Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное  
учреждение высшего образования  
«РУТ (МИИТ)»

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

КУРСОВОЙ ПРОЕКТ

по дисциплине

**«Основы построения защищенных баз данных»**

**на тему  
«Складской учет»**

Выполнил: ст. гр. ТКИ-441

Порхун Д.Д.

Проверил: доц., к.т.н.

Васильева М. А.

Москва 2024

Оглавление

[ЦЕЛЬ КУРСОВОГО ПРОЕКТА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5](#_Toc136201526)

[ЗАДАНИЕ НА КУРСОВОВОЙ ПРОЕКТ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 5](#_Toc136201527)

[ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ\_\_\_\_\_ 6](#_Toc136201528)

[1. Инфологическое проектирование 6](#_Toc136201529)

[1.1 Анализ предметной области 6](#_Toc136201530)

[1.2 Анализ информационных задач и круга пользователей системы 6](#_Toc136201531)

[2. Логическое проектирование реляционной БД 7](#_Toc136201532)

[2.1 Составление реляционных отношений 7](#_Toc136201533)

[3. Физическое проектирование БД 11](#_Toc136201534)

[3.1 Разработка скриптов на создание базы данных и таблиц 11](#_Toc136201535)

[3.2 Разработка скриптов на добавление данных в таблицы 13](#_Toc136201536)

[3.3 Разработка необходимых запросов: 14](#_Toc136201537)

[3.4 Разработка необходимых запросов, представлений (view), процедур, функций и триггеров: 17](#_Toc136201538)

[Список литературы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20](#_Toc136201539)

# ЦЕЛЬ КУРСОВОГО ПРОЕКТА

Целью курсового проекта является изучение методов и закрепление знаний в проектировании реляционных баз данных (РБД) в системе управления базами данных (СУБД) Система управления базами данных PostgreSQL.

# ЗАДАНИЕ НА КУРСОВОВОЙ ПРОЕКТ

В данном курсовом проекте ставится задача разработать РБД в СУБД PostgreSQL по заданной теме «Складской учет». Проектирование РБД проводится с помощью метода «Сущность-связь». Проверка построенной модели РБД осуществляется с помощью метода нормализации отношений.

**Описание предметной области.**

БД создаётся для информационного обслуживания склада. Некоторая фирма имеет склад товаров. Эти товары фирма получает от производителя и расфасовывает их для магазинов. Также данная фирма имеет возможность оптового отпуска ассортимента товаров. Для постоянных клиентов предусмотрена система скидок.

**Готовые запросы:**

* Выдавать ассортимент товара, находящегося на складе сейчас.
* Выдавать ассортимент товара, заказанного данным магазинов.
* Показывать список продаж за указанный период времени.
* Показывать список клиентов, имеющих скидку.

# ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

## Инфологическое проектирование

### Анализ предметной области

База данных создаётся для информационного обслуживания склада.

Выделим базовые сущности этой предметной области:

* Клиент
* Скидка
* Дата
* Название
* Количество
* Магазин

ER-диаграмма БД, приведена на (Рисунке 1). На данной ER-диаграмме присутствуют связь многие - ко – многим, поэтому вводится дополнительная промежуточная сущность.

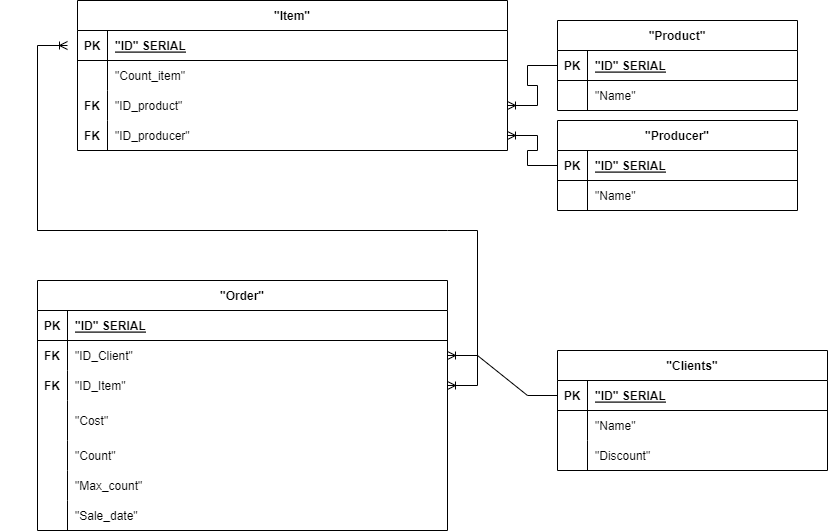


Рисунок 1 – ER-диаграмма

### Анализ информационных задач и круга пользователей системы

Определим границы информационной поддержки пользователей:

**1) Функциональные возможности:**

* ведение БД (запись, чтение, модификация, удаление);
* обеспечение логической непротиворечивости БД;
* реализация наиболее часто встречающихся запросов в готовом виде.

**2) Готовые запросы:**

* Выдавать ассортимент товара, находящегося на складе сейчас.
* Выдавать ассортимент товара, заказанного данным магазинов.
* Показывать список продаж за указанный период времени.
* Показывать список клиентов, имеющих скидку.

## Логическое проектирование реляционной БД

### Составление реляционных отношений

Каждое реляционное отношение соответствует одной сущности (объекту ПрО), и в него вносятся все атрибуты сущности. Для каждого отношения необходимо определить первичный ключ и внешние ключи (если они есть). В том случае, если базовое отношение не имеет потенциальных ключей, вводится *суррогатный первичный ключ*, который не несёт смысловой нагрузки и служит только для идентификации записей.

Были созданы следующие сущности:

1. Таблица "Order" - содержит информацию о заказе. Содержит идентификатор, идентификатор клиента, идентификатор товара на складе, цену, количество заказанных товаров, максимальное кол-во, и дату заказа.
2. Таблица "Сlients" - содержит информацию о клиентах, включая идентификатор, имя и скидку, которая может быть применена к покупкам клиента.
3. Таблица "item" - содержит информацию о продуктах которые есть на складе. Включает идентификатор, количество на складе продукта, идентификатор товара и идентификатор производителя.
4. Таблица " Producer" - содержит информацию о производителе, включая идентификатор и название производителя.
5. Таблица "Product" - содержит информацию о товаре, включая идентификатор и название товара.

**Таблица 1 – Схема отношения** **Заказы** (Orders)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Содержание поля** | **Имя поля** | **Тип** | **Длина** | **Примечания** |
| Первичный ключ | ID | Целое число | - | первичный ключ |
| идентификатор клиента | ID\_Client | Строка | 255 | обязательное поле |
| идентификатор товара на складе | Vendor\_Code | Целое число | - | обязательное поле |
| Цена товара | Cost | Число с плавающей запятой | 10, 2 | обязательное поле |
| Количество товара в заказе | Count | Целое число | - | обязательное поле |
| Дата продажи | Sale\_date | date | - | обязательное поле |

Таблица 2 – Схема отношения Клиенты (Clients)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Содержание поля** | **Имя поля** | **Тип** | **Длина** | **Примечания** |
| Первичный ключ | id | Целое число | - | первичный ключ |
| Наименование клиента | name | Строка | 255 | обязательное поле |
| Скидка клиента | discount | Число с плавающей запятой | 4, 2 | значение по умолчанию - 0 |

Таблица 3 – Схема отношения Магазины (Items)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Содержание поля** | **Имя поля** | **Тип** | **Длина** | **Примечания** |
| Первичный ключ | id | Целое число | - | первичный ключ |
| Количество на складе | Count\_item | Целое число | - | обязательное поле |
| идентификатор продукта | ID\_product | Строка | 255 | обязательное поле |
| идентификатор производителя | ID\_producer | Строка | 255 | обязательное поле |

Таблица 4 – Схема отношения Производители (producers)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Содержание поля** | **Имя поля** | **Тип** | **Длина** | **Примечания** |
| Первичный ключ | id | Целое число | - | первичный ключ |
| Наименование магазина | name | Строка | 255 | обязательное поле |

Таблица 5 – Схема отношения Товары (products)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Содержание поля** | **Имя поля** | **Тип** | **Длина** | **Примечания** |
| Первичный ключ | id | Целое число | - | первичный ключ |
| Наименование магазина | name | Строка | 255 | обязательное поле |

Нормализация полученных отношений

**1НФ.** Для приведения таблиц к 1НФ требуется составить таблицы (один атрибут – один столбец) и разбить сложные атрибуты на простые, а многозначные атрибуты вынести в отдельные отношения.  
В наших таблицах все атрибуты простые.

**2НФ.** Все отношения находятся во 2НФ, так как для каждого из них введен уникальный ключ.

**3НФ**. Все неключевые атрибуты зависят только от первичного ключа и не зависят друг от друга. Поэтому спроектированная **база данных находится в третьей нормальной форме**.

Созданную схему Вы можете показана на рисунке 2.

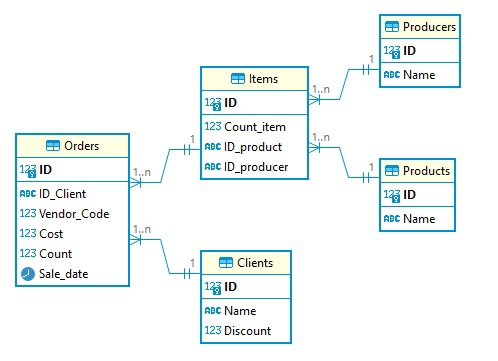


Рисунок 2 – ER-диаграмма в DBeaver

## Физическое проектирование БД

### Разработка скриптов на создание базы данных и таблиц

БД «Складской учет» проектируется в среде Система управления базами данных PostgresSQL.

*Создание базы данных* «Складской учет»:

Создание таблицы Products:

-- таблица "Продукт"

**create** **table** "Products"(

"ID" **INT** **SERIAL** **PRIMARY** **KEY**

,"Name" **Varchar**(255) **NOT** **null**

);

Создание органичений для таблицы Products:

-- Делаем ячейку Name уникальной

**ALTER** **TABLE** "Products"

**ADD** **CONSTRAINT** UQ\_Products\_Name **UNIQUE** ("Name");

Создание таблицы Producers:

-- таблица "Производитель"

**create** **table** "Producers"(

"ID" **INT** **SERIAL** **PRIMARY** **KEY**

,"Name" **Varchar**(255) **NOT** **null**

);

Создание органичений для таблицы Producers:

-- Делаем ячейку Name уникальной

**ALTER** **TABLE** "Producers"

**ADD** **CONSTRAINT** UQ\_Producers\_Name **UNIQUE** ("Name");

Создание таблицы Clients:

-- таблица "Клиент"

**create** **table** "Clients" (

"ID" **INT** **SERIAL** **PRIMARY** **KEY**

, "Name" **VARCHAR**(255) **NOT** **null**

, "Discount" **DECIMAL**(4, 2) **DEFAULT** 0

);

Создание органичений для таблицы Clients:

-- Делаем ячейку Name уникальной

**ALTER** **TABLE** "Clients"

**ADD** **CONSTRAINT** UQ\_Clients\_Name **UNIQUE** ("Name");

Создание таблицы Items:

-- таблица "Склад"

**create** **table** "Items" (

"ID" **INT** **SERIAL** **PRIMARY** **KEY**

,"Count\_item" **INT** **NOT** **NULL**

,"ID\_product" **VARCHAR**(255) **NOT** **NULL**

,"ID\_producer" **VARCHAR**(255) **NOT** **NULL**

);

Создание органичений для таблицы Items:

-- Создание условия что кол-во единиц на складе должно быть более или равен 0

**ALTER** **TABLE** "Items"

**ADD** **CONSTRAINT** CH\_Count\_item

**CHECK** ("Count\_item" >= 0);

-- Создание внешнего ключа между двумя таблицами

**ALTER** **TABLE** "Items"

**ADD** **CONSTRAINT** FK\_Items\_Products

**FOREIGN** **KEY** ("ID\_product")

**REFERENCES** "Products"("Name");

-- Создание внешнего ключа между двумя таблицами

**ALTER** **TABLE** "Items"

**ADD** **CONSTRAINT** FK\_Items\_Producers

**FOREIGN** **KEY** ("ID\_producer")

**REFERENCES** "Producers"("Name");

Создание таблицы Orders:

-- таблица "Заказ"

**create** **table** "Orders" (

"ID" **INT** **SERIAL** **PRIMARY** **KEY**

,"ID\_Client" **VARCHAR**(255) **NOT** **NULL**

,"Vendor\_Code" **INT** **NOT** **NULL**

,"Cost" **DECIMAL**(10, 2) **NOT** **NULL**

,"Count" **INT** **NOT** **null**

,"Sale\_date" **DATE** **NOT** **NULL**

);

Создание органичений для таблицы Orders:

-- Создание внешнего ключа между двумя таблицами

**ALTER** **TABLE** "Orders"

**ADD** **CONSTRAINT** FK\_Orders\_Item\_ID

**FOREIGN** **KEY** ("Vendor\_Code")

**REFERENCES** "Items"("ID");

-- Создание внешнего ключа между двумя таблицами

**ALTER** **TABLE** "Orders"

**ADD** **CONSTRAINT** FK\_Orders\_Clients

**FOREIGN** **KEY** ("ID\_Client")

**REFERENCES** "Clients"("Name");

--Создание условия что кол-во в заказе не должно быть меньше 0

**ALTER** **TABLE** "Orders"

**ADD** **CONSTRAINT** CH\_Orders\_Count

**CHECK** ("Count" >= 0);

### 3.2 Разработка скриптов на добавление данных в таблицы

-- Добавить клиентов

**INSERT** **INTO** **"Clients"** (**"Name"**, **"Discount"**) **VALUES** (**'Клиент 1'**, 0);

**INSERT** **INTO** **"Clients"** (**"Name"**, **"Discount"**) **VALUES** (**'Клиент 2'**, 5);

**INSERT** **INTO** **"Clients"** (**"Name"**, **"Discount"**) **VALUES** (**'Клиент 3'**, 10);

-- Добавить товары

**INSERT** **INTO** **"Product"** (**"Name"**) **VALUES** (**'Товар 1'**);

**INSERT** **INTO** **"Product"** (**"Name"**) **VALUES** (**'Товар 2'**);

**INSERT** **INTO** **"Product"** (**"Name"**) **VALUES** (**'Товар 3'**);

-- Добавить производителей

**INSERT** **INTO** **"Producer"** (**"Name"**) **VALUES** (**'Производитель 1'**);

**INSERT** **INTO** **"Producer"** (**"Name"**) **VALUES** (**'Производитель 2'**);

**INSERT** **INTO** **"Producer"** (**"Name"**) **VALUES** (**'Производитель 3'**);

-- Добавить товары

**INSERT** **INTO** **"Items"** (**"Count\_item"**, **"ID\_product"**, **"ID\_producer"**) **VALUES** (1, **'Товар 2'**, **'Производитель 1'**);

**INSERT** **INTO** **"Items"** (**"Count\_item"**, **"ID\_product"**, **"ID\_producer"**) **VALUES** (2, **'Товар 1'**, **'Производитель 1'**);

**INSERT** **INTO** **"Items"** (**"Count\_item"**, **"ID\_product"**, **"ID\_producer"**) **VALUES** (3, **'Товар 3'**, **'Производитель 2'**);

**INSERT** **INTO** **"Items"** (**"Count\_item"**, **"ID\_product"**, **"ID\_producer"**) **VALUES** (4, **'Товар 3'**, **'Производитель 2'**);

**INSERT** **INTO** **"Items"** (**"Count\_item"**, **"ID\_product"**, **"ID\_producer"**) **VALUES** (15, **'Товар 2'**, **'Производитель 2'**);

**INSERT** **INTO** **"Items"** (**"Count\_item"**, **"ID\_product"**, **"ID\_producer"**) **VALUES** (0, **'Товар 3'**, **'Производитель 3'**);

--Добавить заказ

**INSERT** **INTO** **"Orders"** (**"ID\_Client"**, **"Vendor\_Code"**, **"Cost"**, **"Count"**, **"Sale\_date"**) **values** (**'Клиент 1'**,3,250,1,**'2023-05-03'**);

**INSERT** **INTO** **"Orders"** (**"ID\_Client"**, **"Vendor\_Code"**, **"Cost"**, **"Count"**, **"Sale\_date"**) **values** (**'Клиент 1'**,4,250,1,**'2023-05-02'**);

**INSERT** **INTO** **"Orders"** (**"ID\_Client"**, **"Vendor\_Code"**, **"Cost"**, **"Count"**, **"Sale\_date"**) **values** (**'Клиент 1'**,2,250,2,**'2023-05-03'**);

**INSERT** **INTO** **"Orders"** (**"ID\_Client"**, **"Vendor\_Code"**, **"Cost"**, **"Count"**, **"Sale\_date"**) **values** (**'Клиент 2'**,5,250,2,**'2023-05-03'**);

### Разработка необходимых запросов:

1. Выдать ассортимент товара, находящегося на складе сейчас:

**CREATE** **OR** **REPLACE** **FUNCTION** GetAvailableItems(min\_quantity **INT**)

**RETURNS** **SETOF** **"Items"** **AS** **$$**

**BEGIN**

**RETURN** QUERY

**SELECT** \*

**FROM** **"Items"**

**WHERE** **"Count\_item"** > min\_quantity;

**END**;

**$$** **LANGUAGE** plpgsql;

-- Вызов функции

**select** GetAvailableItems(0);

Результат:

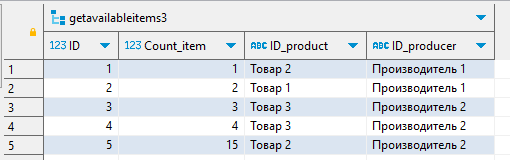


Рисунок 3 – Результат 1 запроса

1. Выдать ассортимент товара, заказанного данным магазином:

**CREATE** **OR** **REPLACE** **FUNCTION** GetItemsOrdersByStore3(ID\_product **INT**)

**RETURNS** **TABLE** (**"ID"** **INT**, **"Name"** **varchar**(255), **"Cost"** **numeric**, **"quantity"** **INT**) **AS** **$$**

**BEGIN**

**RETURN** QUERY

**SELECT** p.**"ID"**, p.**"Name"**, o.**"Cost"**, i.**"Count\_item"** **as** quantity

**FROM** **"Products"** p

**JOIN** **"Items"** i **ON** p.**"Name"** = i.**"ID\_product"**

**JOIN** **"Orders"** o **ON** i.**"ID"** = o.**"Vendor\_Code"**

**WHERE** o.**"Vendor\_Code"** = ID\_product;

**END**;

**$$** **LANGUAGE** plpgsql;

-- Вызов функции

**select** \* **from** GetItemsOrdersByStore3(3);

1. Результат:



Рисунок 4 – Результат 2 запроса

1. Показывать список продаж за указанный период времени:

**CREATE** **OR** **REPLACE** **FUNCTION** GetSalesBetweenDates(StartDate **date**, EndDate **date**)

**RETURNS** **TABLE** (order\_id **INT**, client\_name **varchar**(255), product\_name **varchar**(255), quantity **INT**, sale\_date **date**) **AS** **$$**

**BEGIN**

**RETURN** QUERY

**SELECT** o.**"ID"** **as** order\_id,

o.**"ID\_Client"** **as** client\_name,

p.**"Name"** **as** product\_name,

i.**"Count\_item"** **as** quantity,

o.**"Sale\_date"** **as** sale\_date

**FROM** **"Orders"** o

**JOIN** **"Items"** i **ON** o.**"Vendor\_Code"** = i.**"ID"**

**JOIN** **"Products"** p **ON** i.**"ID\_product"** = p.**"Name"**

**WHERE** o.**"Sale\_date"** **BETWEEN** StartDate **AND** EndDate;

**END**;

**$$** **LANGUAGE** plpgsql;

-- Вызов функции

**select** \* **from** GetSalesBetweenDates(**'2023-05-03'**,**'2023-05-03'**);

Результат:

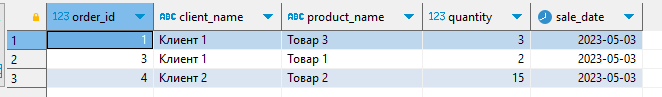


Рисунок 5 – Результат 3 запроса

1. Показывать список клиентов, имеющих скидку:

**CREATE** **OR** **REPLACE** **FUNCTION** GetListClientsWithDiscount(minDiscount **INT**)

**RETURNS** **SETOF** **"Clients"** **AS** **$$**

**BEGIN**

**RETURN** QUERY

**SELECT** \*

**FROM** **"Clients"**

**WHERE** **"Discount"** > minDiscount;

**END**;

**$$** **LANGUAGE** plpgsql;

**select** GetListClientsWithDiscount(0);

Результат:

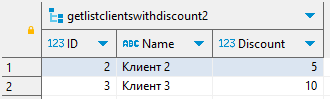


Рисунок 6 – Результат 4 запроса

### Разработка представлений (view), процедур, функций и триггеров:

Создание представления view:

**CREATE** **VIEW** "product\_sales" **as**

**SELECT**

o."ID" **as** order\_id,

c."Name" **as** client\_name,

p."Name" **as** product\_name,

i."Count\_item" **as** quantity,

o."Sale\_date" **as** sale\_date

**FROM**

"Order" o

**JOIN**

"Clients" c **ON** o."ID\_Client" = c."ID"

**JOIN**

"Item" i **ON** o."ID\_Item" = i."ID"

**JOIN**

"Product" p **ON** i."ID\_product" = p."ID"

Вызов представления view:

**SELECT** \* **FROM** "product\_sales"

Результат:

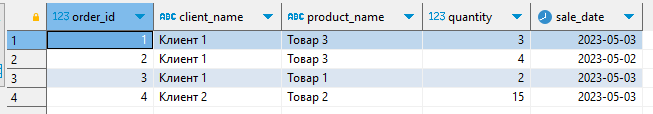


Рисунок 7 – Вывод view

Пример процедуры:

-- Процедура "Сумма покупок клиентов"

**CREATE** **PROCEDURE** get\_total\_sales()

**AS** **$$**

**BEGIN**

**PERFORM** c.**"Name"** **AS** client\_name, **SUM**(o.**"Cost"** \* o.**"Count"**) **AS** total\_sales

**FROM** **"Order"** o

**JOIN** **"Clients"** c **ON** o.**"ID\_Client"** = c.**"ID"**

**GROUP** **BY** c.**"Name"**;

**END**;

**$$** **LANGUAGE** plpgsql;

Вызов процедуры:

**CALL** get\_total\_sales();

*Пример функции:*

-- Функция "Сумма заказанных товаров клиента"

**CREATE** **OR** **REPLACE** **FUNCTION** get\_ordered\_quantity(client\_id **VARCHAR**(255))

**RETURNS** **INT** **AS** **$$**

**DECLARE**

total\_quantity **INT**;

**BEGIN**

**SELECT** **SUM**(o.**"Count"**) **INTO** total\_quantity

**FROM** **"Orders"** o

**WHERE** o.**"ID\_Client"** = get\_ordered\_quantity.client\_id; -- явное указание параметра функции

**RETURN** total\_quantity;

**END**;

**$$** **LANGUAGE** plpgsql;

Вызов функции:

**SELECT** get\_ordered\_quantity(**'Клиент 1'**);

Результат:



Рисунок 8 – Вывод процедуры

Создание триггера:

При удалении записей из таблицы "Order", будет вызываться и выдавать уведомление с информацией о удаленной строке.

**CREATE** **OR** **REPLACE** **FUNCTION** on\_delete\_orders()

**RETURNS** **TRIGGER** **AS** **$$**

**BEGIN**

**RAISE** **NOTICE** **'Строка с "Order\_ID" % была удалена из таблицы products.'**, **OLD**.**"ID"**;

**RETURN** **OLD**;

**END**;

**$$** **LANGUAGE** plpgsql;

**CREATE** **TRIGGER** delete\_orders\_trigger

**AFTER** **DELETE** **ON** **"Orders"**

**FOR** **EACH** **ROW**

**EXECUTE** **FUNCTION** on\_delete\_orders();

Проверка триггера:

**DELETE** **FROM** **"Orders"** **WHERE** **"ID"** = 3;

Результат:

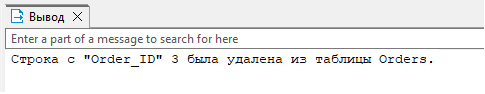


Рисунок 9 – Вывод триггера

# Список литературы

* + - 1. Васильева М. А, Филипченко К. М. Фильтрация набора данных. Рекомендации по выполнению работы и перечень типовых заданий: Учебно-методическое пособие. Издание третье, исправленное и дополненное. Москва–2024, 133 с.
      2. Васильева М. А., Клепцов М. Я. Группировка и обобщение данных. Рекомендации по выполнению работы и перечень типовых заданий. Учебно-методическое пособие. 2-е издание. 2024.– 59с.
      3. Васильева М. А., Балакина Е. П., Васильева М. В. Соединение данных из множества таблиц. Рекомендации по выполнению работы и перечень типовых заданий. Учебно–методическое пособие. Издание второе, исправленное. – М.: РУТ (МИИТ).2024. – 72с.
      4. Балакина Е.П., Васильева М.А., Филипченко К.М. Информационное обеспечение систем управления. Методические указания к курсовому проектированию. Учебно-методическое пособие. Издание второе, исправленное и дополненное, 2023.102–с.