

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ  
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра управления инновациями (УИ)

## **БАЗА ДАННЫХ ДОМАШНЕЙ КОНДИТЕРСКОЙ**

Отчёт по курсовой работе  
по дисциплине «Базы данных»

Выполнил:

Студент гр. 022

\_\_\_\_\_ А.М. Зильберман

« \_\_\_\_ » \_\_\_\_\_ 2024 г.

Проверила:

Доцент кафедры УИ

\_\_\_\_\_ Ю.О. Лобода

оценка

« \_\_\_\_ » \_\_\_\_\_ 2024 г.

Томск 2024

## Оглавление

Введение.....	3
1 ОСНОВНАЯ ЧАСТЬ.....	5
1.1 Теоретическая часть.....	5
1.2 Проектирование работы.....	7
1.3 Техническая реализация.....	8
1.4 Результат работы.....	25
Заключение.....	26
Список использованных источников.....	27
Приложение А (обязательное) Код создания базы данных.....	28
Приложение Б (обязательное) Полный код с оформлением базы данных.....	29
Приложение В (обязательное) Код вызова таблиц, зависимости, изменения, удаления данных.....	33
Приложение Г (обязательное) Код вызова функции, представления, очистка и удаления таблицы (представления).....	34

## **Введение**

В любое время людям необходимо было сохранять свои знания, поскольку человек не мог запомнить всё. Для этого изобретались различные средства и способы, одним из них являются современные электронные базы данных. Целью данной курсовой работы является создание базы данных в специальном программном обеспечении - средстве управления базами данных (СУБД).

В соответствии с поставленной целью, база данных должна решать следующие задачи:

- 1) Предоставлять возможность внесения, редактирования или удаления данных из базы;
- 2) Сортировать данные во время их просмотра;
- 3) Обеспечивать функционал и закрывать все нужды организации, для которой создаётся база данных.

В курсовой работе представлена база данных домашней кондитерской. Она предоставляет пользователю следующие функции:

1. Просмотр каждой таблицы базы данных с сортировкой;
2. Создание новых таблиц;
3. Добавление в таблицы новых данных;
4. Редактирование определённых данных в таблице;
5. Удаление таблиц;
6. Удаление данных в таблице;
7. Просмотр количества элементов в таблице.

8. Вызов представления, объединяющего несколько таблиц, с вычислениями.

# **1 ОСНОВНАЯ ЧАСТЬ**

## **1.1 Теоретическая часть**

База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.

Базы данных значительно изменились с момента их появления в начале 1960-х годов. Исходными системами, которые использовались для хранения и обработки данных, были навигационные базы данных — например, иерархические базы данных (которые опирались на древовидную модель и допускали только отношение «один-ко-многим») и базы данных с сетевой структурой (более гибкая модель, допускающая множественные отношения).

Несмотря на простоту, эти ранние системы были негибкими. В 1980-х годах стали популярными реляционные базы данных, в 1990-х годах за ними последовали объектно-ориентированные базы данных. Совсем недавно вследствие роста Интернета и возникновения необходимости анализа неструктурированных данных появились базы данных NoSQL. В настоящее время облачные базы данных и автономные базы данных открывают новые возможности в отношении способов сбора, хранения, использования данных и управления ими.

Типы баз данных:

1. Реляционные базы данных. Они стали преобладать в 1980-х годах. Данные в реляционной базе организованы в виде таблиц, состоящих из столбцов и строк. Реляционная СУБД обеспечивает быстрый и эффективный доступ к структурированной информации.

2. Объектно-ориентированные базы данных. Информация в объектно-ориентированной базе данных представлена в форме объекта, как в объектно-ориентированном программировании.

3. Распределенные базы данных. Распределенная база данных состоит из двух или более частей, расположенных на разных серверах. Такая база данных может храниться на нескольких компьютерах.

4. Хранилища данных. Будучи централизованным репозиторием для данных, хранилище данных представляет собой тип базы данных, специально предназначенной для быстрого выполнения запросов и анализа.

5. Облачные базы данных. Облачная база данных представляет собой набор структурированных или неструктурированных данных, размещённый на частной, публичной или гибридной платформе облачных вычислений. Существует два типа моделей облачных баз данных: традиционная база данных и база данных как услуга (DBaaS). В модели DBaaS административные задачи и обслуживание выполняются поставщиком облачных услуг.

Подробнее о них можно прочитать в учебном пособии по базам данных, автор которого В.Д. Сибилёв [1], либо в книге о реляционных базах данных за авторством С.С. Куликова [2].

При выполнении курсовой работы, мы будем использовать СУБД PostgreSQL, который создаёт базы данных с помощью языка SQL. Язык SQL (Structured Query Language) предназначен для

определения ресурсов данных, манипулирования данными и управления доступом к данным на концептуальном уровне. SQL не содержит никаких процедурных средств.

Современный SQL является компромиссом между теоретическими положениями реляционной модели данных и практикой использования баз данных. Поэтому его конструкции лишь частично соответствуют требованиям РМД. Например, стандарты не поддерживают реляционные домены, допускают определение таблицы без первичного ключа и содержат ряд других отклонений от РМД. На самом деле SQL – это самостоятельная модель данных, ориентированная на практическое применение в компьютерных системах накопления и обработки информации.

Более подробно язык описан в книге об SQL Уолтера Шилдса, откуда и взята информация выше [3].

## **1.2 Проектирование работы**

На данном этапе формирования программы определены цель, задачи, необходимый функционал.

База данных будет содержать следующие таблицы с данными:

- Список товаров с названием, ценниками, типами продукции, временем приготовления;
- Типы продукции;
- Список работников и связь между ними;
- Список заказов, где указан товар, покупатель, работник и сумма;
- Список покупателей с их псевдонимами и номерами телефонов.

Ниже, на рисунке 1.1 представлена схема ERD по базе данных, разрабатываемой в ходе курсовой работы.

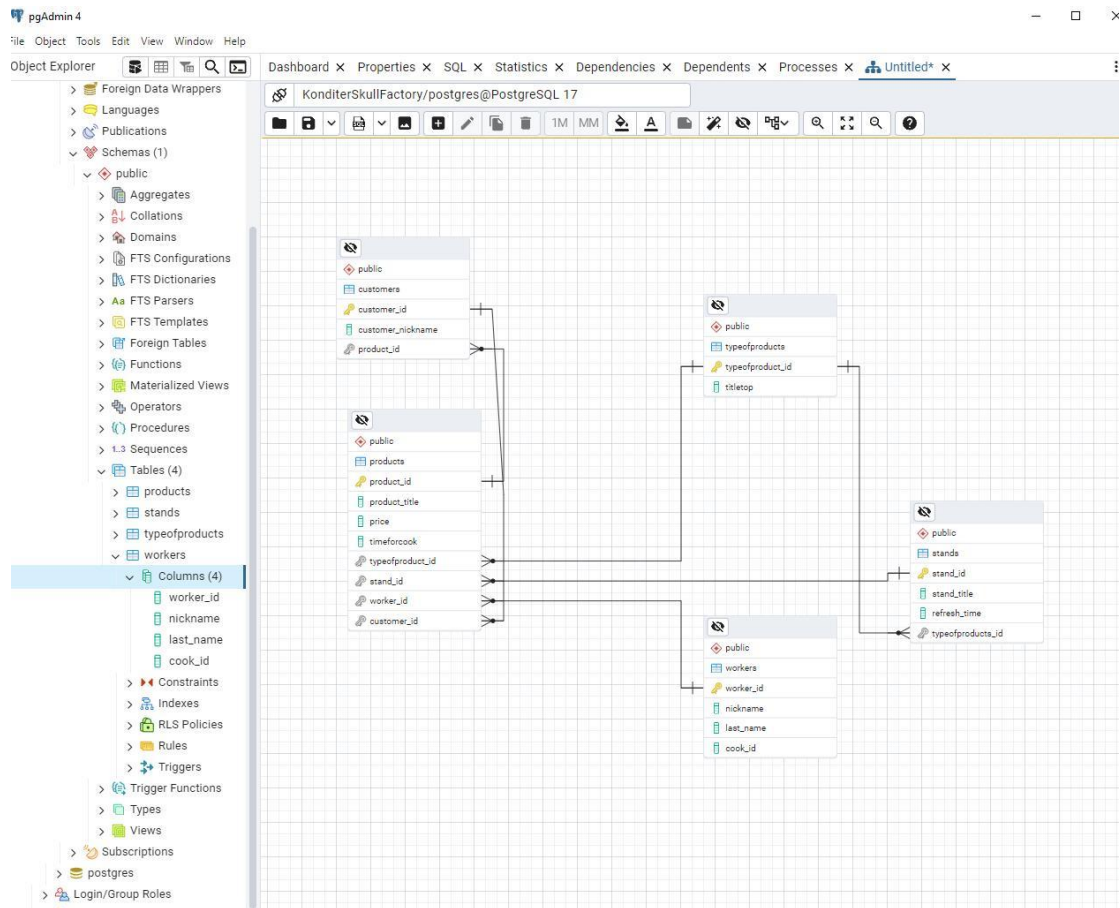


Рисунок 1.1 - Схема ERD

Для написания базы данных использовался язык SQL в СУБД PostgreSQL.

Все операции производились через Query Tool, то есть командную строку СУБД, а не через пользовательский интерфейс, в котором также можно создавать базы данных без знания кода.

## 1.3 Техническая реализация

### 1.3.1 Создание базы данных



Самая первая команда - создание самой базы данных. С помощью команды CREATE создаются новые элементы, DATABASE - тип элемента, конкретно база данных. Далее, KonditerSkullFactory11 - название базы данных, которое может быть любым.

```
CREATE DATABASE KonditerSkullFactory11
```

### **1.3.2 Создание таблицы с типами продукции**

Следующими шагами мы создаём таблицы в базе данных, которые будут содержать данные, необходимые для работы предприятия. CREATE TABLE создаёт таблицу, далее идёт её название - typeofproducts. Эта таблица содержит информацию о типах продукции в компании и идентификационный, уникальный ключ к каждому типу. Поэтому в скобках после CREATE TABLE мы задаём столбцы таблицы через запятую в следующем формате: название\_столбца тип\_данных. После этого идут параметры столбца. Идентификационный ключ есть в каждой таблице и обозначается параметром PRIMARY KEY, т.е. основной ключ, который не может повторяться в конкретной таблице, и данные, привязанные к этому ключу, закреплены за ним. Строка с PRIMARY KEY не может быть пустой. Параметр NOT NULL означает, что строка не может быть пустой. Типы данных integer означает целочисленные значения, а text - текстовое значение, оформленное в одинаковых кавычках.

INSERT INTO добавляет строки данных в таблицу и записывается в виде INSERT INTO название\_таблицы VALUES и далее в скобках через запятую записываются данные по

количеству столбцов в таблице. В данной таблице 2 столбца, поэтому каждая строка в скобках состоит из двух значений.

```
CREATE TABLE typeofproducts
(
    typeofproduct_id integer PRIMARY KEY,
    titletop text NOT NULL
);
```

```
INSERT INTO typeofproducts
VALUES
(1, 'Bakerys'),
(2, 'Drinks'),
(3, "Candys");
```

### **1.3.3 Создание таблицы с продукцией**

Создание таблицы описано выше и далее будет только дополняться новыми данными и объясняться назначение таблицы. Данная таблица содержит данные о конкретных продуктах, которые продаются компанией. Параметр REFERENCES привязывает столбец к столбцу из другой таблицы и записывается в виде: название\_таблицы(название\_столбца). Это помогает в сортировке и выводе информации далее.

```
CREATE TABLE products
(
    product_id integer PRIMARY KEY,
    product_title text NOT NULL,
```

```

        price integer NOT NULL,
        timeforcook integer NOT NULL,
        typeofproduct_id integer REFERENCES
typeofproducts(typeofproduct_id),
        stand_id integer REFERENCES stands(stand_id)
    );

```

```

INSERT INTO products
VALUES
(100, 'Puncake', 299, 30, 1, 1),
(101, 'Cupcake', 249, 40, 1, 1),
(102, "Homemade Sunflower", 400, 120, 1, 2),
(103, "Rainday", 59, 15, 1, 2),
(104, "Cookie", 19, 45, 1, 3),
(200, "Coca-Cola", 199, 0, 2, 4),
(201, "Sprite", 199, 0, 2, 4),
(202, "Orange Juice", 189, 0, 2, 5),
(300, "RotFront", 79, 0, 3, 6);

```

#### **1.3.4 Создание таблицы с стойками**

Таблица со стойками содержит данные с названиями стоек, где хранятся товары, а также с временем их обновления.

```

CREATE TABLE stands
(
    stand_id integer PRIMARY KEY,
    stand_title text,
    refresh_time integer,

```

```

        typeofproduct_id          integer          REFERENCES
typeofproducts(typeofproduct_id)
);

```

```

INSERT INTO stands
VALUES
(1, 'Bakerys_1', 120, 1),
(2, 'Bakerys_2', 120, 1),
(3, 'Bakerys_3', 120, 1),
(4, 'Freezer_1', 60, 2),
(5, 'Freezer_1', 60, 2),
(6, 'Candys_1', 240, 3);

```

### 1.3.5 Создание таблицы с работниками

Таблица с работниками ведёт учёт всех работников с их именами и прозвищами, а также их руководителей.

```

CREATE TABLE workers
(
    worker_id integer PRIMARY KEY,
    nickname text,
    last_name text,
    cook_id integer
);

```

```

INSERT INTO workers
VALUES
(1, 'Father', 'Luke', null),

```

```
(2, 'Son', 'Anakin', 1),  
(3, 'Son', 'Lavr', 1),  
(4, 'Padavan', 'Scam', 2),  
(5, 'Padavan', 'Srum', 2),  
(6, 'Padavan', 'Max', 3),  
(7, 'Creep', 'Dotl', 2);
```

### **1.3.6 Создание таблицы с покупателями**

Таблица с покупателями заполнена данными о покупателях с их прозвищами и номерами телефонов. Тип данных `varchar` содержит текстовое значение в скобках. `Varchar` можно ограничить, если написать рядом в скобках число, которое будет считаться максимальным значением символов.

```
CREATE TABLE customers  
(  
    customer_id integer PRIMARY KEY,  
    customer_nickname varchar,  
    phonenumber varchar(12)  
);
```

```
INSERT INTO customers  
VALUES  
(1, 'Batman', '89144889464'),  
(2, 'Robin', '89144999565'),  
(3, 'Bobin', '89249950102'),  
(4, 'Kassandra', '89569569565'),  
(5, 'Doono', 'unknown'),
```

(6, 'Emptyello', 'unknown');

### **1.3.7 Создание таблицы с заказами**

Таблица с заказами содержит номера заказов, покупателя, продукции и работника, а также количество продукции.

```
CREATE TABLE orders
(
    order_id integer PRIMARY KEY,
    customer_id integer REFERENCES customers(customer_id),
    product_id integer REFERENCES products(product_id),
    worker_id integer REFERENCES workers(worker_id),
    ordercount integer
);
```

```
INSERT INTO orders
VALUES
(1, 1, 100, 4, 2),
(2, 1, 200, 4, 1),
(3, 3, 103, 5, 4),
(4, 4, 300, 6, 1);
```

### **1.3.8 Создание представления с полными заказами**

В данном блоке создаётся представление (VIEW). Представление - это своеобразная таблица, которая объединяет столбцы других таблиц в себе, но добавлять новые столбцы в него нельзя. Через представление можно добавлять в таблицы новые

данные. Создаётся представление командой CREATE OR REPLACE VIEW название\_представление AS, где OR REPLACE означает, что при существующем представлении оно заменится на новое. Строки после AS описывают представление. SELECT добавляет в представление написанные столбцы из таблиц, которые написаны после FROM. Столбцом может быть результат вычисления нескольких столбцов, как произведение в примере, а название этого столбца задаётся после AS. Функция JOIN присоединяет написанную далее таблицу к таблице, которая написана после FROM ранее. Сопоставляются таблицы с помощью сравнения значений между друг другом. В данном примере, таблицы сравнивают одинаковые значения в столбцах с одинаковым названием, которые записаны после USING(название\_столбца).

ORDER BY сортирует таблицу представления по заданному столбцу. ASC - параметр сортировки, означает сортировку по возрастанию.

Данное представление предоставляет полную информацию о заказах.

```
CREATE OR REPLACE VIEW view_orders_orderers AS
SELECT  order_id,  workers.nickname,  workers.last_name,
product_title,
ordercount, customer_nickname, phonenumber,
products.price * ordercount AS totalprice
FROM orders
JOIN products USING(product_id)
JOIN workers USING(worker_id)
JOIN customers USING(customer_id)
ORDER BY order_id ASC;
```

### 1.3.9 Создание представления с хранением продукции

Данное представление показывает, где и какой товар, с указанием типа продукции, хранится из стендов.

```
CREATE OR REPLACE VIEW view_products_place AS
SELECT      products.product_id,      typeofproducts.titletop,
products.product_title, products.price, stands.stand_title
FROM products
JOIN typeofproducts USING(typeofproduct_id)
JOIN stands USING(stand_id)
ORDER BY product_id;
```

### 1.3.10 Функция, возвращающая цену введённой продукции

В этом разделе создаётся функция. Функция - команда, с которую может вызвать пользователь короткой строкой. Действие функции заранее прописывается. Данная функция принимает название продукции и возвращает цену на эту продукцию, записанную в таблице.

Команда CREATE OR REPLACE FUNCTION создаёт или пересоздаёт при существующей функцию в следующем формате:

```
CREATE      OR      REPLACE      FUNCTION
название_функции(название_переменной
тип_подаваемой_переменной)      RETURNS
тип_возвращаемой_переменной AS $$.
```

После \$\$ описывается действие функции. Команда WHERE ищет строку, которая удовлетворяет написанному условию. Закрывается действие



функции теми же \$\$ LANGUAGE SQL описывает, что функция работает на языке SQL.

```
CREATE OR REPLACE FUNCTION
get_product_price(OUTproduct_title text) RETURNS int AS $$
    SELECT price FROM products WHERE OUTproduct_title =
product_title
$$ LANGUAGE SQL;

SELECT get_product_price('Cupcake');
```

### **1.3.11 Вывод таблиц базы данных**

Данный блок выводит указанную таблицу на экран. \* означает вывод всей таблицы полностью. Название таблицы записывается после FROM.

```
SELECT * FROM products
ORDER BY product_id;
```

### **1.3.12 Вывод зависимости между работниками**

Этот блок описывает вывод внутреннего соединения, которое показывает работников и их начальников. Соединение производится как и ранее командой JOIN, а сопоставление происходит после ON.

```
SELECT a.nickname || ' ' || a.last_name AS cook,
       w.nickname || ' ' || w.last_name AS worker
FROM workers a
```

```
INNER JOIN workers w ON w.worker_id = a.cook_id  
ORDER BY cook;
```

### **1.3.13 Изменение данных в таблице**

Здесь описывается обновление строк в столбце таблицы. Команда UPDATE обновляет данные указанной таблицы. SET меняет значение указанного столбца на написанное значение. WHERE ищет заданное соответствие.

```
UPDATE workers  
SET last_name = 'Scam' WHERE worker_id = 4;
```

### **1.3.14 Удаление данных из таблицы**

Блок удаляет данные из таблицы. NULL означает, что строка пустая и никакой информации не несёт, кроме того, что она пустая..

```
UPDATE products  
SET stand_id = NULL WHERE product_id = 201;
```

### **1.3.15 Очистка и удаление таблицы**

TRUNCATE TABLE полностью очищает все данные в таблице без следов. DROP TABLE IF EXISTS удаляет полностью таблицу, если она существует, однако об этом остаётся след в логах. Также можно удалять VIEW, если заменить TABLE на него.

```
TRUNCATE TABLE orders;
```

DROP TABLE IF EXISTS orders;

### **1.3.16 Изменение столбцов в таблице**

Данный блок описывается изменения столбцов в таблице. RENAME меняет название указанного столбца на записанный после TO. ADD COLUMN добавляет в конец новый столбец.

ALTER TABLE products

RENAME price TO money;

ALTER TABLE products

ADD COLUMN product\_company text;

## **1.4 Результаты работы**

Результат кода создания базы данных из приложения А, рисунок 1.2.

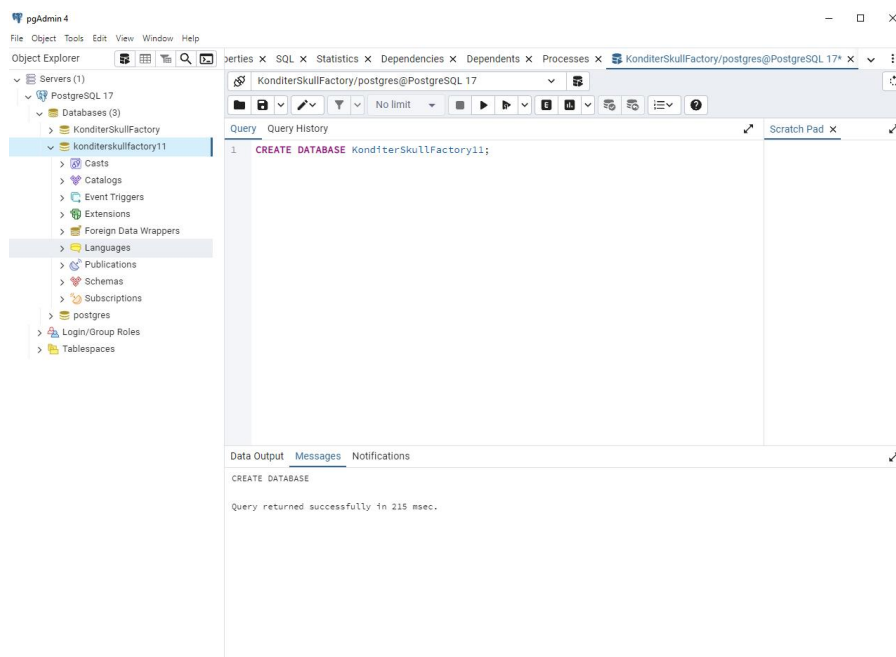


Рисунок 1.2 - Создание базы данных

Добавление всех таблиц, заполнение данными, создание представлений и функций после инициализации кода из приложения Б, рисунок 1.3.

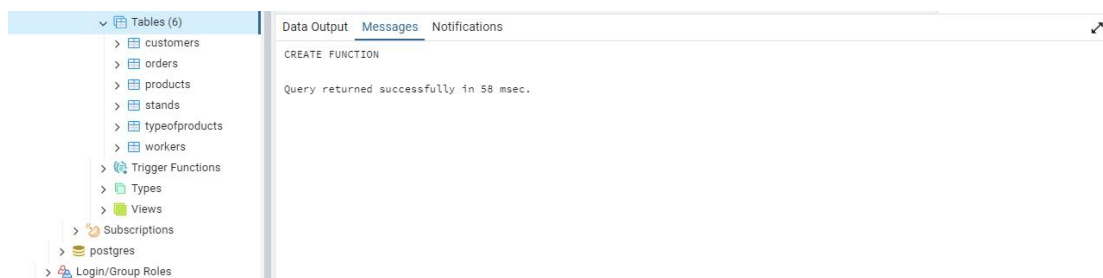
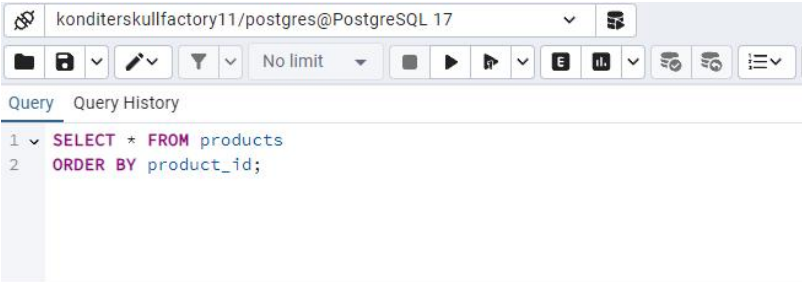


Рисунок 1.3 - Добавление данных

Вывод таблиц, код из приложения В, рисунок 1.4.



Query Query History

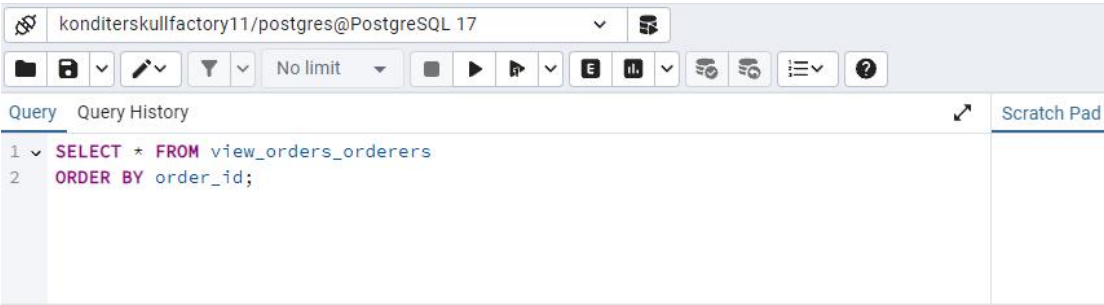
```
1 SELECT * FROM products
2 ORDER BY product_id;
```

Data Output Messages Notifications

	product_id [PK] integer	product_title text	price integer	timeforcook integer	typeofproduct_id integer	stand_id integer
1	100	Puncake	299	30	1	1
2	101	Cupcake	249	40	1	1
3	102	Homemade Sunflower	400	120	1	2
4	103	Rainday	59	15	1	2
5	104	Cookie	19	45	1	3
6	200	Coca-Cola	199	0	2	4
7	201	Sprite	199	0	2	4
8	202	Orange Juice	189	0	2	5
9	300	RotFront	79	0	3	6

Рисунок 1.4 - Вывод таблицы

Вывод представления функции, приложение Г, рисунок 1.5.



Query Query History

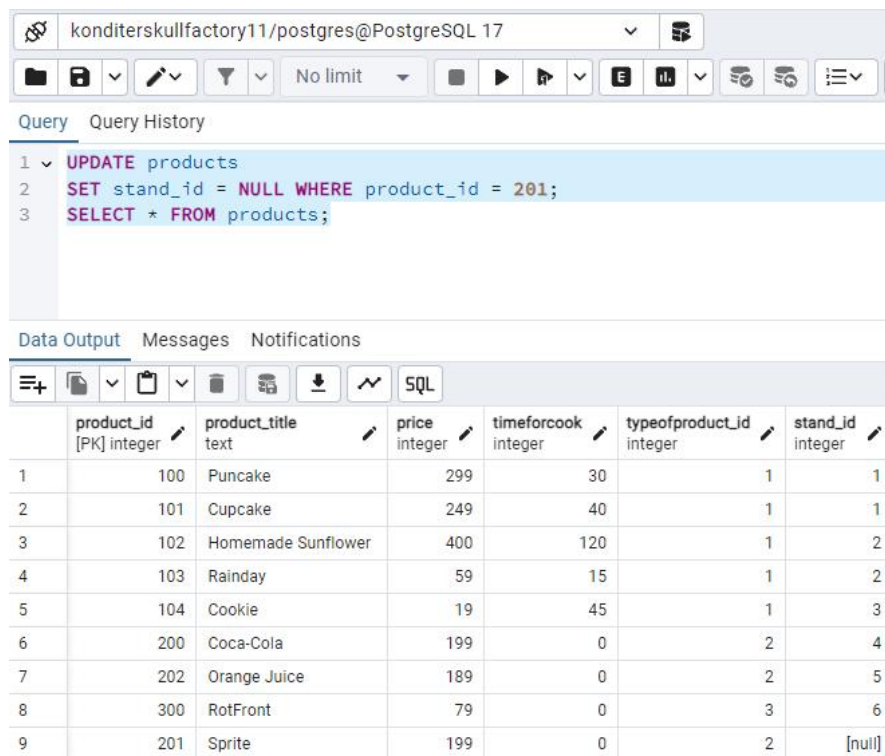
```
1 SELECT * FROM view_orders_orderers
2 ORDER BY order_id;
```

Data Output Messages Notifications

	order_id integer	nickname text	last_name text	product_title text	ordercount integer	customer_nickname character varying	phonenummer character varying (12)	totalprice integer
1	1	Padavan	Scam	Puncake	2	Batman	89144889464	598
2	2	Padavan	Scam	Coca-Cola	1	Batman	89144889464	199
3	3	Padavan	Srum	Rainday	4	Bobin	89249950102	236
4	4	Padavan	Max	RotFront	1	Kassandra	89569569565	79

Рисунок 1.5 - Вывод представления

Удаление и изменение данных, описанные в приложении В, рисунки 1.6 и 1.7.



Query Query History

```

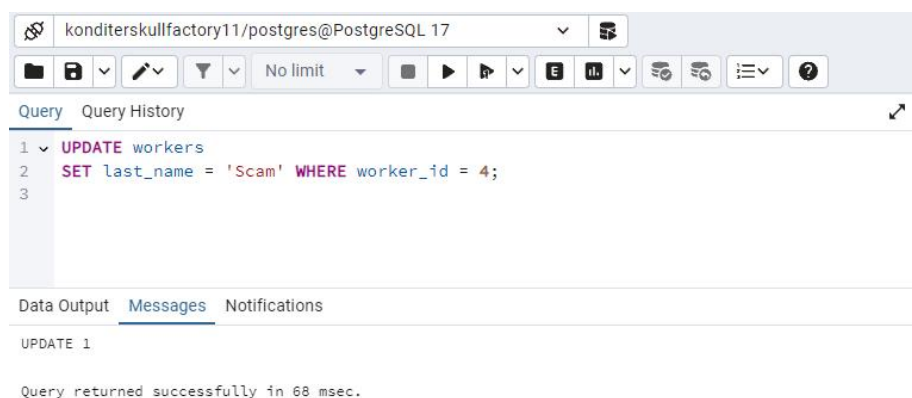
1 UPDATE products
2 SET stand_id = NULL WHERE product_id = 201;
3 SELECT * FROM products;

```

Data Output Messages Notifications

	product_id [PK] integer	product_title text	price integer	timeforcook integer	typeofproduct_id integer	stand_id integer
1	100	Puncake	299	30	1	1
2	101	Cupcake	249	40	1	1
3	102	Homemade Sunflower	400	120	1	2
4	103	Rainday	59	15	1	2
5	104	Cookie	19	45	1	3
6	200	Coca-Cola	199	0	2	4
7	202	Orange Juice	189	0	2	5
8	300	RotFront	79	0	3	6
9	201	Sprite	199	0	2	[null]

Рисунок 1.6 - Удаление данных



Query Query History

```

1 UPDATE workers
2 SET last_name = 'Scam' WHERE worker_id = 4;
3

```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 68 msec.

Рисунок 1.7 - Изменение данных

Изменение столбца описано в приложении Г, рисунок 1.8.

The screenshot shows a PostgreSQL query editor with the following SQL commands:

```

1 ALTER TABLE products
2   RENAME price TO money;
3
4 ALTER TABLE products
5   ADD COLUMN product_company text;
6
7 SELECT * FROM products;

```

The 'Data Output' tab displays the following table:

	product_id [PK] integer	product_title text	money integer	timeforcook integer	typeofproduct_id integer	stand_id integer	product_company text
1	100	Puncake	299	30	1	1	[null]
2	101	Cupcake	249	40	1	1	[null]
3	102	Homemade Sunflower	400	120	1	2	[null]
4	103	Rainday	59	15	1	2	[null]
5	104	Cookie	19	45	1	3	[null]
6	200	Coca-Cola	199	0	2	4	[null]
7	202	Orange Juice	189	0	2	5	[null]
8	300	RotFront	79	0	3	6	[null]
9	201	Sprite	199	0	2	[null]	[null]

Рисунок 1.8 - Изменение столбца

Вывод таблицы с зависимостью работников к руководителям описано в приложении В, рисунок 1.9.

The screenshot shows a PostgreSQL query editor with the following SQL query:

```

1 SELECT a.nickname || ' ' || a.last_name AS cook,
2        w.nickname || ' ' || w.last_name AS worker
3 FROM workers a
4 INNER JOIN workers w ON w.worker_id = a.cook_id
5 ORDER BY cook;

```

The 'Data Output' tab displays the following table:

	cook text	worker text
1	Creep Dotl	Son Anakin
2	Padavan Max	Son Lavr
3	Padavan Scam	Son Anakin
4	Padavan Srum	Son Anakin
5	Son Anakin	Father Luke
6	Son Lavr	Father Luke

Рисунок 1.9 - Вывод зависимости работников

Вызов функции, удаление представления (таблицы), описаны в приложении Г, результат на рисунках 1.10 и 1.11.

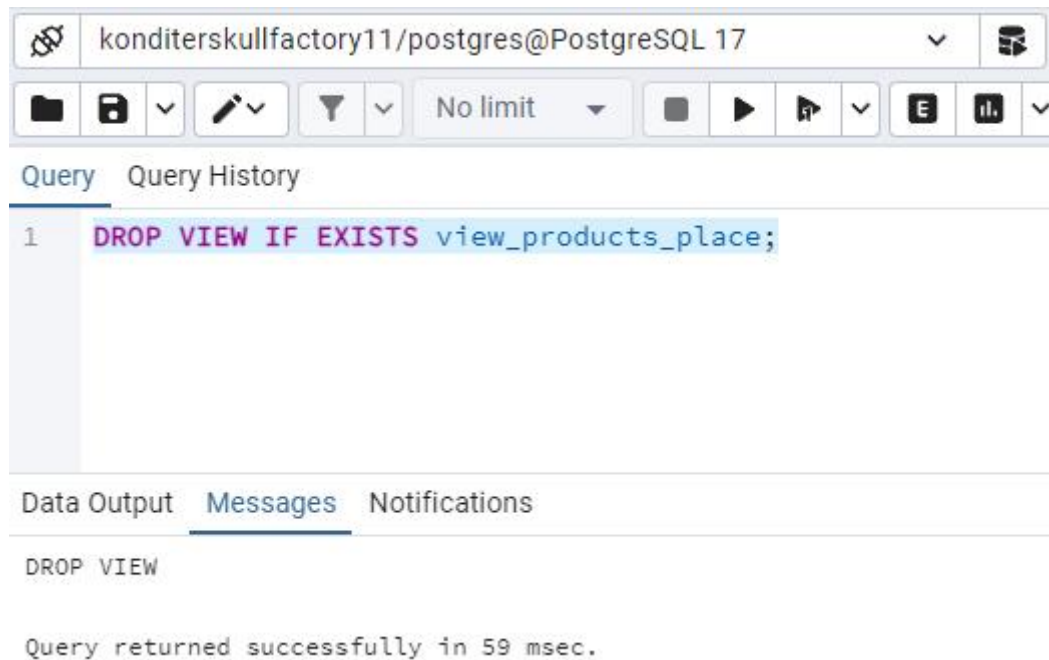


Рисунок 1.10 - Удаление представления (таблицы)

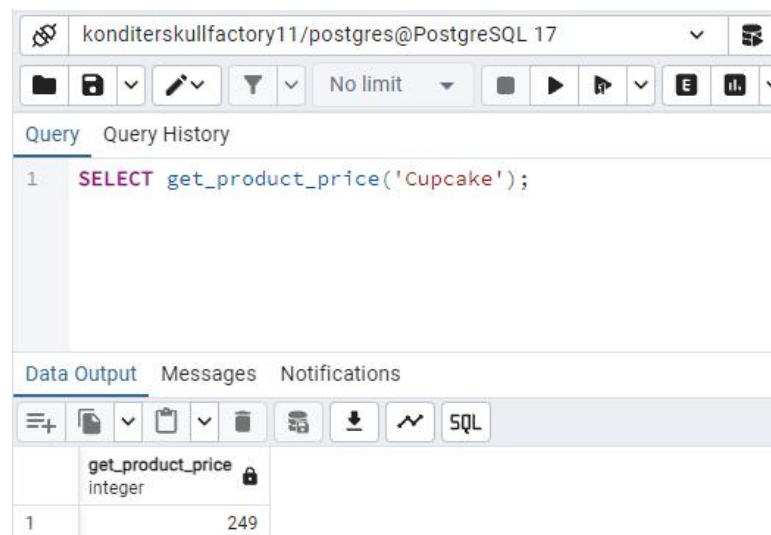


Рисунок 1.11 - Вызов функции



## **Заключение**

В ходе курсовой работы были расширены и закреплены теоретические знания по SQL, а также были получены практические навыки путём решения задачи создания базы данных домашней кондитерской. На основе СУБД PostgreSQL мы написали базу данных, способную сохранять, сортировать введённые данные, а также обновляться по мере необходимости. Настроены представления для удобного чтения базы данных. Таким образом, цель курсовой работы была достигнута, посредством решения поставленных задач.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Сибилёв В.Д. Базы данных. Часть 1 Основы систем баз данных: учебное пособие. Томск: ФДО, ТУСУР, 2021. 260 с.

[2] Куликов С.С. Реляционные базы данных в примерах. Практическое пособие для программистов и тестировщиков. М.: Четыре четверти, 2020. 426 с.

[3] Шилдс У. SQL: быстрое погружение. СПб.: Питер, 2022. 224 с.

[4] Моргунов Е.П. PostgreSQL. Основы языка SQL: учеб. Пособие. СПб.: БХВ-Петербург, 2018. 336 с.

[5] Оптимизация запросов в PostgreSQL / Г. Домбровская [и др.]. М.: ДМК Пресс, 2022. 278 с.

**Приложение А (обязательное) Код создания базы данных**

```
CREATE DATABASE KonditerSkullFactory11;
```

**Приложение Б (обязательное) Полный код с оформлением  
базы данных**

```
CREATE TABLE typeofproducts
(
    typeofproduct_id integer PRIMARY KEY,
    titletop text NOT NULL
);
```

```
INSERT INTO typeofproducts
VALUES
(1, 'Bakerys'),
(2, 'Drinks'),
(3, 'Candys');
```

```
CREATE TABLE stands
(
    stand_id integer PRIMARY KEY,
    stand_title text,
    refresh_time integer,
    typeofproduct_id integer REFERENCES
typeofproducts(typeofproduct_id)
);
```

```
INSERT INTO stands
VALUES
(1, 'Bakerys_1', 120, 1),
(2, 'Bakerys_2', 120, 1),
(3, 'Bakerys_3', 120, 1),
```

```
(4, 'Freezer_1', 60, 2),
(5, 'Freezer_1', 60, 2),
(6, 'Candys_1', 240, 3);
```

```
CREATE TABLE products
(
    product_id integer PRIMARY KEY,
    product_title text NOT NULL,
    price integer NOT NULL,
    timeforcook integer NOT NULL,
    typeofproduct_id integer REFERENCES
typeofproducts(typeofproduct_id),
    stand_id integer REFERENCES stands(stand_id)
);
```

```
INSERT INTO products
VALUES
(100, 'Puncake', 299, 30, 1, 1),
(101, 'Cupcake', 249, 40, 1, 1),
(102, 'Homemade Sunflower', 400, 120, 1, 2),
(103, 'Rainday', 59, 15, 1, 2),
(104, 'Cookie', 19, 45, 1, 3),
(200, 'Coca-Cola', 199, 0, 2, 4),
(201, 'Sprite', 199, 0, 2, 4),
(202, 'Orange Juice', 189, 0, 2, 5),
(300, 'RotFront', 79, 0, 3, 6);
```

```
CREATE TABLE workers
(
```

```
worker_id integer PRIMARY KEY,  
nickname text,  
last_name text,  
cook_id integer  
);
```

```
INSERT INTO workers  
VALUES  
(1, 'Father', 'Luke', null),  
(2, 'Son', 'Anakin', 1),  
(3, 'Son', 'Lavr', 1),  
(4, 'Padavan', 'Scam', 2),  
(5, 'Padavan', 'Srum', 2),  
(6, 'Padavan', 'Max', 3),  
(7, 'Creep', 'Dotl', 2);
```

```
CREATE TABLE customers  
(  
customer_id integer PRIMARY KEY,  
customer_nickname varchar,  
phonenummer varchar(12)  
);
```

```
INSERT INTO customers  
VALUES  
(1, 'Batman', '89144889464'),  
(2, 'Robin', '89144999565'),  
(3, 'Bobin', '89249950102'),  
(4, 'Kassandra', '89569569565'),
```

```
(5, 'Doono', 'unknown'),  
(6, 'Emptyello', 'unknown');
```

```
CREATE TABLE orders  
(  
    order_id integer PRIMARY KEY,  
    customer_id integer REFERENCES customers(customer_id),  
    product_id integer REFERENCES products(product_id),  
    worker_id integer REFERENCES workers(worker_id),  
    ordercount integer  
);
```

```
INSERT INTO orders  
VALUES  
(1, 1, 100, 4, 2),  
(2, 1, 200, 4, 1),  
(3, 3, 103, 5, 4),  
(4, 4, 300, 6, 1);
```

```
CREATE OR REPLACE VIEW view_orders_orderers AS  
SELECT    order_id,    workers.nickname,    workers.last_name,  
product_title,  
ordercount, customer_nickname, phonenumber,  
products.price * ordercount AS totalprice  
FROM orders  
JOIN products USING(product_id)  
JOIN workers USING(worker_id)  
JOIN customers USING(customer_id)  
ORDER BY order_id ASC;
```

```

CREATE OR REPLACE VIEW view_products_place AS
SELECT      products.product_id,      typeofproducts.titletop,
products.product_title, products.price, stands.stand_title
FROM products
JOIN typeofproducts USING(typeofproduct_id)
JOIN stands USING(stand_id)
ORDER BY product_id;

```

```

CREATE      OR      REPLACE      FUNCTION
get_product_price(OUTproduct_title text) RETURNS int AS $$
    SELECT price FROM products WHERE OUTproduct_title =
product_title
    $$ LANGUAGE SQL;

```



**Приложение В (обязательное) Код вызова таблиц,  
зависимости, изменения, удаления данных**

```
SELECT * FROM products;
```

```
SELECT a.nickname || ' ' || a.last_name AS cook,  
       w.nickname || ' ' || w.last_name AS worker  
FROM workers a  
INNER JOIN workers w ON w.worker_id = a.cook_id  
ORDER BY cook;
```

```
ALTER TABLE products  
RENAME price TO money;
```

```
ALTER TABLE products  
ADD COLUMN product_company text;
```

```
UPDATE workers  
SET last_name = 'Scam' WHERE worker_id = 4;
```

```
UPDATE products  
SET stand_id = NULL WHERE product_id = 201;
```

**Приложение Г (обязательное) Код вызова функции,  
представления, очистка и удаления таблицы (представления)**

```
SELECT get_product_price('Cupcake');
```

```
SELECT * FROM view_orders_orderers;
```

```
TRUNCATE TABLE orders;
```

```
DROP TABLE IF EXISTS orders;
```

```
DROP VIEW IF EXISTS view_products_place;
```