютерных комплектующих. Реализация базы данных на языке SQL. Реализация процедур и триггеров.

### 5 Отчетный материал курсовой работы

Пояснительная записка курсовой работы.

Руководитель  Рыженков Д.В.

Задание принял к исполнению: «16» сентября 2019

Подпись студента 

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ.....	5
2 КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ БД.....	7
3 ЛОГИЧЕСКАЯ МОДЕЛЬ БД.....	9
3.1 Разработка логической модели.....	9
3.2 Выявление перечня ограничений целостности и выбор способа реализации его контроля.....	10
3.3 Первая нормальная форма (NF1).....	11
3.4 Вторая нормальная форма (NF2).....	12
3.5 Третья нормальная форма (NF3).....	13
4 ФИЗИЧЕСКАЯ МОДЕЛЬ БД.....	15
5 РЕАЛИЗАЦИЯ БД.....	18
6 РАЗРАБОТКА ХРАНИМЫХ ПРОЦЕДУР И ТРИГГЕРОВ.....	20
7 РЕАЛИЗАЦИЯ ЗАПРОСОВ.....	22
ЗАКЛЮЧЕНИЕ .....	23
Список использованных источников.....	24
Приложение А (Схемы БД).....	25
Приложение Б (Скрипты создания таблиц).....	29
Приложение В (Скрипты заполнения таблиц) .....	38
Приложение Г (Скрипты триггеров и процедур).....	49
Приложение Д (Скрипты запросов).....	56

## **ВВЕДЕНИЕ**

Целью курсовой работы является создание базы данных для интернет-магазина компьютерных комплектующих. База данных должна обеспечивать быстрый доступ к данным, необходимым пользователю.

Задачами курсовой работы, которые необходимо выполнить для достижения поставленной цели, являются:

1. Изучение принципов работы интернет-магазина.
2. Составление описания предметной области.
3. Разработка концептуальной схемы базы данных (нотация Чена).
4. Разработка логической и физической схем базы данных (IDEF1X).
5. Реализация разработанной схемы в определенной СУБД.
6. Создание запросов, процедур и триггеров к реализованной базе данных.

## 1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Предполагаемая база данных должна обеспечивать работу интернет-магазина по учету продаж и поставок компьютерных комплектующих.

В интернет-магазине пользователь может заказать товары удаленно. Каждый клиент является пользователем. Чтобы воспользоваться интернет-магазином, пользователь должен зарегистрироваться, введя следующие данные: ФИО, номер телефона, адрес проживания и адрес электронной почты. Для просмотра каталога товаров регистрация не требуется.

На сайте представлены различные товары, пользователь может просматривать их описание и характеристики. Товары в интернет-магазине разделены на категории (виды). Выбрав один конкретный товар, пользователь может посмотреть информацию о нем: полное название, вид, описание, код, производителя, характеристики и наличие (количество) на складе.

Чтобы получить продукцию, представленную на сайте, пользователь должен оформить заказ. Клиент выбирает товары, формируя содержимое будущего заказа, определяет вид доставки и оплаты. Список выбранных товаров хранится в оперативной памяти устройства клиента и отправляется на сервер только после создания заказа. Следовательно, до момента окончательного оформления заказа клиент может изменять сколь угодно много список желаемых товаров. Пользователю доступны следующие способы доставки: самовывоз, доставка на указанный домашний адрес курьером. Ввиду того, что предполагаемый интернет-магазин будет функционировать в пределах одного города, цена каждого вида доставки фиксированная и указывается при оформлении заказа. Оплата заказа может быть произведена следующими способами: банковской картой при оплате на сайте, наличными при получении заказа. Один заказ может содержать различные товары. Каждый товар может быть заказан в размере, не превышающем его наличие на складе. Каждый клиент может сделать несколько заказов, но каждый заказ может быть сделан только одним

клиентом. На сайте пользователь может просматривать информацию о сделанном заказе. В ней содержится следующее: итоговая цена, дата совершения заказа, список товаров, ожидаемая дата доставки, способ доставки, состояние заказа, способ оплаты, номер клиента, номер сотрудника, обработавшего заказ.

Товары поступают на склад от поставщиков. В одной поставке может находиться несколько видов товара. Формируется реестр поставок. В нем хранится следующая информация: код поставки, дата, цена закупки, наименование компании поставщика, список товаров и их количество. О каждом поставщике известно следующее: название компании, адрес веб-сайта, адрес почты. В цену закупки не входит оплата затрат на ее поставку. Один поставщик может предоставлять разные виды товаров. Стоимость товара в магазине получается из цены его закупки, увеличенной на фиксированное значение. Это означает, что могут существовать два одинаковых товара с разной ценой. Увеличивающая цену константа зависит от вида товара.

Сотрудники обрабатывают заказы. Один сотрудник может оформить несколько заказов, но один заказ не может быть оформлен более чем одним сотрудником. О каждом сотруднике известно следующее: ФИО, номер телефона.

## 2 КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ БД

Концептуальная модель – это отражение предметной области, для которой разрабатывается база данных. Она создается для дальнейшего проектирования базы данных и перевод ее, например, в реляционную базу данных. На концептуальной модели в визуально удобном виде прописываются связи между объектами данных и их характеристиками.

Для единообразия проектирования концептуальных моделей введены следующие понятия:

- Сущность – фактическая вещь или объект.
- Атрибут – характеристика объекта.
- Отношение – показывает связь между объектами.

Выделим объекты предметной области: «Клиент», «Заказ», «Сотрудник», «Товар», «Поставка», «Поставщик».

Сущность «Клиент» характеризуется следующими атрибутами:

- ФИО.
- Домашний адрес.
- Электронная почта.
- Номер телефона.

Сущность «Заказ» характеризуется следующими атрибутами:

- Дата.
- Итоговая цена.
- Состояние.
- Способ оплаты.
- Способ доставки.

Сущность «Сотрудник» характеризуется следующими атрибутами:

- ФИО.
- Номер телефона.



Сущность «Товар» характеризуется следующими атрибутами:

- Название.
- ID товара.
- Категория.
- Описание.
- Характеристики.
- Цена за единицу.
- Количество на складе.
- Производитель.

Сущность «Поставка» характеризуется следующими атрибутами:

- ID поставки.
- Дата.
- Итоговая цена.

Сущность «Поставщик» характеризуется следующими атрибутами:

- Название компании.
- Веб сайт.
- Электронная почта.
- Реквизиты.

Каждый клиент может сделать множество заказов, но каждый заказ может быть сделан только одним клиентом. Каждый Сотрудник может оформить множество заказов, но каждый заказ может быть оформлен только одним сотрудником. В заказе может быть множество товаров и один товар может находиться в нескольких заказах. В каждой поставке может быть множество товаров, и один товар может находиться в нескольких поставках.

На основании всего вышесказанного была построена концептуальная модель базы данных, представленная на рисунке 1 Приложения А.

### 3 ЛОГИЧЕСКАЯ МОДЕЛЬ БД

#### 3.1 Разработка логической модели

Логическая модель находится на более низком уровне предметной области, нежели концептуальная. Она представляет собой модель базы данных, которая не привязана к конкретной СУБД. В ней выделяют основные объекты БД и определяют связи между этими объектами.

Логическая модель ограничивается решениями, принятыми при разработке концептуальной модели. Таким образом устанавливаются пределы, в рамках которых мы можем принимать решения при создании логической модели.

Приняв за основу концептуальную схему, была построена логическая модель. В нее были включены сущности: «Клиент», «Сотрудник», «Заказ», «Товар», «Поставка», «Товар - поставка» и «Товар заказ».

Между сущностями «Клиент» и «Заказ» была определена связь «один ко многим» - один клиент может сделать множество заказов, но каждый заказ может быть сделан только одним клиентом.

Между сущностями «Сотрудник» и «Заказ» была определена связь «один ко многим» - один сотрудник может оформить множество заказов, но каждый заказ может быть оформлен только одним сотрудником.

Между сущностями «Товар» и «Поставка» была определена связь «многие ко многим» - один товар может находиться в разных поставках, а в одной поставке могут находиться множество товаров. Но ввиду особенности бизнес политики магазина, данная связь была заменена двумя связями «один ко многим» и промежуточной сущностью «Товар - поставка».

Следуя политике продажи товара в магазине, при создании сущностей и связей следует учитывать, что каждый товар продается по конкретной, зависящей от его вида наценке. При этом наценка к делается к цене товара из

конкретной поставки. Следующая связь, как и предыдущая, была построена следуя этому условию.

Между сущностями «Заказ» и «Товар - поставка» была определена связь «многие ко многим» - товар из одной поставки может находиться в разных заказах, а в каждом заказе могут находиться товары из разных поставок. Эта связь была заменена двумя связями «один ко многим» и промежуточной сущностью «Товар - заказ».

### **3.2 Выявление перечня ограничений целостности и выбор способа реализации его контроля**

Для корректной работы базы данных необходимо определить ограничения, следуя которым она будет функционировать, а также определить способ реализации их контроля.

В каждом отношении атрибут вида «ID», являющийся первичным ключом (РК), заполняется автоматически. Это правило не применимо к атрибутам вида «ID», которые являются наследуемыми от других отношений (FK).

Атрибут «Цена заказа» сущности «Заказ» может быть только числом больше нуля. Это же ограничение применяется к атрибутам «Количество товара» и «Цена за единицу» сущности «Товар - заказ», атрибутам «Количество товара» и «Цена за единицу» сущности «Товар - поставка», атрибуту «Итоговая цена поставки» сущности «Поставка».

Атрибут «Количество на складе» сущности «Товар» может быть только числом, которое больше нуля или самим нулем.

Атрибут «Дата заказа» сущности «Заказ» не может превышать текущую дату. Это правило применимо к атрибуту «Дата поставки» сущности «Поставка».

Товар не может быть заказан в числе, превышающем его наличие на складе. Исходя из этого, значение атрибута «Количество товара» сущности «Товар - заказ» не может превышать значение атрибута «Количество на складе» сущности «Товар».

Все вышеперечисленные ограничения будут учтены и реализованы с помощью хранимых процедур и триггеров.

Основываясь на концептуальной модели и информации, приведенной выше, была построена логическая модель базы данных.

Не нормализованная логическая модель представлена на рисунке 2 Приложения А.

Перед тем, как перейти к следующему уровню проектирования базы данных, нам необходимо нормализовать логическую модель.

### **3.3 Первая нормальная форма (1NF)**

Чтобы привести логическую модель к первой нормальной форме, мы должны сделать каждый её атрибут атомарным.

Приведем таблицу «Клиент» к 1NF: разделим атрибуты «ФИО» и «Адрес» на составляющие. Полученная таблица представлена на рисунке 3.1.

Приведем таблицу «Сотрудник» к 1NF: разделим атрибут «ФИО» на составляющие. Полученная таблица представлена на рисунке 3.2.

В обоих случаях атрибут «ФИО» делится на три новых: «Фамилия», «Имя», «Отчество». Так же «Адрес» делится на четыре новых атрибута: «Город», «Улица», «Дом» и «Квартира».

Все оставшиеся таблицы удовлетворяют требованиям первой нормальной формы и не требуют изменения.

### 3.4 Вторая нормальная форма (2NF)

Чтобы привести отношение ко второй нормальной форме, необходимо сначала привести её к первой нормальной форме. Затем нужно сделать так, чтобы любой её атрибут, не входящий в состав первичного ключа, функционально полно зависел от первичного ключа. (т.е. от всего ключа, а не от его части).

Клиент
ID клиента
Фамилия
Имя
Отчество
Город
Улица
Дом
Квартира
Электронная почта
Номер телефона

Рисунок 3.1 – Таблица «Клиент» в 1NF

Сотрудник
ID сотрудника
Фамилия
Имя
Отчество
Номер телефона

Рисунок 3.2 – Таблица «Сотрудник» в 1NF

Приведем отношение «Заказ» к NF2, создадим отношения «Состояние», «Способ оплаты», «Способ доставки». Теперь в приведенном отношении будем хранить внешние ключи на соответствующие экземпляры новых отношений. Все новые отношения соединяются с приведенным связью «один ко многим».

Приведем отношение «Товар» к NF2, создадим отношения «Производитель», «Категория» и «Характеристика». В приведенном отношении будем хранить внешние ключи на соответствующие экземпляры новых отношений. Между отношениями «Товар» и «Характеристика» была образована связь «многие ко многим». Исключим ее путем замены на две связи «один ко многим» и новую промежуточное отношение «Товар-характеристика». Оставшиеся новые отношения соединяются с приведенным связью «один ко многим».

Приведем отношение «Поставка» к NF2, создадим отношение «Поставщик». В приведенном отношении будем хранить внешний ключ на соответствующий экземпляр нового отношения. Новое отношение соединяется с приведенным связью «один ко многим».

Все остальные отношения удовлетворяют условия второй нормально формы.

### **3.5 Третья нормальная форма (3NF)**

Отношение находится в 3NF, когда находится во 2NF и каждый не ключевой атрибут не транзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы. Уничтожение транзитивных зависимостей обеспечивает целостность данных в БД. [1]

В получившейся базе данных нет транзитивных зависимостей. Таким образом, логическая модель не требует изменений.

Нормализованная логическая схема данных представлена на рисунке 3  
Приложения А.

## 4 ФИЗИЧЕСКАЯ МОДЕЛЬ БД

Физическая модель базы данных находится на более низком уровне, чем логическая модель. Она содержит все детали, необходимые конкретной СУБД для создания базы: наименования таблиц и столбцов, типы данных, определения первичных и внешних ключей и т.п. Физическая модель строится на основе логической с учетом ограничений, накладываемых возможностями выбранной СУБД.

В данной курсовой работе в качестве СУБД была выбрана MySQL версии 8.0.15.

Построение физической модели производится на основе логической модели. Это значит, что все принятые решения при её построении будут влиять на физическую модель данных. Поэтому очень важно принимать правильные решения с самого начала.

На основании логической модели и выбранной СУБД определим следующие таблицы:

- Отношение «Клиент» - таблица «Client».
- Отношение «Сотрудник» - таблица «Staff».
- Отношение «Заказ» - таблица «Order».
- Отношение «Состояние» - таблица «Status».
- Отношение «Оплата» - таблица «Pay».
- Отношение «Способ доставки» - таблица «Shipping».
- Отношение «Поставщик» - таблица «Supplier».
- Отношение «Товар - заказ» - таблица «Order\_List».
- Отношение «Поставка» - таблица «Supply».
- Отношение «Товар - поставка» - таблица «Supply-Product».
- Отношение «Товар» - таблица «Product».
- Отношение «Производитель» - таблица «Manufacturer».
- Отношение «Категория» - таблица «Category».



- Отношение «Характеристика» - таблица «Characteristic».
- Отношение «Товар - характеристика» - таблица «Product\_Characteristics».

В базе данных применяются стандартные типы данных, среди них:

1. «INT» - обыкновенный целочисленный тип. Используется для «ID» ключей (первичных и наследуемых), а также обозначения цены или количества.

2. «VARCHAR» - строковый тип переменной длины. Используется в атрибутах, связанных с хранением информации о клиенте, сотруднике. Этот тип также применяется в атрибутах, хранящих название чего – либо.

3. «DATE» - хранит дату в формате ГГГГ-ММ-ДД. Используется для атрибутов, связанных с хранением даты.

4. «DATETIME» - хранит дату и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС. Используется для атрибутов, где нужно зафиксировать не только дату, но и время.

Почти все атрибуты получили характеристику NOT NULL, что делает их обязательными для заполнения. Исключениями стали логически обоснованные атрибуты, приведенные ниже.

Атрибут, содержащий отчество человека - «Patronymic». Это обосновывается тем, что существуют люди без отчества.

Атрибут «Apartment» - люди, живущие в частных домах, не смогут указать номер своей квартиры.

Атрибут «Measure\_Unit» - для некоторых характеристик продукта единицы измерения не указываются.

Атрибут «Description» - не всем продуктам нужно описание, поэтому решено было сделать заполнение данного атрибута не обязательным.

Данная модель базы данных требует исключение связи «многие ко многим». Для этого необходимо создать промежуточную таблицу и использовать две связи «один ко многим» от старых таблиц к новой. В полученной модели отсутствуют связи «многие ко многим».

Ссылочная целостность – это ограничение базы данных, гарантирующее, что ссылки между данными являются действительно правомерными и неповрежденными. [6]

Ссылочная целостность реализуемой базы данных обеспечивается с помощью NO ACTION. Эта команда запрещает удаление записи, если имеются зависимые записи с соответствующим значением внешнего ключа.

Получившаяся физическая модель представлена на рисунке 4 приложения А.

## 5 РЕАЛИЗАЦИЯ БД

SQL (Structured Query Language – Структурированный язык запросов) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Является информационно – логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. SQL считается языком программирования, в общем случае (без ряда современных расширений) не являющимся тьюринг – полным, но вместе с тем стандарт языка предусматривает возможность его процедурных расширений. [3]

Чтобы приступить к реализации БД, необходимо создать её. В моем случае скрипт создания выглядит так: `CREATE DATABASE IF NOT EXISTS `MyStore` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;`

Для того, чтобы была возможность хранить в БД символы и на латинице, и на кириллице, необходимо установить кодировку `utf8mb4`. Соответственно способ, с помощью которого следует упорядочивать и сравнивать данные в БД будет `utf8mb4_unicode_ci`;

Теперь нам необходимо создать все таблицы, которые определены у нас в физической модели. Ниже приведен скрипт создания таблицы «Supply», остальные скрипты создания таблиц приведены в Приложении Б.

```
CREATE TABLE IF NOT EXISTS `MyStore`.`Supply` (  
  `ID_Supply` INT(5) NOT NULL AUTO_INCREMENT,  
  `Date_supply` DATE NOT NULL,  
  `Supply_Price` INT(8) NOT NULL,  
  `Supplier_ID_Supplier` INT(4) NOT NULL,  
  PRIMARY KEY (`ID_Supply`, `Supplier_ID_Supplier`),
```

```

INDEX `fk_Supply_Supplier1_idx` (`Supplier_ID_Supplier` ASC)
VISIBLE,
FOREIGN KEY (`Supplier_ID_Supplier`) REFERENCES
`MyStore`.`Supplier` (`ID_Supplier`)
ON UPDATE NO ACTION
ON DELETE NO ACTION)
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Оператор PRIMARY KEY определяет первичные ключи таблицы. С помощью FOREIGN KEY .. REFERENCES мы определяем внешние ключи таблицы и то, откуда они наследуются. Для каждого внешнего ключа командой INDEX создается индекс. Он служит для повышения производительности, так как ускоряет процесс запроса, предоставляя более быстрый доступ к строкам данных в таблице, подобно тому, как указатель в книге помогает найти нужную нам информацию.

Для работы с БД нам необходимо заполнить таблицы данными. Для добавления информации в таблицу используется оператор INSERT INTO.

Ниже приведен скрипт заполнения таблицы «Supply», остальные скрипты заполнения таблиц приведены в приложении В.

```

INSERT INTO `MyStore`.`Supply` (`Date_supply`, `Supply_price`,
`Supplier_ID_Supplier`) VALUES
('2019-05-12', 0, 1),
('2019-05-15', 0, 3),
('2019-07-24', 0, 4),
('2019-09-07', 0, 2),
('2019-09-30', 0, 4),
('2019-11-11', 0, 3);

```

## 6 РАЗРАБОТКА ХРАНИМЫХ ПРОЦЕДУР И ТРИГГЕРОВ

Хранимая процедура представляет собой набор инструкций, которые выполняются как единое целое. Они позволяют упростить комплексные операции и вынести их в единый объект. Это обеспечивает лучшую производительность: хранимые процедуры выполняются быстрее, чем обычные SQL – инструкции. Все потому что код процедур компилируется один раз при первом её запуске, а затем сохраняется в скомпилированной форме. [4]

Триггеры представляют собой специальный тип хранимой процедуры, которая вызывается автоматически при выполнении определенного действия над таблицей, в частности, при добавлении, изменении или удалении данных, то есть при выполнении команд INSERT, UPDATE, DELETE. [4]

Ниже представлен скрипт триггера на увеличение количества товара на складе при его поставке:

```
DELIMITER //
CREATE TRIGGER `prod_incr`
AFTER INSERT ON `Supply-Product`
FOR EACH ROW
BEGIN
UPDATE `Product`
SET `Product`.`In_Store`=`Product`.`In_Store` +
    NEW.`Supply_prod_amount`
WHERE `Product`.`ID_Product` = NEW.`Product_ID_Product` ;
END//
DELIMITER ;
```

Команда DELIMITER используется для замены стандартного разделителя “ ; “ на другой символ. Это нужно делать при создании новой процедуры или триггера. После написания триггера или процедуры необходимо вернуть стандартный разделитель.

Скрипты остальных триггеров представлены в Приложении Г.

Также я добавил несколько процедур. Ниже представлена процедура для создания нового заказа. Она устанавливает атрибут `Date\_order` в значение, возвращаемое функцией NOW(). Скрипт процедуры приведен ниже.

```
DELIMITER //

CREATE PROCEDURE `Create_Order` (c_id INT(5), s_id INT(5), p_id
INT(2), sh_id INT(2), st_ID INT(3))
BEGIN
    INSERT INTO `Order` (`Client_ID_Client`, `Staff_ID_Staff`,
`Date_order`, `Final_price`, `Pay_Pay_ID`, `Shipping_Shipping_ID`, `Status_ID` )
VALUES (c_id, s_id, NOW(), 0, p_id, sh_id, st_ID);
END //

DELIMITER ;
```

Скрипты остальных процедур представлены в Приложении Г.

## 7 РЕАЛИЗАЦИЯ ЗАПРОСОВ

Запрос представляет собой набор условий, в зависимости от которых база данных выберет и выдаст пользователю определенную информацию.

Обязательно в каждом запросе должны быть только две составляющие: SELECT и FROM. Помимо обязательных операторов в запросе может использоваться целый ряд других команд: WHERE, ORDER BY, GROUP BY и т.д. Для выбора данных из нескольких таблиц можно использовать JOIN. Для создания дополнительных условий выбора используем WHERE.

В результате выполнения запроса формируется новое отношение и выводится пользователю.

Ниже представлен запрос на получение общей выручки за каждый месяц 2019 года.

```
SELECT      MONTHNAME(`Order`.`Date_order`)      AS      `Month`,
YEAR(`Order`.`Date_order`) AS `Year`, SUM(`Order_List`.`Product_amount` *
`Order_List`.`New_product_price`) AS `Total`
FROM `Order`, `Order_List`
WHERE (`Order`.`ID_Order` = `Order_List`.`Order_ID_Order`) AND
(YEAR(`Order`.`Date_order`) = 2019)
GROUP BY MONTHNAME(`Order`.`Date_order`);
```

Результат выполнения запроса представлен на рисунке 7.1.

	Month	YEAR	Total
►	October	2019	19800
	November	2019	191050
	December	2019	155550

Рисунок 7.1 – Выполнение запроса на получение выручки по месяцам

Скрипты остальных запросов представлены в Приложении Д.

## **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы мы определили и описали предметную область, создали концептуальную, логическую и физическую модели, реализовали базу данных на языке запросов SQL, создали запросы, хранимые процедуры и триггеры. Таким образом все поставленные цели были достигнуты.



### Список использованных источников

1. Хабр. Нормализация отношений [Электронный ресурс]. URL: <https://habr.com/ru/post/254773/>
2. Helpiks. Физическая модель БД [Электронный ресурс]. URL: <https://helpiks.org/2-92209.html>
3. Википедия. SQL. [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/SQL>
4. Metanit. SQL [Электронный ресурс]. URL: <https://metanit.com/sql/>
5. Документация MySQL [Электронный ресурс]. URL: <https://dev.mysql.com/doc/>
6. CIT Forum. Ссылочная целостность является важной для баз данных [Электронный ресурс]. URL: <http://citforum.ru/database/articles/blaha1/>

**Приложение А**  
**(обязательное)**  
**Схемы БД**

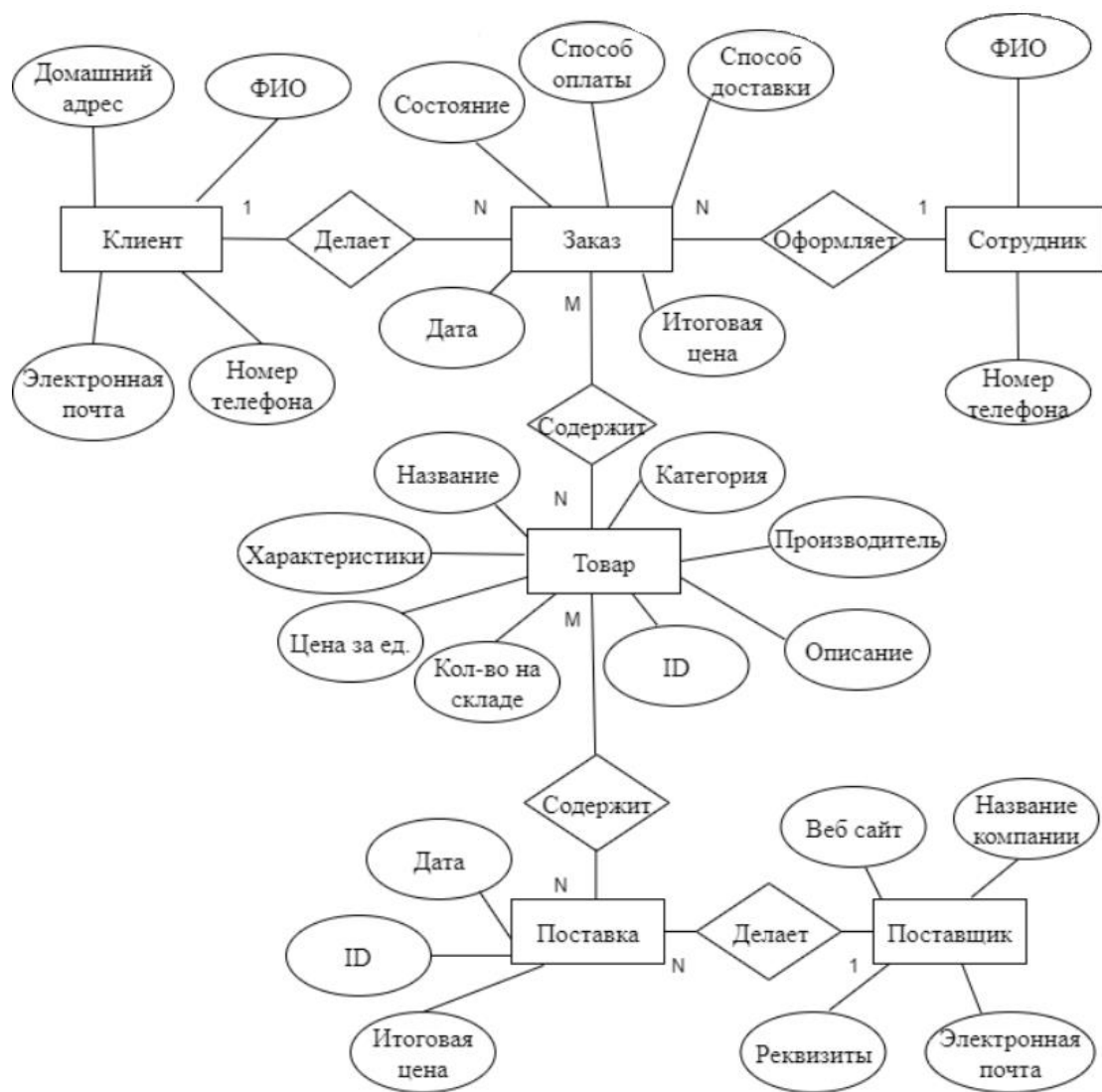


Рисунок 1 – Концептуальная модель.

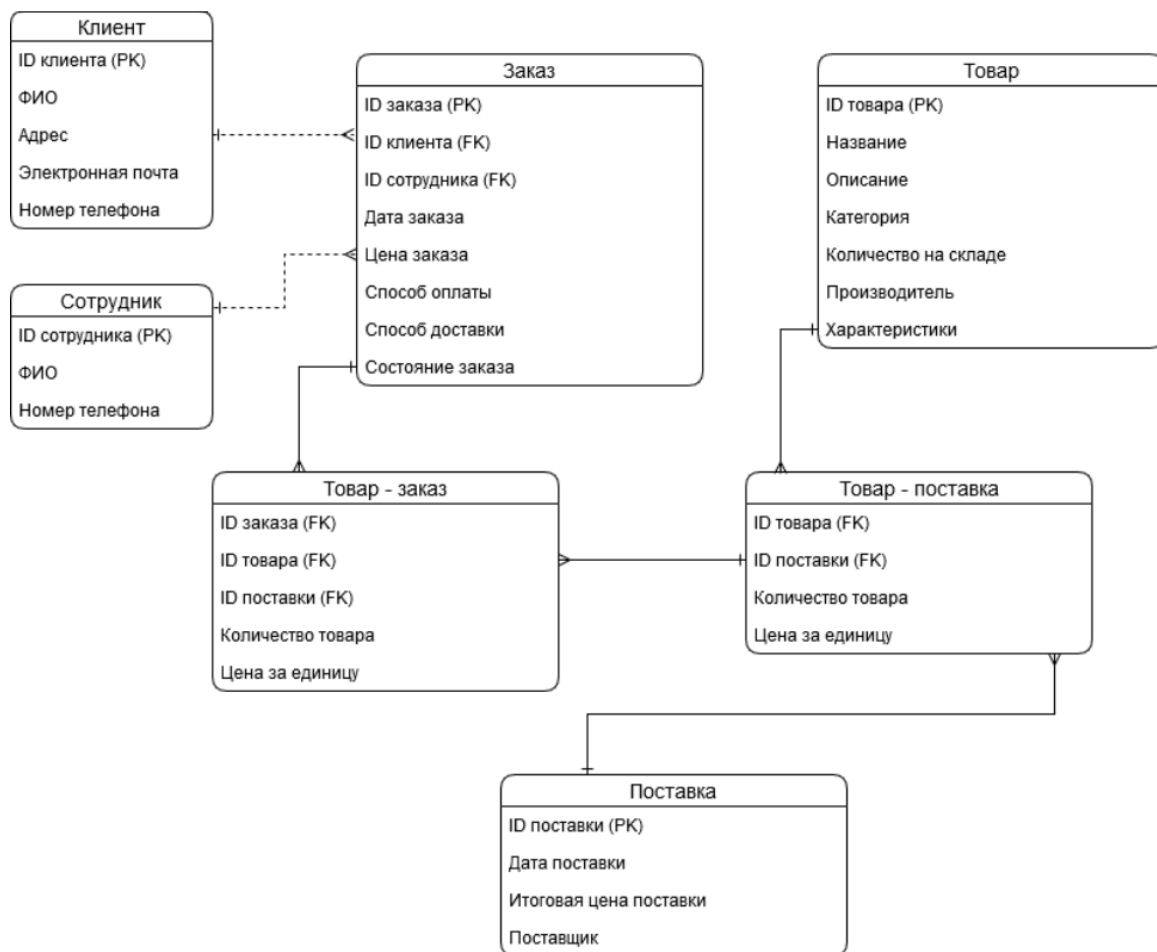


Рисунок 2 – Не нормализованная логическая модель.

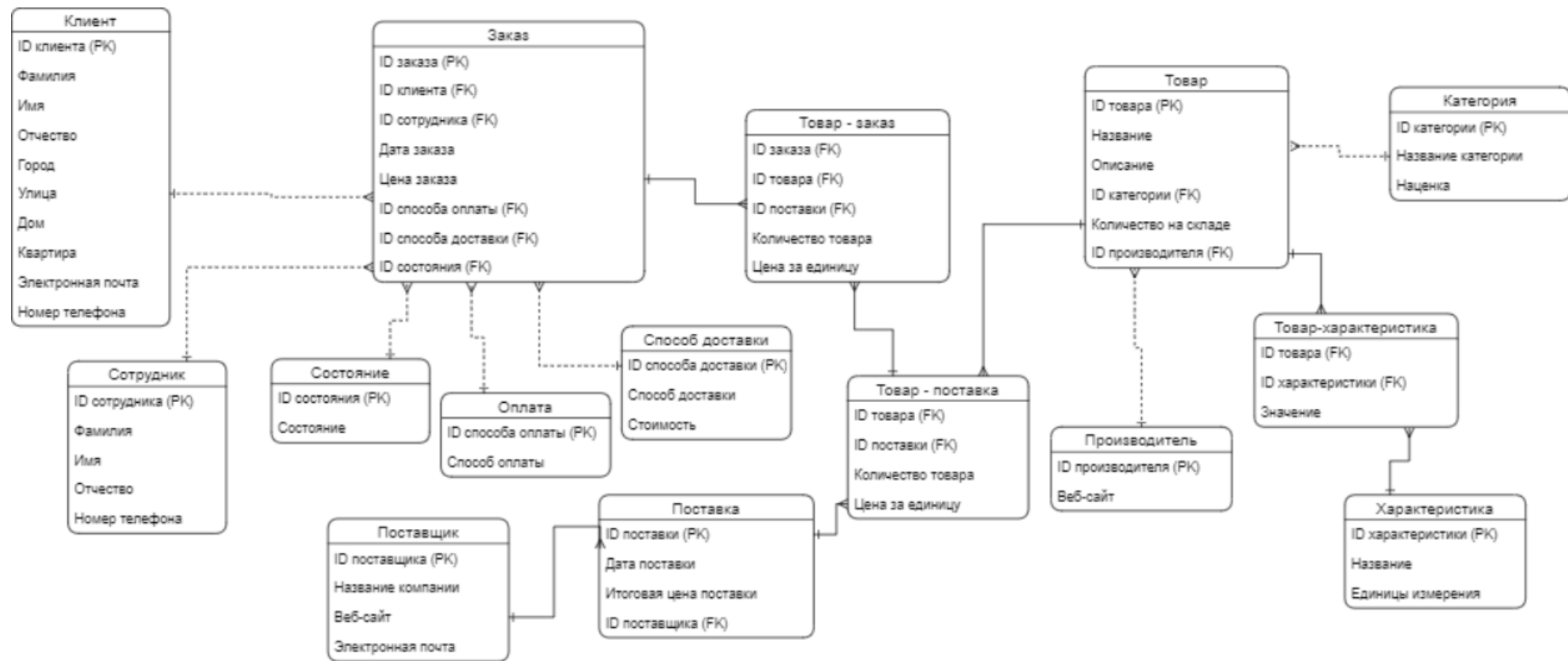


Рисунок 3 – Нормализованная логическая модель.

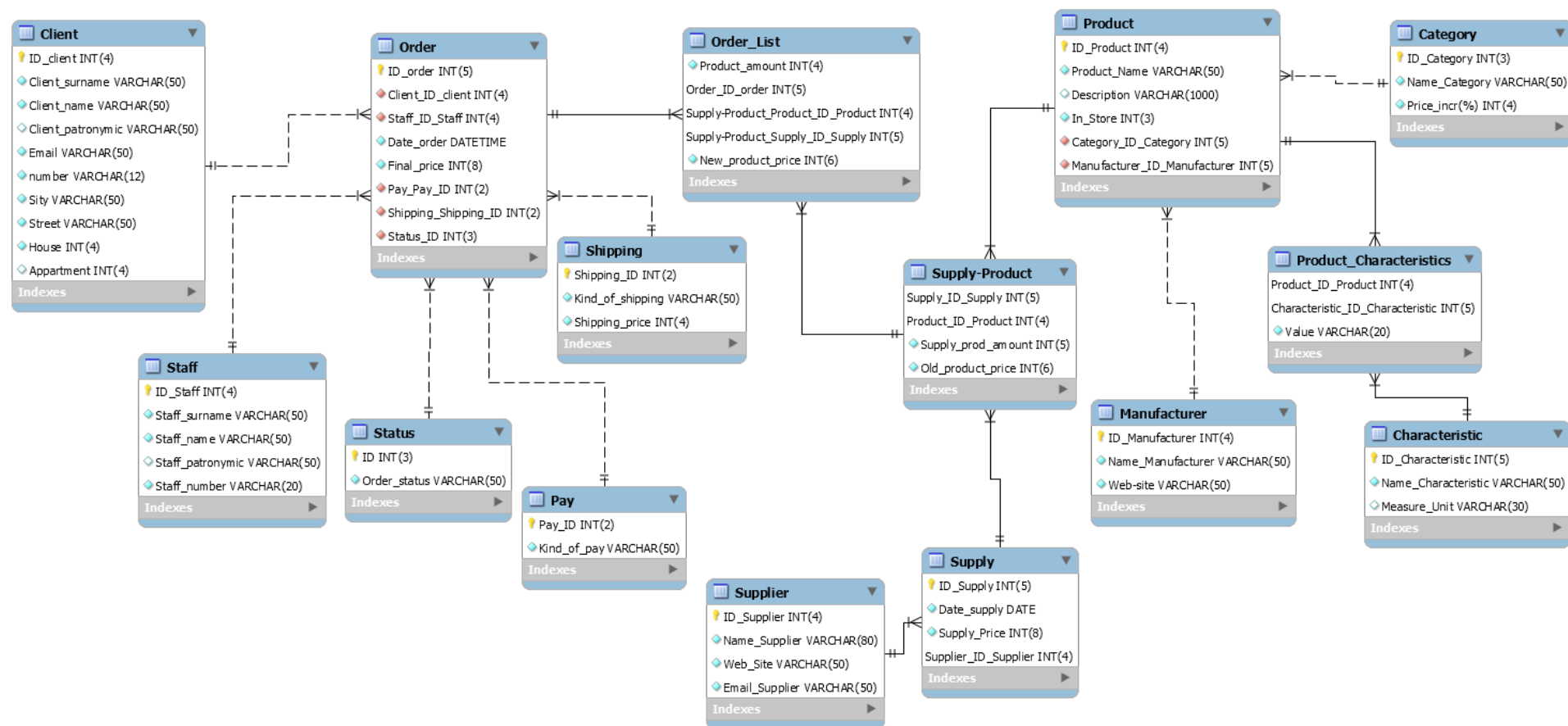


Рисунок 4 – Физическая модель.

## Приложение Б

### (Обязательное)

#### Скрипты создания таблиц

```
CREATE TABLE IF NOT EXISTS `MyStore`.`Client` (  
    `ID_client` INT(4) NOT NULL AUTO_INCREMENT,  
    `Client_surname` VARCHAR(50) COLLATE utf8mb4_unicode_ci  
NOT NULL,  
    `Client_name` VARCHAR(50) COLLATE utf8mb4_unicode_ci  
NOT NULL,  
    `Client_patronymic` VARCHAR(50) COLLATE  
utf8mb4_unicode_ci NULL,  
    `City` VARCHAR(50) COLLATE utf8mb4_unicode_ci NOT  
NULL,  
    `Street` VARCHAR(50) COLLATE utf8mb4_unicode_ci NOT  
NULL,  
    `House` INT(4) NOT NULL,  
    `Appartment` INT(4) NOT NULL,  
    `Email` VARCHAR(50) COLLATE utf8mb4_unicode_ci NOT  
NULL,  
    `Phone_number` VARCHAR(12) COLLATE utf8mb4_unicode_ci  
NOT NULL,  
    PRIMARY KEY (`ID_client`))  
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_unicode_ci;
```

```
CREATE TABLE IF NOT EXISTS `MyStore`.`Staff` (  
    `ID_Staff` INT(4) NOT NULL AUTO_INCREMENT,  
    `Staff_surname` VARCHAR(50) COLLATE utf8mb4_unicode_ci  
NOT NULL,  
    `Staff_name` VARCHAR(50) COLLATE utf8mb4_unicode_ci
```

```

NOT NULL,
    `Staff_patronymic` VARCHAR(50) COLLATE
utf8mb4_unicode_ci NULL,
    `Staff_number` VARCHAR(12) COLLATE utf8mb4_unicode_ci
NOT NULL,
    PRIMARY KEY (`ID_Staff`))
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

```

CREATE TABLE IF NOT EXISTS `MyStore`.`Pay` (
    `Pay_ID` INT(2) NOT NULL AUTO_INCREMENT,
    `Kind_of_pay` VARCHAR(50) COLLATE utf8mb4_unicode_ci
NOT NULL,
    PRIMARY KEY (`Pay_ID`))
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

```

CREATE TABLE IF NOT EXISTS `MyStore`.`Status` (
    `ID` INT(3) NOT NULL AUTO_INCREMENT,
    `Order_status` VARCHAR(50) COLLATE utf8mb4_unicode_ci
NOT NULL,
    PRIMARY KEY (`ID`))
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

```

CREATE TABLE IF NOT EXISTS `MyStore`.`Shipping` (
    `Shipping_ID` INT(2) NOT NULL AUTO_INCREMENT,
    `Kind_of_shipping` VARCHAR(50) COLLATE
utf8mb4_unicode_ci NOT NULL,
    `Shipping_Price` INT(4) NOT NULL,

```

```

PRIMARY KEY (`Shipping_ID`))
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

```

CREATE TABLE IF NOT EXISTS `MyStore`.`Order` (
  `ID_order` INT(5) NOT NULL AUTO_INCREMENT,
  `Client_ID_client` INT(4) NOT NULL,
  `Staff_ID_Staff` INT(4) NOT NULL,
  `Date_order` DATETIME NOT NULL,
  `Final_price` INT(8) NOT NULL,
  `Pay_Pay_ID` INT(2) NOT NULL,
  `Shipping_Shipping_ID` INT(2) NOT NULL,
  `Status_ID` INT(3) NOT NULL,
  PRIMARY KEY (`ID_order`),
  INDEX `fk_Order_Client_idx` (`Client_ID_client` ASC) VISIBLE,
  INDEX `fk_Order_Staff1_idx` (`Staff_ID_Staff` ASC) VISIBLE,
  INDEX `fk_Order_Pay1_idx` (`Pay_Pay_ID` ASC) VISIBLE,
  INDEX `fk_Order_Shipping1_idx` (`Shipping_Shipping_ID` ASC)
VISIBLE,
  INDEX `fk_Order_Status_idx` (`Status_ID` ASC) VISIBLE,
  FOREIGN KEY (`Client_ID_client`) REFERENCES
`MyStore`.`Client` (`ID_client`)
  ON UPDATE NO ACTION
  ON DELETE NO ACTION,
  FOREIGN KEY (`Staff_ID_Staff`) REFERENCES
`MyStore`.`Staff` (`ID_Staff`)
  ON UPDATE NO ACTION
  ON DELETE NO ACTION,
  FOREIGN KEY (`Pay_Pay_ID`) REFERENCES `MyStore`.`Pay`
(`Pay_ID`)

```



```

        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
        FOREIGN KEY (`Shipping_Shipping_ID`) REFERENCES
`MyStore`.`Shipping` (`Shipping_ID`)
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
        FOREIGN KEY (`Status_ID`) REFERENCES `MyStore`.`Status`
(`ID`)
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
    )
    ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
    COLLATE=utf8mb4_unicode_ci;

```

```

CREATE TABLE IF NOT EXISTS `MyStore`.`Supplier` (
    `ID_Supplier` INT(4) NOT NULL AUTO_INCREMENT,
    `Name_Supplier` VARCHAR(80) COLLATE utf8mb4_unicode_ci
NOT NULL,
    `Web_Site` VARCHAR(50) COLLATE utf8mb4_unicode_ci NOT
NULL,
    `Email_Supplier` VARCHAR(50) COLLATE utf8mb4_unicode_ci
NOT NULL,
    PRIMARY KEY (`ID_Supplier`))
    ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
    COLLATE=utf8mb4_unicode_ci;

```

```

CREATE TABLE IF NOT EXISTS `MyStore`.`Supply` (
    `ID_Supply` INT(5) NOT NULL AUTO_INCREMENT,
    `Date_supply` DATE NOT NULL,
    `Supply_Price` INT(8) NOT NULL,

```

```

`Supplier_ID_Supplier` INT(4) NOT NULL,
PRIMARY KEY (`ID_Supply`, `Supplier_ID_Supplier`),
INDEX `fk_Supply_Supplier1_idx` (`Supplier_ID_Supplier` ASC)
VISIBLE,
FOREIGN KEY (`Supplier_ID_Supplier`) REFERENCES
`MyStore`.`Supplier` (`ID_Supplier`)
ON UPDATE NO ACTION
ON DELETE NO ACTION)
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

```

CREATE TABLE IF NOT EXISTS `MyStore`.`Characteristic` (
  `ID_Characteristic` INT(5) NOT NULL AUTO_INCREMENT,
  `Name_Characteristic` VARCHAR(50) COLLATE
utf8mb4_unicode_ci NOT NULL,
  `Measure_Unit` VARCHAR(30) COLLATE utf8mb4_unicode_ci
NULL,
  PRIMARY KEY (`ID_Characteristic`))
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

```

CREATE TABLE IF NOT EXISTS `MyStore`.`Category` (
  `ID_Category` INT(3) NOT NULL AUTO_INCREMENT,
  `Name_Category` VARCHAR(50) COLLATE
utf8mb4_unicode_ci NOT NULL,
  `Price_incr(%)` INT(4) NOT NULL,
  PRIMARY KEY (`ID_Category`))
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

```

CREATE TABLE IF NOT EXISTS `MyStore`.`Manufacturer` (
  `ID_Manufacturer` INT(4) NOT NULL AUTO_INCREMENT,
  `Name_Manufacturer` VARCHAR(50) COLLATE
utf8mb4_unicode_ci NOT NULL,
  `Web-site` VARCHAR(50) COLLATE utf8mb4_unicode_ci NOT
NULL,
  PRIMARY KEY (`ID_Manufacturer`))
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

```

CREATE TABLE IF NOT EXISTS `MyStore`.`Product` (
  `ID_Product` INT(4) NOT NULL AUTO_INCREMENT,
  `Product_Name` VARCHAR(50) COLLATE
utf8mb4_unicode_ci NOT NULL,
  `Description` VARCHAR(1000) COLLATE utf8mb4_unicode_ci
NULL,
  `In_Store` INT(3) NOT NULL,
  `Category_ID_Category` INT(5) NOT NULL,
  `Manufacturer_ID_Manufacturer` INT(5) NOT NULL,
  PRIMARY KEY (`ID_Product`),
  INDEX `fk_Category_ID_Category_idx`
(`Category_ID_Category` ASC) VISIBLE,
  INDEX `fk_Manufacturer_ID_Manufacturer_idx`
(`Manufacturer_ID_Manufacturer` ASC) VISIBLE,
  FOREIGN KEY (`Category_ID_Category`) REFERENCES
`MyStore`.`Category` (`ID_Category`)
ON UPDATE NO ACTION
ON DELETE NO ACTION,
  FOREIGN KEY (`Manufacturer_ID_Manufacturer`)
REFERENCES `MyStore`.`Manufacturer` (`ID_Manufacturer`)

```

```

        ON UPDATE NO ACTION
        ON DELETE NO ACTION)
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

```

CREATE TABLE IF NOT EXISTS
`MyStore`.`Product_Characteristics` (
    `Product_ID_Product` INT(4) NOT NULL AUTO_INCREMENT,
    `Characteristic_ID_Characteristic` INT(5) NOT NULL,
    `Value` VARCHAR(20) COLLATE utf8mb4_unicode_ci NOT
NULL,
    PRIMARY KEY (`Product_ID_Product`,
`Characteristic_ID_Characteristic`),
    INDEX `fk_Product_Characteristics_Product1_idx`
(`Product_ID_Product` ASC) VISIBLE,
    INDEX `fk_Product_Characteristics_Characteristic1_idx`
(`Characteristic_ID_Characteristic` ASC) VISIBLE,
    FOREIGN KEY (`Product_ID_Product`) REFERENCES
`MyStore`.`Product` (`ID_Product`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
    FOREIGN KEY (`Characteristic_ID_Characteristic`)
REFERENCES `MyStore`.`Characteristic` (`ID_Characteristic`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

```

CREATE TABLE IF NOT EXISTS `MyStore`.`Supply-Product` (
    `Supply_ID_Supply` INT(5) NOT NULL,

```

```

        `Product_ID_Product` INT(4) NOT NULL,
        `Supply_prod_amount` INT(5) NOT NULL,
        `Old_product_price` INT(6) NOT NULL,
        INDEX `fk_table1_Supply1_idx` (`Supply_ID_Supply` ASC)
VISIBLE,
        INDEX `fk_table1_Product1_idx` (`Product_ID_Product` ASC)
VISIBLE,
        PRIMARY KEY (`Product_ID_Product`, `Supply_ID_Supply`),
        FOREIGN KEY (`Supply_ID_Supply`) REFERENCES
`MyStore`.`Supply` (`ID_Supply`)
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
        FOREIGN KEY (`Product_ID_Product`) REFERENCES
`MyStore`.`Product` (`ID_Product`)
        ON UPDATE NO ACTION
        ON DELETE NO ACTION)
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

```

CREATE TABLE IF NOT EXISTS `MyStore`.`Order_List` (
        `Product_amount` INT(4) NOT NULL,
        `Order_ID_order` INT(5) NOT NULL,
        `Supply-Product_Product_ID_Product` INT(4) NOT NULL,
        `Supply-Product_Supply_ID_Supply` INT(5) NOT NULL,
        `New_product_price` INT(6) NOT NULL,
        INDEX `fk_Order-Product_Order1_idx` (`Order_ID_order` ASC)
VISIBLE,
        PRIMARY KEY (`Order_ID_order`, `Supply-
Product_Product_ID_Product`, `Supply-Product_Supply_ID_Supply`),
        INDEX `fk_Order_List_Supply-Product1_idx` (`Supply-

```

```

Product_Product_ID_Product` ASC, `Supply-Product_Supply_ID_Supply`
ASC) VISIBLE,
        FOREIGN KEY (`Order_ID_order`) REFERENCES
`MyStore`.`Order` (`ID_order`)
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
        FOREIGN KEY (`Supply-Product_Product_ID_Product` ,
`Supply-Product_Supply_ID_Supply`) REFERENCES `MyStore`.`Supply-
Product` (`Product_ID_Product` , `Supply_ID_Supply`)
        ON UPDATE NO ACTION
        ON DELETE NO ACTION)
ENGINE = InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

## Приложение В

### (обязательное)

#### Скрипты заполнения таблиц

```
INSERT INTO `MyStore`.`Client` (`Client_surname`,
`Client_name`, `Client_patronymic`, `City`, `Street`, `House`, `Appartment`,
`Email`, `Phone_number`) VALUES
    ('Андреев', 'Сергей', 'Семенович', 'Орел', 'Покровская', 13, 33,
    'Andreev_Ivan94@mail.ru', '89107485050'),
    ('Ефремова', 'Елена', 'Алексеевна', 'Орел', 'Наугорское шоссе',
    74, 143, 'Lenka64@mail.ru', '89152056677'),
    ('Покровский', 'Андрей', 'Олегович', 'Орел', 'Герцена', 46, 55,
    'PoKrOV274@gmail.com', '89548348989'),
    ('Быков', 'Антон', 'Павлович', 'Орел', 'Наугорское шоссе', 74,
    89, 'Toxa99@mail.ru', '89776541345'),
    ('Гаврилов', 'Сергей', 'Геннадьевич', 'Орел', 'Максима
    Горького', 33, 10, 'SeriVolkD@gmail.com', '89158880974'),
    ('Петров', 'Олег', 'Анатолевич', 'Орел', 'Октябрьская', 65, 44,
    'PetrovI@gmail.com', '89107482033'),
    ('Покровский', 'Данил', 'Олегович', 'Орел', 'Наугорское шоссе',
    72, 99, 'VORKoP@mail.ru', '89996547848'),
    ('Данилова', 'Дарья', 'Борисовна', 'Орел', 'Ленина', 34, 78,
    'Dashka78@mail.ru', '89996540983');
```

```
INSERT INTO `MyStore`.`Staff` (`Staff_surname`, `Staff_name`,
`Staff_patronymic`, `Staff_number`) VALUES
    ('Алексеев', 'Павел', 'Павлович', '89990997878'),
    ('Борисов', 'Михаил', 'Денисович', '89764561234'),
    ('Семенов', 'Дмитрий', 'Петрович', '89567589779'),
    ('Польская', 'Варвара', 'Петровна', '89876784565'),
    ('Быков', 'Сергей', 'Павлович', '89999995678'),
```

```
( 'Норвежский', 'Антон', 'Игоревич', '89770932323'),
( 'Иванов', 'Иван', 'Иванович', '89991111919');
```

```
INSERT INTO `MyStore`.`Status` (`Order_status`) VALUES
( 'Ожидание оплаты'),
( 'Получена оплата'),
( 'Ожидание отправки'),
( 'Отправлено'),
( 'Ждет вас в пункте выдачи'),
( 'Ожидайте звонка от курьера'),
( 'Завершен');
```

```
INSERT INTO `MyStore`.`Pay` (`Kind_of_pay`) VALUES
( 'Наличный расчет'),
( 'Безналичный расчет');
```

```
INSERT INTO `MyStore`.`Shipping` (`Kind_of_shipping`,
`Shipping_Price`) VALUES
( 'Доставка курьером', 500),
( 'Самовывоз', 200);
```

```
INSERT INTO `MyStore`.`Supplier` (`Name_Supplier`,
`Web_Site`, `Email_Supplier`) VALUES
( 'Восточная Звезда', 'www.eastStar.ru', 'eastStar@mail.ru'),
( 'Ясон', 'www.Yason135.ru', 'YAsoNcomp@mail.ru'),
( 'ОрелЦифромаркет',
'www.Orelsifro.com', 'SifromarK@mail.ru'),
( 'Багира', 'www.bagiracomp.ru', 'Bagiracomp@mail.ru');
```

```
INSERT INTO `MyStore`.`Category` (`Name_Category`,
```



```
`Price_incr(%)`) VALUES
```

```
    ('Материнская плата',120 ),
    ('Процессор',115 ),
    ('Опреативная память', 110),
    ('Видеокарта', 110),
    ('Корпус', 115),
    ('Винчестер', 110),
    ('Блок питания', 125),
    ('Система охлаждения', 120);
```

```
INSERT INTO `MyStore`.`Characteristic` (`Name_Characteristic`,
`Measure_Unit`) VALUES
```

```
    ('Сокет', '-'),
    ('Чипсет', '-'),
    ('Мах кол-во процессоров', 'шт'),
    ('Формат платы', '-'),
    ('Мах объем оперативной памяти', 'Гб'),
    ('Тактовая частота', 'ГГц'),
    ('Частота шины', 'Gt/s'),
    ('Кол-во ядер', 'шт'),
    ('Умножение', '-'),
    ('Объем модуля памяти', 'Гб'),
    ('Частота функционирования', 'МГц'),
    ('Пропускающая способность памяти', 'Гб/сек'),
    ('Частота GPU', 'МГц'),
    ('Видеопамять', 'Гб'),
    ('Тип видеопамяти', '-'),
    ('Разрядность шины видеопамяти', 'бит'),
    ('Частота видеопамяти', 'МГц'),
    ('Цвет', '-'),
```

```

('Емкость', 'Тб'),
('Тип', '-'),
('Скорость чтения', 'Мб/сек'),
('Скорость записи', 'Мб/сек'),
('Время доступа', 'мс'),
('Тип охлаждения', '-'),
('Кол-во вентиляторов', 'шт'),
('Скорость вращения', 'об/мин'),
('Мощность', 'Вт');

```

```

INSERT INTO `MyStore`.`Supply` (`Date_supply`,
`Supply_Price`, `Supplier_ID_Supplier`) VALUES
('2019-05-12', 0, 1),
('2019-05-15', 0, 3),
('2019-07-24', 0, 4),
('2019-09-07', 0, 2),
('2019-09-30', 0, 4),
('2019-11-11', 0, 3);

```

```

INSERT INTO `MyStore`.`Manufacturer` (`Name_Manufacturer`,
`Web-Site`) VALUES
('AsRock', 'www.AsRock.com'),
('GIGABYTE', 'www.GIGABYTE.com'),
('ASUS', 'www.ASUS.com'),
('Intel', 'www.Intel.com'),
('AMD', 'www.AMD.com'),
('ADATA', 'www.ADATA.com'),
('Crucial', 'www.Crucial.com'),
('MSI', 'www.MSI.com'),
('Powercool', 'www.Powercool.com'),

```

```

('ExeGate', 'www.ExeGate.com'),
('Powerman', 'www.Powerman.com'),
('Seagate', 'www.Seagate.com'),
('Toshiba', 'www.Toshiba.com'),
('HGST', 'www.HGST.com'),
('Aerocool', 'www.Aerocool.com'),
('Chieftec', 'www.Chieftec.com'),
('Akasa', 'www.Akasa.com'),
('Alphacool', 'www.Alphacool.com'),
('Aardwolf', 'www.Aardwolf.com');

```

```

INSERT INTO `MyStore`.`Product` (`Product_Name`,
`Description`, `In_Store`, `Category_ID_Category`,
`Manufacturer_ID_Manufacturer`) VALUES
('B365', 'Материнская плата ASRock ...', 0, 1, 1),
('970 Pro3', 'Материнская плата ASRock ...', 0, 1, 1),
('M5A78L', 'Материнская плата ASUS ...', 0, 1, 3),
('GA-78LMT', 'Материнская плата GIGABYTE ...', 0, 1, 2),
('Core i5-4460', 'Процессор Intel ...', 0, 2, 4),
('Core i7-5820', 'Процессор Intel ...', 0, 2, 4),
('A10-7800', 'Процессор AMD ...', 0, 2, 5),
('FX-4350', 'Процессор AMD ...', 0, 2, 5),
('DDR3 DIMM', 'Модуль памяти ADATA...', 0, 3, 6),
('DDR4 DIMM CL17', 'Модуль памяти Crucial...', 0, 3, 7),
('DDR4 DIMM CL19', 'Модуль памяти Crucial...', 0, 3, 7),
('GeForce RTX 2080 Ti', 'Видеокарта GIGABYTE...', 0, 4, 2),
('GeForce GT 710', 'Видеокарта ASUS...', 0, 4, 3),
('GeForce GT 1030', 'Видеокарта MSI...', 0, 4, 8),
('GeForce GTX 1050', 'Видеокарта GIGABYTE...', 0, 4, 2),
('S602P', 'Корпус PowerCool...', 0, 5, 9),

```

('CP-604', 'Корпус EхеGate...', 0, 5, 10),  
 ('DA812BK', 'Корпус Powerman...', 0, 5, 11),  
 ('SATA', 'Винчестер GIGBYTE...', 0, 6, 2),  
 ('Barracuda', 'Винчестер Seagate...', 0, 6, 12),  
 ('P300', 'Винчестер Toshiba...', 0, 6, 13),  
 ('TravelStar', 'Винчестер HGST...', 0, 6, 14),  
 ('KCAS PLUS-400w', 'Блок питания Aerocool...', 0, 7, 15),  
 ('VX-450', 'Блок питания Aerocool...', 0, 7, 15),  
 ('Smart GPS-400A8', 'Блок питания Chieftec...', 0, 7, 16),  
 ('Venom A10', 'Водяное охлаждение Akasa...', 0, 8, 17),  
 ('Eisbaer LT360', 'Водяное охлаждение Alphacool', 0, 8, 18),  
 ('Performa V120', 'Кулер Aardwolf...', 0, 8, 19);

INSERT INTO `MyStore`.`Supply-Product` VALUES

(1, 1, 30, 8000),  
 (1, 8, 20, 3000),  
 (1, 15, 30, 9000),  
 (1, 22, 40, 4000),  
 (1, 26, 20, 6000),  
 (2, 2, 50, 5000),  
 (2, 3, 40, 4000),  
 (2, 17, 15, 2000),  
 (2, 21, 50, 3000),  
 (2, 24, 25, 2000),  
 (3, 9, 50, 1500),  
 (3, 12, 10, 86000),  
 (3, 19, 20, 8000),  
 (4, 5, 35, 14000),  
 (4, 10, 30, 4500),  
 (4, 13, 25, 3000),

```
(4, 18, 20, 4000),
(4, 25, 10, 2000),
(4, 28, 15, 1500),
(5, 4, 15, 14000),
(5, 6, 20, 13000),
(5, 16, 25, 3000),
(5, 23, 10, 3000),
(5, 27, 20, 10000),
(6, 7, 40, 6500),
(6, 11, 30, 5000),
(6, 14, 35, 5000),
(6, 20, 30, 9000);
```

```
INSERT INTO `MyStore`.`Order` (`Client_ID_Client`,
`Staff_ID_Staff`, `Date_order`, `Final_price`, `Pay_Pay_ID`,
`Shipping_Shipping_ID`, `Status_ID`) VALUES
(1, 1, '2019-10-15 01:29:31', 0, 1, 2, 7),
(2, 2, '2019-11-21 05:44:01', 0, 2, 1, 7),
(3, 2, '2019-11-29 12:11:24', 0, 1, 1, 7),
(4, 6, '2019-12-05 07:55:44', 0, 2, 2, 5),
(1, 2, '2019-12-08 19:08:42', 0, 2, 2, 3);
```

```
INSERT INTO `MyStore`.`Order_List` VALUES
(2, 1, 15, 1, 0),
(5, 2, 10, 4, 0),
(4, 2, 26, 1, 0),
(10, 3, 19, 3, 0),
(5, 3, 20, 6, 0),
(1, 4, 12, 3, 0),
(12, 5, 16, 5, 0),
```

```
(1, 5, 6, 5, 0),  
(2, 5, 17, 2, 0);
```

```
INSERT INTO `MyStore`.`Product_Characteristics` VALUES  
  (1, 1, 'LGA1151'),  
  (1, 2, 'Intel B365'),  
  (1, 3, '1'),  
  (1, 4, 'ATX'),  
  (1, 5, '64'),  
  (2, 1, 'AM3'),  
  (2, 2, 'AMD 970'),  
  (2, 3, '1'),  
  (2, 4, 'ATX'),  
  (2, 5, '32'),  
  (3, 1, 'AM3'),  
  (3, 2, 'AMD 760G'),  
  (3, 3, '1'),  
  (3, 4, 'microATX'),  
  (3, 5, '32'),  
  (4, 1, 'AM3'),  
  (4, 2, 'AMD 760G'),  
  (4, 3, '1'),  
  (4, 4, 'microATX'),  
  (4, 5, '16'),  
  (5, 1, 'LGA1151'),  
  (5, 5, '32'),  
  (5, 6, '3.2'),  
  (5, 7, '5'),  
  (5, 8, '4'),  
  (5, 9, '32'),
```

(6, 1, 'LGA2011-3'),  
(6, 5, '64'),  
(6, 6, '3.3'),  
(6, 7, '5'),  
(6, 8, '6'),  
(6, 9, '33'),  
(7, 1, 'FM2 plus'),  
(7, 5, '64'),  
(7, 6, '3.5'),  
(7, 7, '5'),  
(7, 8, '4'),  
(7, 9, '35'),  
(8, 1, 'AM3'),  
(8, 5, '128'),  
(8, 6, '4.2'),  
(8, 7, '5.2'),  
(8, 8, '4'),  
(8, 9, '21'),  
(9, 10, '4'),  
(9, 11, '1600'),  
(9, 12, '12.5'),  
(10, 10, '16'),  
(10, 11, '2400'),  
(10, 12, '19'),  
(11, 10, '16'),  
(11, 11, '2666'),  
(11, 12, '21'),  
(12, 12, '616'),  
(12, 13, '1665'),  
(12, 14, '11'),

(12, 15, 'GDDR6'),  
(12, 16, '352'),  
(12, 17, '3500'),  
(13, 12, '500'),  
(13, 13, '954'),  
(13, 14, '1'),  
(13, 15, 'GDDR5'),  
(13, 16, '32'),  
(13, 17, '1252'),  
(14, 12, '550'),  
(14, 13, '1430'),  
(14, 14, '2'),  
(14, 15, 'DDR4'),  
(14, 16, '64'),  
(14, 17, '2100'),  
(15, 12, '550'),  
(15, 13, '1493'),  
(15, 14, '2'),  
(15, 15, 'GDDR5'),  
(15, 16, '128'),  
(15, 17, '1752'),  
(16, 18, 'Черный'),  
(17, 18, 'Синий'),  
(18, 18, 'Черный'),  
(19, 19, '1'),  
(19, 20, 'SSD'),  
(19, 21, '560'),  
(19, 22, '520'),  
(19, 23, '-'),  
(20, 19, '1'),



(20, 20, 'SSD'),  
(20, 21, '560'),  
(20, 22, '530'),  
(20, 23, '-'),  
(21, 19, '1'),  
(21, 20, 'HDD'),  
(21, 21, '-'),  
(21, 22, '-'),  
(21, 23, '4.2'),  
(22, 19, '1'),  
(22, 20, 'HDD'),  
(22, 21, '-'),  
(22, 22, '-'),  
(22, 23, '4.2'),  
(23, 27, '400'),  
(24, 27, '450'),  
(25, 27, '450'),  
(26, 24, 'Водяное охлаждение'),  
(26, 25, '1'),  
(26, 26, '1900'),  
(27, 24, 'Водяное охлаждение'),  
(27, 25, '2'),  
(27, 26, '1700'),  
(28, 24, 'Кулер для процессора'),  
(28, 25, '1'),  
(28, 26, '1500');

**Приложение Г****(обязательное)****Скрипты триггеров и процедур**

```
DELIMITER //
```

```
CREATE TRIGGER `negative_order`
```

```
BEFORE INSERT ON `Order`
```

```
FOR EACH ROW
```

```
BEGIN
```

```
IF NEW.`Final_Price` < 0
```

```
THEN SIGNAL SQLSTATE '45000' SET message_text = 'Error! Wrong  
count.';
```

```
END IF;
```

```
END //
```

```
DELIMITER ;
```

```
DELIMITER //
```

```
CREATE TRIGGER `negative_supply`
```

```
BEFORE INSERT ON `Supply`
```

```
FOR EACH ROW
```

```
BEGIN
```

```
IF NEW.`Supply_Price` < 0
```

```
THEN SIGNAL SQLSTATE '45000' SET message_text = 'Error! Wrong  
count.';
```

```
END IF;
```

```
END //
```

```
DELIMITER ;
```

```
DELIMITER //
```

```
CREATE TRIGGER `negative_product`
```

```
BEFORE INSERT ON `Product`
```

```

FOR EACH ROW
BEGIN
    IF NEW.`In_Store` < 0
    THEN SIGNAL SQLSTATE '45000' SET message_text = 'Error! Wrong
count.';
    END IF;
END //
DELIMITER ;

DELIMITER //
CREATE TRIGGER `negative_sup_prod`
BEFORE INSERT ON `Supply-Product`
FOR EACH ROW
BEGIN
    IF (NEW.`Old_product_price` < 0) OR (NEW.`Supply_prod_amount` <=
0)
    THEN SIGNAL SQLSTATE '45000' SET message_text = 'Error! Wrong
count.';
    END IF;
END //
DELIMITER ;

DELIMITER //
CREATE TRIGGER `negative_order_list`
BEFORE INSERT ON `Order_List`
FOR EACH ROW
BEGIN
    IF (NEW.`New_product_price` < 0) OR (NEW.`Product_amount` <= 0)
    THEN SIGNAL SQLSTATE '45000' SET message_text = 'Error! Wrong
count.';

```

```
END IF;
```

```
END //
```

```
DELIMITER ;
```

```
--Триггер для supply-Product
```

```
DELIMITER //
```

```
CREATE TRIGGER `prod_incr`
```

```
AFTER INSERT ON `Supply-Product`
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    UPDATE `Product`
```

```
    SET `Product`.`In_Store`=`Product`.`In_Store` +
```

```
NEW.`Supply_prod_amount`
```

```
    WHERE `Product`.`ID_Product` = NEW.`Product_ID_Product` ;
```

```
END//
```

```
DELIMITER ;
```

```
--Триггер для заполнения итоговой стоимости поставки
```

```
DELIMITER //
```

```
CREATE TRIGGER `supply_price_incr`
```

```
AFTER INSERT ON `Supply-Product`
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    UPDATE `Supply`
```

```
    SET `Supply`.`Supply_Price`=`Supply`.`Supply_Price` +
```

```
(NEW.`Supply_prod_amount` * NEW.`Old_product_price`)
```

```
    WHERE `Supply`.`ID_Supply` = NEW.`Supply_ID_Supply` ;
```

```
END//
```

```
DELIMITER ;
```

```
--Триггер для стартовой стоимости заказа
DELIMITER //
CREATE TRIGGER `start_order_price`
BEFORE INSERT ON `Order`
FOR EACH ROW
BEGIN
    SET NEW.`Final_price`=(SELECT `Shipping_Price` FROM `Shipping`
WHERE `Shipping`.`Shipping_ID`= NEW.`Shipping_Shipping_ID`);
END//
DELIMITER ;
```

```
--Триггер для заполнения итоговой стоимости заказа
DELIMITER //
CREATE TRIGGER `order_price_incr`
AFTER INSERT ON `Order_List`
FOR EACH ROW
BEGIN
    UPDATE `Order`
    SET `Order`.`Final_Price`=`Order`.`Final_Price` +
    (NEW.`Product_amount` * NEW.`New_product_price`)
    WHERE `Order`.`ID_order` = NEW.`Order_ID_order` ;
END//
DELIMITER ;
```

```
--Триггер для проверки кол-ва товара на складе при заказе
DELIMITER //
CREATE TRIGGER `prod_check`
BEFORE INSERT ON `Order_List`
FOR EACH ROW
```

```

BEGIN
    IF (SELECT `In_Store` FROM `Product` WHERE
`Product`.`ID_Product` = NEW.`Supply-Product_Product_ID_Product`) >=
NEW.`Product_amount` THEN
        UPDATE `Product` SET `Product`.`In_Store` = `Product`.`In_Store` -
NEW.`Product_amount`
        WHERE `Product`.`ID_Product` = NEW.`Supply-
Product_Product_ID_Product`;
    ELSE set @message_text = concat('На данный момент такого
количества товара нет:', NEW.`Product_amount`);
    SIGNAL SQLSTATE '45000'
    set message_text = @message_text;
END IF;
END //
DELIMITER ;

--Триггер для получения новой цены
DELIMITER //
CREATE TRIGGER `get_new_price`
BEFORE INSERT ON `Order_List`
FOR EACH ROW
BEGIN
    SET NEW.`New_product_price` =
    (
        SELECT `Supply-Product`.`Old_product_price` *
`Category`.`Price_incr(%)` / 100
        FROM `Supply-Product` INNER JOIN `Product` ON
`Product`.`ID_Product` = `Supply-Product`.`Product_ID_Product`
        INNER JOIN `Category` ON `Product`.`Category_ID_Category` =
`Category`.`ID_Category`
    )

```

```

WHERE (`Product_ID_Product` = `ID_Product`) AND
(`Category_ID_Category` = `ID_Category`) AND (NEW.`Supply-
Product_Product_ID_Product` = `Product_ID_Product`) AND
(NEW.`Supply-Product_Supply_ID_Supply` = `Supply_ID_Supply`)
);
END//
DELIMITER ;

```

--Процедура для создания нового заказа

```

DELIMITER //

CREATE PROCEDURE `Create_Order` (c_id INT(5), s_id INT(5), p_id
INT(2), sh_id INT(2), st_ID INT(3))
BEGIN
    INSERT INTO `Order` (`Client_ID_Client`, `Staff_ID_Staff`,
`Date_order`, `Final_price`, `Pay_Pay_ID`, `Shipping_Shipping_ID`,
`Status_ID` )
        VALUES (c_id, s_id, NOW(), 0, p_id, sh_id, st_ID);
    END //
DELIMITER ;

```

--Процедура для вывода всех незавершенных заказов сотрудника

```

DELIMITER //

CREATE PROCEDURE `Check_staff` (id INT(5))
BEGIN
    SELECT DISTINCT `Order`.`ID_order` AS `ID`, `Order`.`Date_order`
AS `Date`, `Order`.`Final_price` AS `Price`, `Client`.`Client_surname` AS
`Client`, `Status`.`Order_status` AS `Status`
    FROM `Order`, `Client`, `Staff`, `Status`
    WHERE (`Order`.`Status_ID` != 7) AND (`Order`.`Staff_ID_Staff` = id)
AND (`Order`.`Client_ID_client` = `Client`.`ID_client`) AND

```

```
(`Order`.`Status_ID` = `Status`.`ID`)  
;  
END //  
DELIMITER ;
```



**Приложение Д**  
**(обязательное)**  
**Скрипты запросов**

-- Выручка за месяцы

```
SELECT MONTHNAME(`Order`.`Date_order`) AS `Month`,
YEAR(`Order`.`Date_order`) AS `Year`,
SUM(`Order_List`.`Product_amount` *
`Order_List`.`New_product_price`) AS `Total`
FROM `Order`, `Order_List`
WHERE (`Order`.`ID_Order` = `Order_List`.`Order_ID_Order`)
AND (YEAR(`Order`.`Date_order`) = 2019)
GROUP BY MONTHNAME(`Order`.`Date_order`);
```

--Пять производителей, продукции которых более всего в магазине

```
SELECT `Manufacturer`.`Name_Manufacturer` AS `Manufacturer`,
COUNT(`Manufacturer`.`ID_Manufacturer`) AS `Amount`
FROM `Product`, `Manufacturer`
WHERE `Product`.`Manufacturer_ID_Manufacturer` =
`Manufacturer`.`ID_Manufacturer`
GROUP BY `Manufacturer`.`ID_Manufacturer`
ORDER BY `Amount` DESC
LIMIT 5;
```

--Вывод всех характеристик всех товаров и их производителей

```
SELECT `Manufacturer`.`Name_Manufacturer` AS
`Manufacturer`, `Product`.`Product_Name` AS `Product`,
`Characteristic`.`Name_Characteristic` AS `Characteristic`,
`Product_Characteristics`.`Value` AS `Value`,
```

```

`Characteristic`.`Measure_Unit` AS `Measure Unit`
FROM `Product`, `Product_Characteristics`, `Characteristic`,
`Manufacturer`
WHERE (`Manufacturer`.`ID_Manufacturer` =
`Product`.`Manufacturer_ID_Manufacturer`) AND
(`Product`.`ID_Product` =
`Product_Characteristics`.`Product_ID_Product`) AND
(`Characteristic`.`ID_Characteristic` =
`Product_Characteristics`.`Characteristic_ID_Characteristic`)
ORDER BY `Product`.`Product_Name`;

```

```

--Три лучших клиента
SELECT `Client`.`Client_surname` AS `Surname`,
`Client`.`Client_name` AS `Name`, `Client`.`Client_patronymic` AS
`Patronymic`, SUM(`Order`.`Final_Price`) AS `MONEY`
FROM `Client`, `Order`
WHERE (`Client`.`ID_client` = `Order`.`Client_ID_Client`)
GROUP BY `Client`.`ID_Client`
ORDER BY `Money` DESC
LIMIT 3;

```

```

--Ведущий работник(С наибольшим числом обработанных
заказов)
SELECT `Staff`.`Staff_Surname` AS `Surname`,
`Staff`.`Staff_name` AS `Name`, `Staff`.`Staff_patronymic` AS
`Patronymic`, COUNT(`Staff`.`ID_Staff`) AS `Order_Amount`
FROM `Staff`, `Order`
WHERE (`Staff`.`ID_Staff` = `Order`.`Staff_ID_Staff`) AND
(`Order`.`Status_ID` = 7)
GROUP BY `Staff`.`ID_Staff`

```

```
ORDER BY `Order_Amount` DESC  
LIMIT 1;
```