Министерство науки и высшего образования РФ  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
“Забайкальский государственный университет”  
Энергетический факультет  
Кафедра информатики и вычислительной техники

**Курсовой проект**  
по предмету:  
**Базы данных**

На тему:

**Разработка реляционной базы данных по выбранной предметной области (Ведение заказов)**

**Выполнила:**  
Студентка: Борисова. Е. О.  
Группа: ИВТ-20  
Курс: Второй   
Форма обучения: Очная

**Руководитель:-**  
 Должность: Доцент кафедры ИВТ и ПМ  
 ФИО: Гончаров Денис Сергеевич

Чита - 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ   
Федеральное государственное бюджетное образовательное учреждение высшего образования   
«Забайкальский государственный университет»   
(ФГБОУ ВО «ЗабГУ»)   
Факультет: Энергетический факультет  
 Кафедра: Информатики и вычислительной техники

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА   
к курсовому проекту

По дисциплине: Базы данных

На тему: создание базы данных для ведения заказов

Цели курсовой работы:

1. Приобретение навыков анализа и моделирования предметной области

2. Реализация модели «сущность-связь» с помощью специализированных

CASE-средств (pgModeler)

3. Приобретение навыков моделирования реляционных баз данных

методом нормализации отношений;

4. Реализация декларативных ограничений целостности; проектирование

таблиц БД с использованием pgModeler и pgAdmin

5. Реализация запросов к БД

6. Реализация процедурных ограничений целостности; создание

триггеров в СУБД PostgreSQL

Выполнила студентка группы ИВТ-20 Борисова Екатерина Олеговна

Руководитель работы: Доцент кафедры ИВТ и ПМ Гончаров Денис Сергеевич

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ   
Федеральное государственное бюджетное образовательное учреждение высшего образования   
«Забайкальский государственный университет»   
(ФГБОУ ВО «ЗабГУ»)   
Факультет: Энергетический факультет Кафедра:   
Информатики и вычислительной техники

ЗАДАНИЕ   
на курсовую работу

По дисциплине базы данных

Студентке Борисовой Екатерине Олеговне

специальности Информатика и Вычислительная техника

1. Тема курсовой работы: Создание базы данных для ведения заказов

2. Срок подачи студентом законченного проекта: 11.06.2022

3. Исходные данные к работе: Рабочая сфера – компания, занимающаяся оптовой продажей различных товаров. Создать базу данных для отслеживание финансовой стороны деятельности компании.

4. Перечень подлежащих разработке курсового проекта вопросов:

Схема БД, Нормализация БД, Ограничения целостности БД, Запросы к БД, Триггеры.

Дата выдачи задания: 12.02.2022

Руководитель курсового проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Задание принял к исполнению  
«1» мая 20\_\_г.  
Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

УТВЕРЖДАЮ

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

Календарный план

выполнения курсового проекта

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Этапы выполнения курсовой работы | Месяцы и недели | | | | | | | | | | | | | | | |
| Март | | | | Апрель | | | | Май | | | | Июнь | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1 | Получение задания на курсовой проект | + | + | + |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Анализ предметной области |  | + | + | + | + | + |  |  |  |  |  |  |  |  |  |  |
| 3 | Написание разделов |  |  |  |  | + | + | + | + | + |  |  |  |  |  |  |  |
| 4 | Оформление работы |  |  |  |  |  |  |  |  | + | + | + | + |  |  |  |  |
| 5 | Представление чернового варианта |  |  |  |  |  |  |  |  |  |  |  | + |  |  |  |  |
| 6 | Защита работы |  |  |  |  |  |  |  |  |  |  |  |  |  | + |  |  |

План выполнен: руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, расшифровка подписи)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**РЕФЕРАТ**

Курсовая работа по предмету базы данных, на тему : «Разработка реляционной базы данных по выбранной предметной области (ведение заказов)», студентки 2-го курса Борисовой Е.О.

Курсовая работа проект состоит из: введения, двух глав, заключения и списка литературы. Содержит вставки кода на языке запросов SQL, скриншоты результатов работы, таблиц для анализа и диаграммы “сущность-связь”.

Цель работы: Приобретение навыков моделирования, разработки и работы с базой данных

Ключевые слова: База данных, SQL, запрос, триггер, таблица, данные, анализ.

**Содержание**

[**Введение** 7](#_Toc106409787)

[**Глава 1. Теоретические сведения** 8](#_Toc106409788)

[**Глава 2. Практическая часть** 13](#_Toc106409789)

[**2.1 Описание предметной области.** 13](#_Toc106409790)

[**2.2 Диаграмма «Сущность-связь»** 13](#_Toc106409791)

[**2.3. Создание физической модели базы данных** 15](#_Toc106409792)

[2.3.1. Описание декларативных ограничений целостности 15](#_Toc106409793)

[2.3.2. Декларация таблиц БД 17](#_Toc106409794)

[2.3.3. Операторы DML 19](#_Toc106409795)

[**2.4. Анализ таблиц на соответствие нормальным формам (3НФ)** 23](#_Toc106409796)

[**2.5. Описание запросов к БД. Оператор SELECT** 24](#_Toc106409797)

[**2.6. Описание хранимых процедур и триггеров** 26](#_Toc106409798)

[**Заключение** 30](#_Toc106409799)

[**Ссылки на источники:** 31](#_Toc106409800)

# **Введение**

База данных (общее понятие) – это представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ).

Главная задача данной курсовой работы – создать базу данных (состоящую из нескольких таблиц) в виде СУБД, которая создаётся с помощью объектно-реляционной системы управления базами данных PostgreSQL с использованием инструмента визуального проектирования pgModeler и платформы администрирования pgAdmin.

База данных (в информатике) – это совокупность массивов и файлов

данных, организованная по определённым правилам, предусматривающим

стандартные принципы описания, хранения и обработки данных независимо от их вида.

**Системой управления базами данных (СУБД)** называется совокупность программных средств, необходимых для использования базы данных и предоставляющих разработчикам и пользователям множество различных представлений данных.

Для выполнения данной курсовой работы были поставлена цель: проанализировать предметную область и создать модель «сущность-связь».

Исходя из этого был выдвинут свой ряд задач:

1) осуществить поиск информации;

2) изучить материал;

3) создать и протестировать базу данных.

# **Глава 1. Теоретические сведения**

**База данных** – это коллекции таблиц, в которой хранится определенный набор структурированных данных. Таблица содержит коллекцию строк, также называемых записями или кортежами, и столбцов, также называемых атрибутами. Каждый столбец в таблице предназначен для хранения конкретного типа данных, например, дат, имен, денежных сумм или чисел.

Первое, что мы создаем при разработке базы данных – это ее модель. Для этой работы нам понадобится специальное приложение. Одно из таких – pgModeler.

**pgModeler** — это свободный и открытый, кроссплатформенный инструмент визуального проектирования баз данных, объединяющий собой классические диаграммы сущность-связь с особенностями PostgreSQL.

Создав модель нам нужно провести ее анализ. Для этого нам нужно провести нормализацию базы данных. В идеале результат будет ровно такой же, как схема, созданная ранее.

Нормальные формы – это рекомендации по проектированию баз данных. Вы не обязаны придерживаться всех пяти нормальных форм при проектировании баз данных. Тем не менее, рекомендуется нормализовать базу данных в некоторой степени потому, что этот процесс имеет ряд существенных преимуществ с точки зрения эффективности и удобства обращения с вашей базой данных.

1. В нормализованной структуре базы данных вы можете производить сложные выборки данных относительно простыми SQL-запросами.
2. **Целостность данных**. Нормализованная база данных позволяет надежно хранить данные.
3. Нормализацияпредотвращает появление избыточности хранимых данных. Данные всегда хранятся только в одном месте, что делает легким процесс вставки, обновления и удаления данных. Есть исключение из этого правила. Ключи, сами по себе, хранятся в нескольких местах потому, что они копируются как внешние ключи в другие таблицы.
4. Масштабируемость – это возможность системы справляться с будущим ростом. Для базы данных это значит, что она должна быть способна работать быстро, когда число пользователей и объемы данных возрастают.

**Первая нормальная форма**

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

**Вторая нормальная форма**

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа(ПК).  
Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.

**Третья нормальная форма**

Отношение находится в 3НФ, когда находится во 2НФ 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.

**Нормальная форма Бойса-Кодда (НФБК) (частная форма третьей нормальной формы)**

Определение 3НФ не совсем подходит для следующих отношений:  
1) отношение имеет два или более потенциальных ключа;  
2) два и более потенциальных ключа являются составными;  
3) они пересекаются, т.е. имеют хотя бы один общий атрибут.  
Для отношений, имеющих один потенциальный ключ (первичный), НФБК является3НФ.  
Отношение находится в НФБК, когда каждая нетривиальная и неприводимая слева функциональная зависимость обладает потенциальным ключом в качестве детерминанта.

**Четвертая нормальная форма**

Отношение находится в 4НФ, если оно находится в НФБК и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от ее потенциальных ключей.

**Пятая нормальная форма**

Отношения находятся в 5НФ, если оно находится в 4НФ и отсутствуют сложные зависимые соединения между атрибутами.

Стоит отметить, что нормализация позволяет опустить предыдущий этап, но это все же нежелательно.

Следующим шагом в создании базы данных является ограничение целостности данных.

**Ограничения целостности баз данных** — это специальные средства в [базах данных](http://cyclowiki.org/w/index.php?title=%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85&action=edit&redlink=1), главное назначение которых — не допустить попадания в базу ошибочных данных, например — тридцатый день в [феврале](http://cyclowiki.org/wiki/%D0%A4%D0%B5%D0%B2%D1%80%D0%B0%D0%BB%D1%8C) или восьмой день недели.

**Первичный ключ (primarykey)**

Используется для обеспечения уникальности данных в столбцах и, в основном, для обеспечения ссылок на другие таблицы посредством связывания их внешними ключами.

**Внешний ключ (foreignkey)**

Применяется вместе с определённым раннее первичным ключом или же ограничением уникальности (unique) в связанной таблице. Условие на значение внешнего ключа одной таблицы ставит в соответствие один или несколько столбцов другой таблицы.

**Ограничение уникальности (unique)**

Назначается чтобы запретить повторение значений в столбце таблицы. Для столбца, на котором определено ограничение первичного ключа, не может быть определено ограничение уникальности, так как уникальный индекс данного столбца уже создан.

**Проверочное ограничение (check)**

Устанавливает, какие значения может хранить столбец. Это ограничение, например, можно использовать для столбца, хранящего номера квартир в многоквартирном доме.

**NotNull**

Устанавливает поля, которые не могут быть пустыми.

Дальнейшие ограничения создаются при помощи триггеров.

**Триггер** – это особая разновидность хранимых процедур в базе данных. Особенность триггеров заключается в том, что SQL код, написанные в теле триггера, будет исполнен после того, как в базе данных произойдет какое-либо событие.

Главная особенность триггеров в том, что они позволяют работать с добавлением/изменением/удалением данных не только в рамках одной таблицы. К примеру, у нас есть таблицы склада и покупок в интернет магазине. Когда пользователь попытается что-то купить должна добавиться запись о покупке и измениться запись о количестве товара на складе. Это мы и сможем реализовать за счет триггеров.

Ну и думаю последний этап – это тестирование базы данных. Для этого необходимо провести ряд разнообразных запросов к базе данных, по результатам которых мы выявим ошибки и исправим их, если такие имеются.

**Запросы** — **это** объект **базы** **данных**, который служит для извлечения **данных** из таблиц и предоставления их пользователю в удобном виде.

Думаю, на этом можно закончить основной процесс разработки базы данных. Рассмотрим каждый из этих этапов подробнее на примере базы данных для учета стоимости прошедшей в эфире рекламы.

# **Глава 2. Практическая часть**

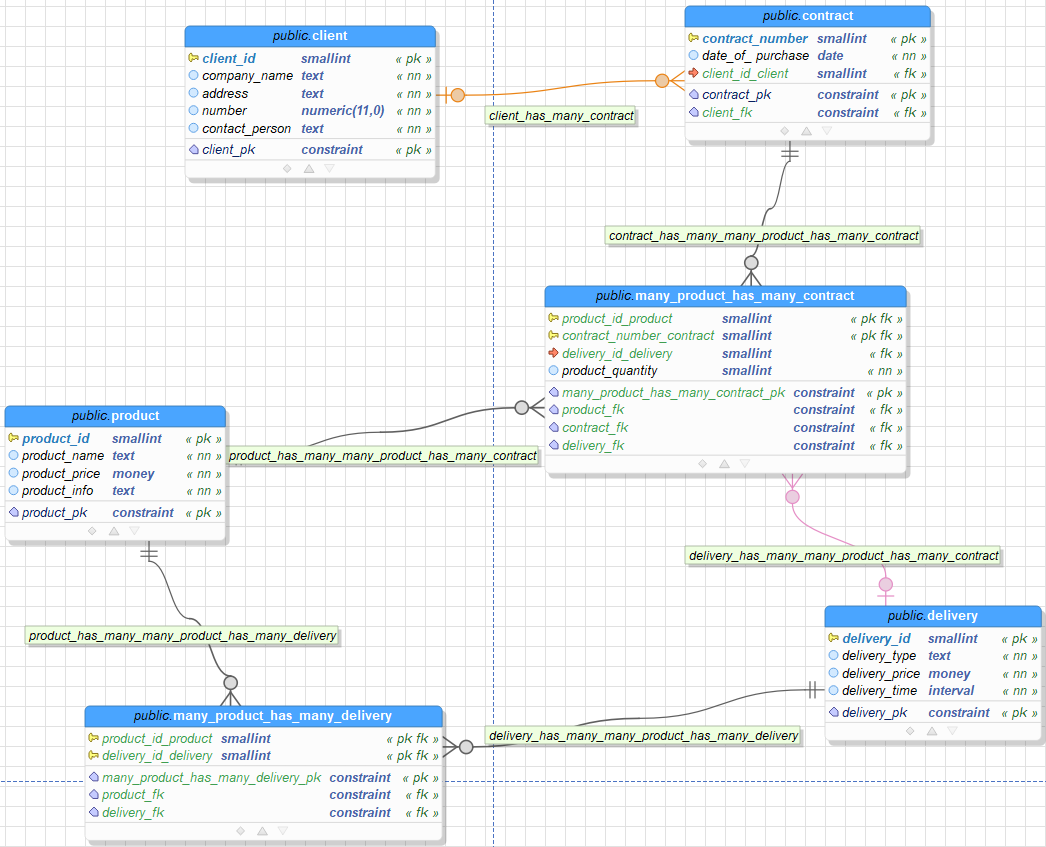
## **2.1 Описание предметной области.**

Вы работаете в компании, занимающейся оптовой продажей различных товаров. Вашей задачей является отслеживание финансовой стороны ее работы. Деятельность компании организована следующим образом: компания торгует товарами из определенного спектра. Каждый из этих товаров характеризуется ценой, справочной информацией и признаком наличия или отсутствия доставки. В вашу компанию обращаются заказчики. Для каждого из них вы запоминаете в базе данных стандартные данные (наименование, адрес, телефон, контактное лицо) и составляете по каждой сделке документ, запоминая наряду с заказчиком количество купленного им товара и дату покупки.

Доставка разных товаров может производиться способами, различными по цене и скорости. Нужно хранить информацию о том, какими способами может осуществляться доставка каждого товара, и о том, какой вид доставки (а соответственно, и какую стоимость доставки) выбрал клиент при заключении сделки.

## **2.2 Диаграмма «Сущность-связь»**

В процессе анализа рабочей сферы была создана следующая схема:



1. Сущность Клиент (client) имеет 5 атрибутов:

* client\_id – первичный ключ компании;
* company\_name – название компании;
* address – адрес компании;
* number – номер телефона компании;
* contact\_person – представитель компании.

1. Сущность продукт (product) имеет 4 атрибута:

* product\_id – первичный ключ продукта;
* product \_name – название продукта;
* product\_price – цена за единицу товара;
* product\_info – информация о товаре.

1. Сущность доставка (delivery) имеет 4 атрибута:

* delivery\_id – первичный ключ доставки;
* delivery\_type – тип доставки;
* delivery\_price – цена доставки;
* delivery\_time – время доставки

1. Сущность контракт (contract) имеет 3 атрибута:

* contract\_number – первичный ключ контракта;
* date\_of\_purchase – дата заключения контракта;
* client\_id\_client – id клиента (внешний ключ).

Сущности **client** и **contract** имеют связь один ко многим (обязательная), так как у клиента может быть много или один контракт.

Сущности **product и** **delivery** имеют связь многие ко многим (обязательная), так как у продукта может быть несколько видов доставки.

Сущности **product** и **contract** имеют связь многие ко многим, так как в контракте может быть несколько продуктов и продукты могут быть в нескольких контактах. В этой связи присутствует атрибут product\_quantity – количество товара в контракте.

Сущность **delivery** имеет связь один ко многим с **many\_product\_has\_many\_contract**, так как в контракте должен указываться тип доставки.

## **2.3. Создание физической модели базы данных**

### 2.3.1. Описание декларативных ограничений целостности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя таблицы | Ограничение | Атрибут | Назначение | SQL-code |
| client | PRIMARY KEY | client \_id | Атрибут должен быть уникальным | CONSTRAINT client\_pk PRIMARY KEY (client\_id) |
| client | NOT NULL | company\_name  contact\_person  address  number | Атрибуты не могут быть пустыми | company\_name text NOT NULL UNIQUE,  address text NOT NULL,  number numeric (11,0) NOT NULL UNIQUE,  contact\_person text NOT NULL UNIQUE, |
| product | PRIMARY KEY | product\_id | Атрибут должен быть уникальным | CONSTRAINT product\_pk PRIMARY KEY (product\_id) |
| product | NOT NULL | product\_name  product\_price  product\_info | Атрибуты не могут быть пустыми | product\_name text NOT NULL,  product\_price money NOT NULL,  product\_info text NOT NULL, |
| delivery | PRIMARY KEY | delivery\_id | Атрибут должен быть уникальным | CONSTRAINT delivery\_pk PRIMARY KEY (delivery\_id) |
| delivery | NOT NULL | delivery\_type  delivery\_price  delivery\_time | Атрибуты не могут быть пустыми | delivery\_type smallint NOT NULL,  delivery\_price money NOT NULL,  delivery\_time interval NOT NULL, |
| contract | PRIMARY KEY | contract\_number | Атрибут должен быть уникальным | CONSTRAINT contract\_pk PRIMARY KEY (contract\_number) |
| contract | NOT NULL | date\_of\_purchase | Атрибут не может быть пустыми | date\_of\_purchase date NOT NULL, |
| contract | FOREIGN KEY | client\_id\_client | Не может быть пустым. Связывает таблицы contract и client | ALTER TABLE public.contract ADD CONSTRAINT client\_fk FOREIGN KEY (client\_id\_client) |

### 2.3.2. Декларация таблиц БД

CREATE TABLE IF NOT EXISTS public.client (

client\_id smallint NOT NULL GENERATED ALWAYS AS

IDENTITY,

company\_name text NOT NULL UNIQUE,

address text NOT NULL,

number numeric (11,0) NOT NULL UNIQUE,

contact\_person text NOT NULL UNIQUE,

CONSTRAINT client\_pk PRIMARY KEY (client\_id)

);

ALTER TABLE public.client OWNER TO postgres;

CREATE TABLE IF NOT EXISTS public.product (

product\_id smallint NOT NULL GENERATED ALWAYS AS

IDENTITY,

product\_name text NOT NULL,

product\_price money NOT NULL,

product\_info text NOT NULL,

CONSTRAINT product\_pk PRIMARY KEY (product\_id)

);

ALTER TABLE public.product OWNER TO postgres;

CREATE TABLE IF NOT EXISTS public.delivery (

delivery\_id smallint NOT NULL GENERATED ALWAYS AS

IDENTITY,

delivery\_type smallint NOT NULL,

delivery\_price money NOT NULL,

delivery\_time interval NOT NULL,

CONSTRAINT delivery\_pk PRIMARY KEY (delivery\_id)

);

ALTER TABLE public. delivery OWNER TO postgres;

CREATE TABLE IF NOT EXISTS public.contract (

contract\_number smallint NOT NULL GENERATED ALWAYS

AS IDENTITY,

date\_of\_purchase date NOT NULL,

client\_id\_client smallint,

CONSTRAINT contract\_pk PRIMARY KEY (contract\_number),

CONSTRAINT client\_fk FOREIGN KEY (client\_id\_client)

REFERENCES public.client (client\_id) MATCH FULL

ON DELETE SET NULL ON UPDATE CASCADE

);

ALTER TABLE public.contract OWNER TO postgres;

CREATE TABLE IF NOT EXISTS

public.many\_product\_has\_many\_contract (

product\_id\_product smallint NOT NULL,

contract\_number\_contract smallint NOT NULL,

delivery\_id\_delivery smallint,

product\_quantity smallint NOT NULL,

CONSTRAINT many\_product\_has\_many\_contract\_pk PRIMARY

KEY (product\_id\_product,contract\_number\_contract),

CONSTRAINT product\_fk FOREIGN KEY (product\_id\_product)

REFERENCES public.product (product\_id) MATCH FULL

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT contract\_fk FOREIGN KEY

(contract\_number\_contract)

REFERENCES public.contract (contract\_number) MATCH FULL

ON DELETE CASCADE ON UPDATE CASCADE

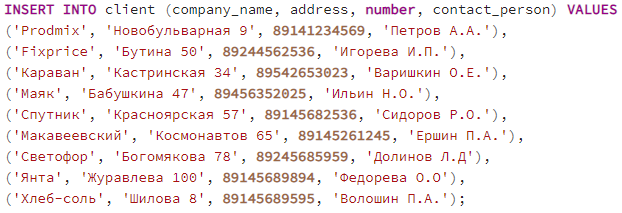
);

ALTER TABLE public.many\_product\_has\_many\_contract OWNER TO postgres;

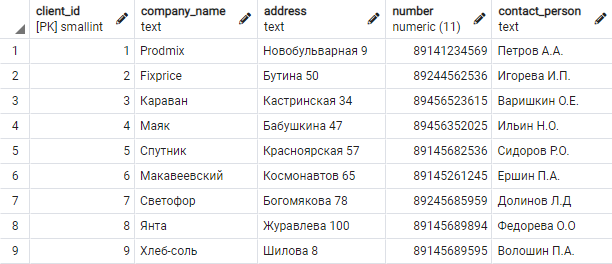
### 2.3.3. Операторы DML

**Оператор INSERT**

Для таблицы **client**

****

В результате в таблицу добавятся 9 клиентов.



Однако в случае нарушения ограничения целостности сгенерируется исключение. Например, если в таблицу попытаться добавить id клиента, то произойдёт нарушение:





Для этой же таблицы попытка добавить уже существующую компанию





**Оператор UPDATE**

Так как в каждой из представленных таблиц имеется первичный ключ, обновлять данные лучше всего именно по нему.

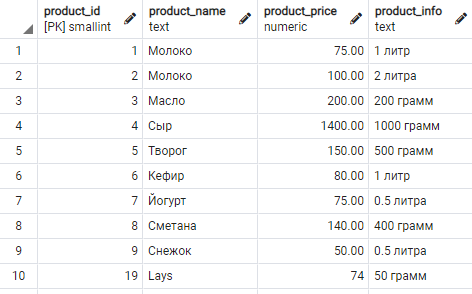
Например, в таблице **product** нужно изменить строку с id равным 19, а именно поменять название продукта.



Для этого необходимо прописать следующее:



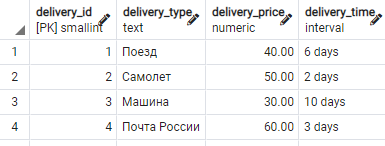
И как результат получить таблицу с изменёнными данными:



**Оператор DELETE**

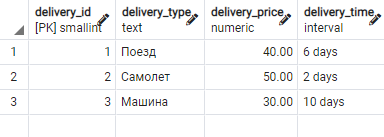
Для удаления также лучше использовать первичный ключ. Для примера возьмем таблицу **delivery**

До удаления:

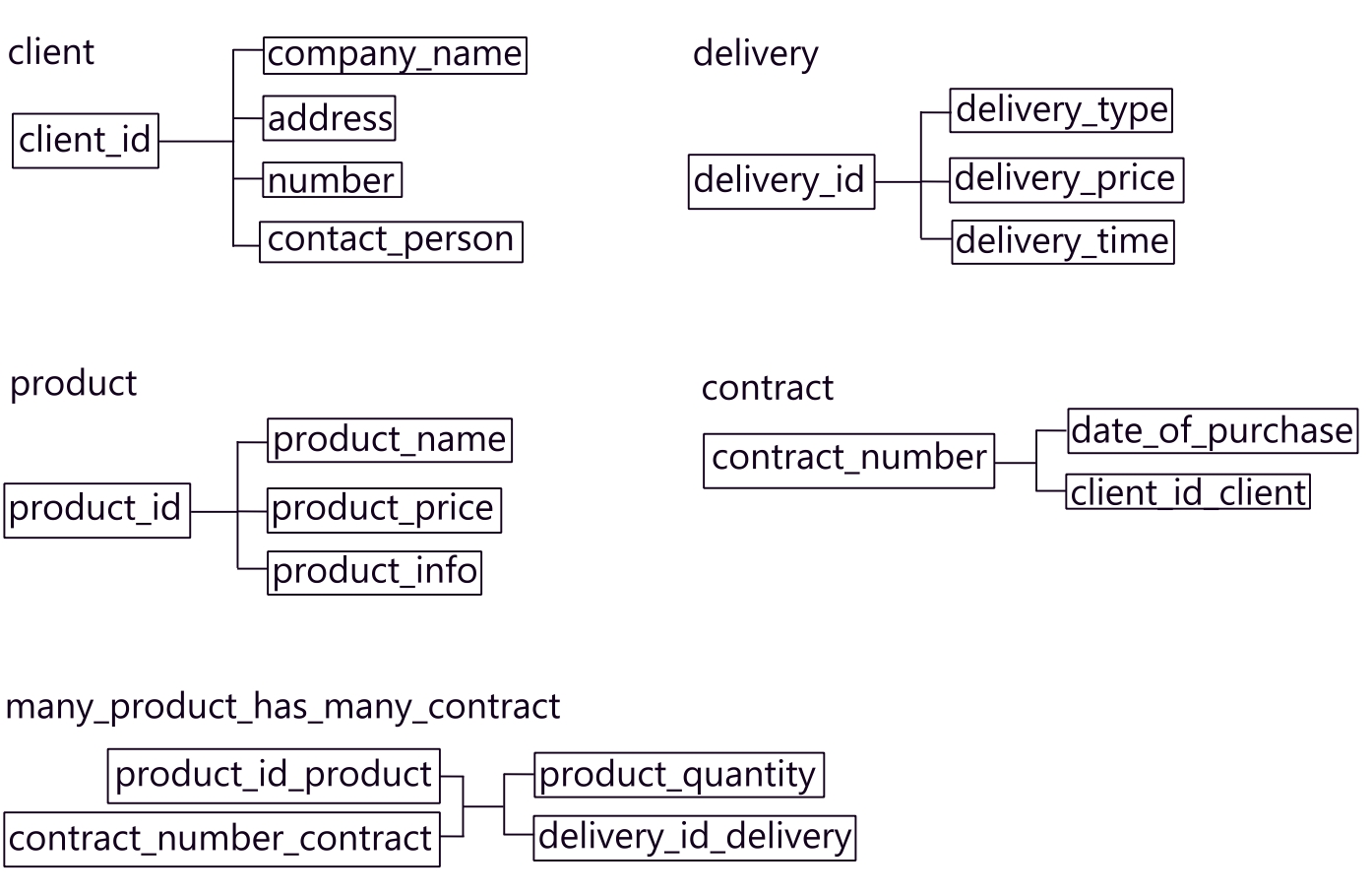


После:





## **2.4. Анализ таблиц на соответствие нормальным формам (3НФ)**



Все таблицы находятся в 1-й нормальной форме, т.к. атрибуты таблиц являются атомарными и не имеют повторяющихся значений.

Также таблицы находятся во 2-й нормальной форме, потому что неключевые атрибуты неприводимо зависят от первичного.

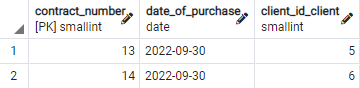
И также таблицы находятся в 3-й нормальной форме, т.к. неключевые атрибуты нетранзитивно зависят от первичного.

## **2.5. Описание запросов к БД. Оператор SELECT**

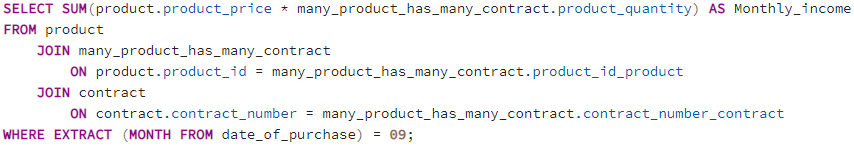
1) Список заказов за определенную дату:



Результат:



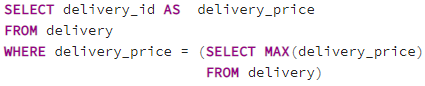
2) Прибыль компании за месяц:



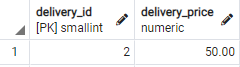
Результат:



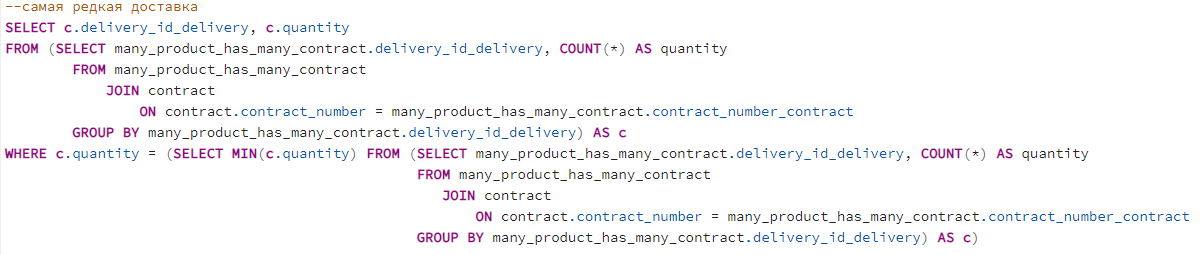
3) Самый прибыльный вид доставки:



Результат:



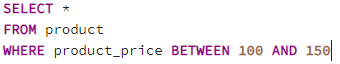
4) Самый редкий способ доставки для каждого заказчика:



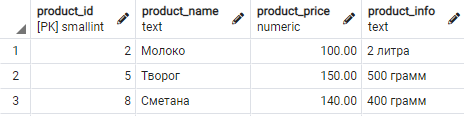
Результат:



5) Список товаров в заданном ценовом диапазоне:



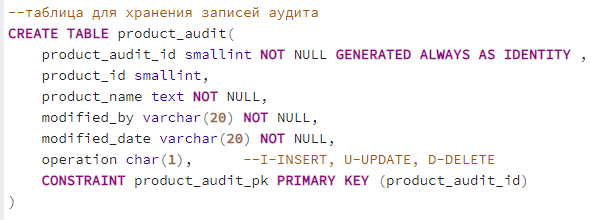
Результат:



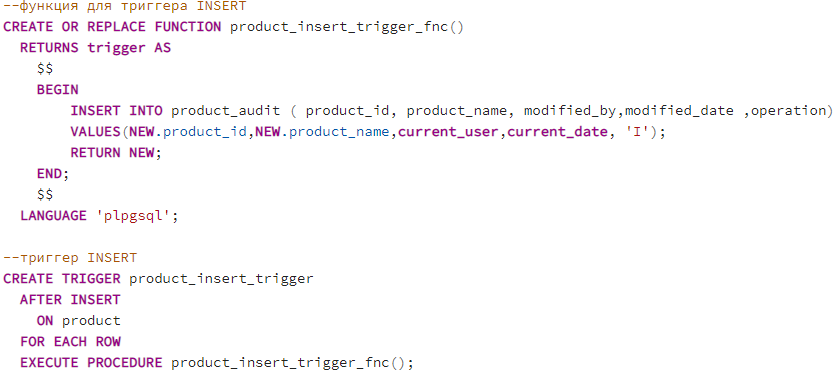
## 

## **2.6. Описание хранимых процедур и триггеров**

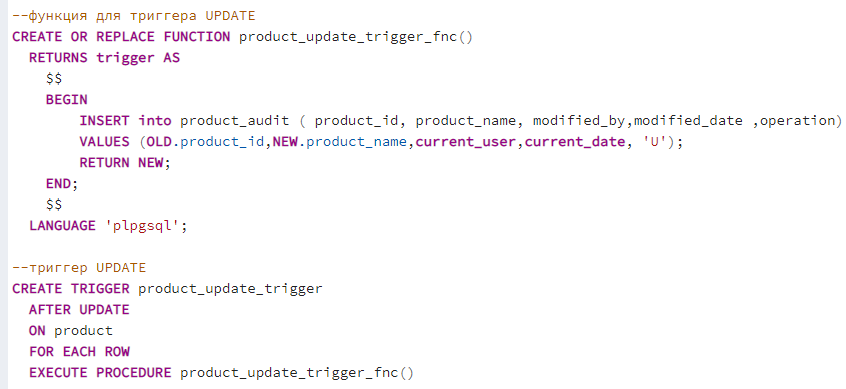
Триггеры, которые добавляют в таблицу аудита информацию об изменении таблицы **product**



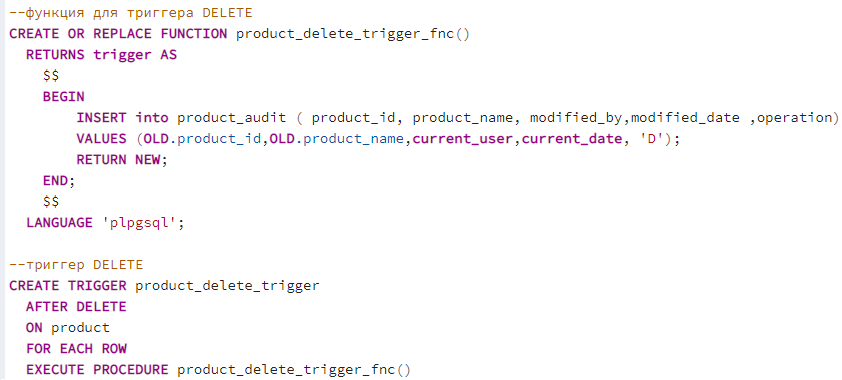
1)Триггер на вставку в таблицу **product**, который вставляет в таблицу аудита «I» при выполнении INSERT



2) Триггер на вставку в таблицу **product**, который вставляет в таблицу аудита «U» при выполнении UPDATE



3) Триггер на вставку в таблицу **product**, который вставляет в таблицу аудита «D» при выполнении DELETE



Результат работы триггеров



# **Заключение**

Данная работа была наполнена полезной теорией и обильным количеством практики, что позволило с легкостью освоить материал, а самое главное добиться цели работы – научиться моделировать, разрабатывать и работать с базой данных.

Для себя я выделила несколько особо важных навыков:

1. Умение создавать запросы, что, по моему мнению, стоит на первом месте по важности при работе с базами данных.
2. Вторым по важности я считаю умение смоделировать и нормализовать базу данных, так как, понимая ее архитектуру с нуля, можно сделать все что угодно, даже изначально не понимая, как это работает.

Эта работа заставила меня смотреть на проекты по-другому, так как раньше я задумывалась о том, как это работает и пыталась воссоздать или придумать алгоритм, а теперь я смотрю еще и на оборот данных в выбранной системе.

# **Ссылки на источники:**

1. «Вунивере.ру» Базы данных. Декларативная целостность данных [Электронный ресурс] / Режим доступа:   
   <https://vunivere.ru/work82463>
2. «Stud24» Программирование и компьютеры. Аномалии баз данных [Электронный ресурс] / Режим доступа:  
   <https://www.stud24.ru/programming-computer/anomalii-bd/384733-1260211-page1.html>
3. «Metanit» Нормализация. Первая нормальная форма [Электронный ресурс] / Режим доступа: <https://metanit.com/sql/tutorial/2.2.php#:~:text=Первая%20нормальная%20форма%20предполагает%2C%20что,значений%20для%20ключевого%20атрибута%20таблицы>
4. «Telegram» ZabGU\_IVT-20\_Db Лекции Гончарова Д.С.

5. Документация pgAdmin [Электронный ресурс] / Режим доступа:

https://www.pgadmin.org/docs/– Загл. с экрана.

6. Документация PostgreSQL [Электронный ресурс] / Режим доступа:

https://postgrespro.ru/docs/postgresql/ – Загл. с экрана.

7. Система управления базами данных [Электронный ресурс] / Режим

доступа: http://www.infosgs.narod.ru/31.htm - Загл. с экрана.