

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
 «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

Лабораторная работа №5

Выполнил: обучающийся гр. ВКБ43

Ковалев Данил Петрович

Проверил: Доцент Скляров А.В.

« » 2025 г.

г. Ростов-на-Дону

2025

**Задание на самостоятельную работу**

**Задание 1.** Реализовать в таблице статические ограничения целостности (NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY).

Создаём таблицы Группы товаров и Товары с нужными столбцами и определите статические ограничения целостности, такие как NOT NULL, UNIQUE, PRIMARY KEY, и FOREIGN KEY.

Группы товаров представлены на рисунке 1 ниже.

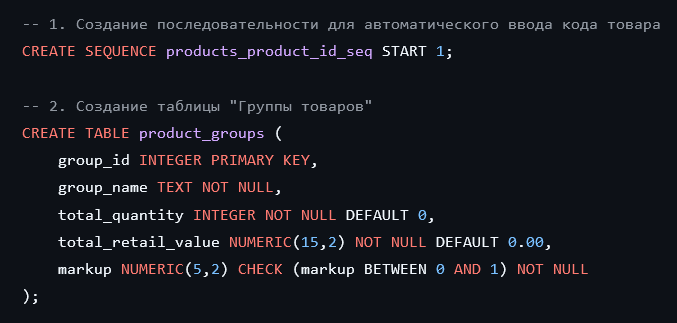


Рисунок 1 – таблица, которая описывает группы товаров

В этой таблице каждый товар связан с группой товаров. Столбец markup ограничен значениями от 0 до 1. (Markup — это поле, которое представляет наценку на товар, то есть процент, на который увеличивается цена товара при продаже относительно его закупочной цены.).

Sequence можно не создавать вручную, но автором лабораторной работы было принято такое решение ради разнообразия. PostgreSQL сам вполне умеет распоряжаться Sequence автоматически.

Создадим таблицу товаров, код представлен на рисунке 2.

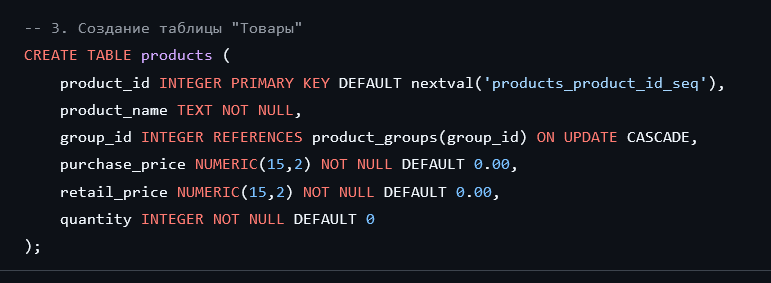


Рисунок 2 – таблица “Товары”

Столбец group\_code связан с таблицей group\_products через внешний ключ, что помогает поддерживать целостность данных. При удалении группы, все связанные товары также удалятся (опция ON DELETE CASCADE). Не нужно отдельно создавать последовательность для product\_code. Этот идентификатор будет автоматически увеличиваться при добавлении новых товаров.

**Задание 2.** Реализовать триггеры автоматической проверки и расчета динамических ограничений целостности для следующей задачи.

Создадим триггер, который автоматически будет рассчитывать розничную цену на основе приходной цены и наценки группы. Для этого добавим функцию и триггер. Функция update\_retail\_price вычисляет розничную цену и сохраняет ее в retail\_price. Триггер retail\_price\_trigger срабатывает перед вставкой или обновлением записи, чтобы цена пересчитывалась автоматически. Функция и триггер представлены на рисунке 3.

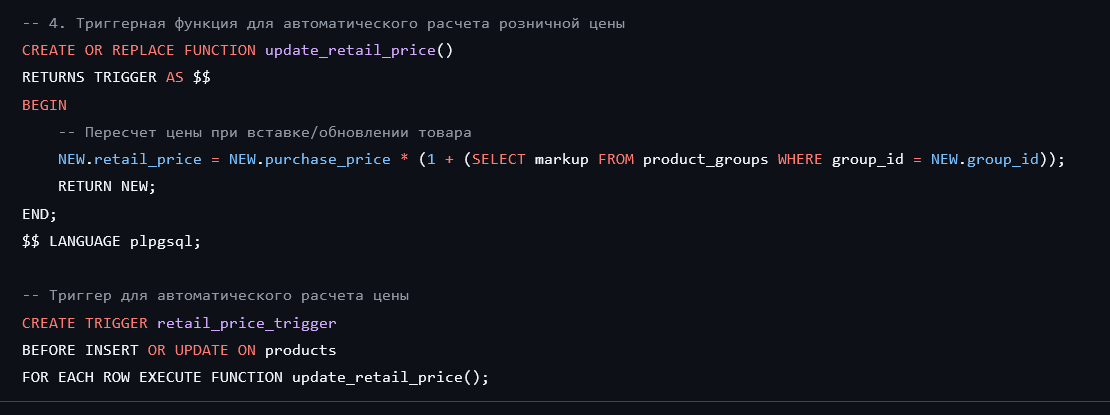


Рисунок 3 – триггер и функция для автоматического расчета розничной цены

Теперь напишем триггер для обновления "Количество на складе" и "Сводной стоимости" для группы. Данный триггер представлен на рисунке 4.

Для начала рассмотрим функцию update\_group\_stats(). Она умеет взаимодействовать с 3 операциями – DELETE, INSERT, UPDATE.

Если товар был удален, то мы пересчитываем значения total\_quantity и total\_retail\_value, уменьшая их.

Если же товар был обновлен, то проверяем, что ID группы совпадают или нет. Если же совпадает, то делаем изменения над старой группой, в ином случае над новой.

Если же товар был добавлен, то операции обратные от DELETE, меняются просто операторы.

Говоря коротко, функция update\_group\_totals обновляет общее количество и стоимость группы при добавлении или изменении товара.



Рисунок 4 – триггер для обновления данных групп

Теперь напишем триггер для пересчета розничной цены при изменении наценки в группе. Для этих целей создадим триггер для автоматического обновления розничной цены товаров, если в группе меняется наценка. Функция и триггер представлены на рисунке 5.

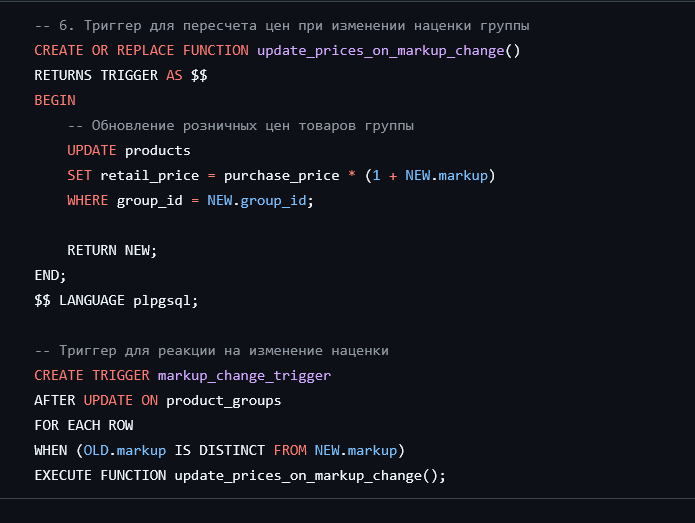


Рисунок 5 – триггер для пересчета цена при изменении наценки группы

Триггер markup\_change\_trigger срабатывает при изменении наценки в таблице group\_products и обновляет розничные цены товаров в этой группе. В таблице “Товары” значения retail\_price и stock\_quantity по умолчанию установлены в 0, чтобы избежать NULL значений, если данные не заполнены при добавлении записи.

Теперь перейдем к проверке работы функций и триггеров. Для проверки работы триггеров и функций выполним вставку тестовых данных и проследите за автоматическими изменениями. Для взаимодействия с базой данных я буду использовать Pycharm Professional.

Пример вставки данных:

INSERT INTO product\_groups (

group\_id,

group\_name,

markup

) VALUES (

2,

'Телевизоры',

0.2

);

INSERT INTO products (

product\_name,

group\_id,

purchase\_price,

quantity

) VALUES (

'Монитор Philips',

1,

10000,

5

);

После этого проверим, что retail\_price рассчитана правильно.

Убедимся, что при изменении количества товаров в группе обновляется сводная стоимость и количество на складе. Результат представлен на рисунке 6.

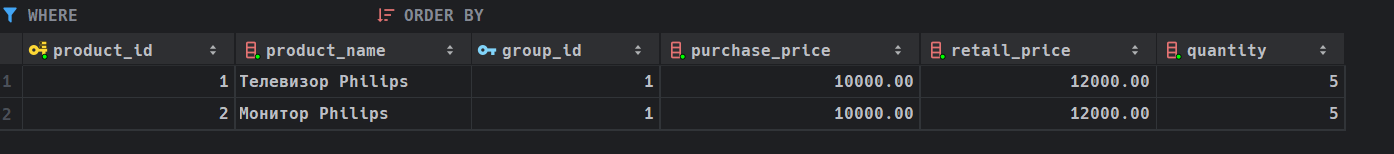


Рисунок 6 – результат расчета retail\_price

**Контрольные вопросы:**

**1. Статические ограничения целостности**

Статические ограничения целостности — это правила, обеспечивающие правильность и допустимость данных в базе данных. К основным ограничениям относятся:

NOT NULL — запрещает полю содержать NULL значения. Поле должно всегда быть заполнено.

UNIQUE — обеспечивает уникальность значений в столбце, что не допускает дублирование.

PRIMARY KEY — уникальный идентификатор для каждой строки в таблице, совмещает свойства NOT NULL и UNIQUE.

FOREIGN KEY — внешний ключ, который создает связь между таблицами. Он гарантирует, что значения в поле внешнего ключа должны существовать в связанной таблице.

**2. PL/pgSQL**

PL/pgSQL — процедурный язык для PostgreSQL, который добавляет возможность написания логики на сервере в виде функций и триггеров. Основные возможности PL/pgSQL:

Используется для создания функций, процедур и триггеров.

Добавляет управляющие конструкции (условия, циклы) в SQL.

Может выполнять сложные вычисления и автоматизировать задачи в базе данных.

**3. Последовательности**

Последовательность (sequence) — объект в PostgreSQL, который автоматически увеличивает значение. Используется для автоматической генерации уникальных идентификаторов, таких как product\_code. В данной работе последовательности создаются автоматически при использовании типа SERIAL.

**4. Триггеры**

Триггер — это объект базы данных, который запускает выполнение заданной логики при определенных событиях (вставка, обновление или удаление данных). В этой лабораторной работе триггеры выполняют автоматические вычисления и обновления:

BEFORE INSERT/UPDATE — триггер, который срабатывает перед вставкой или обновлением строки.

AFTER INSERT/UPDATE — триггер, который выполняется после завершения операции вставки или обновления.

Использование триггеров позволяет автоматически обновлять значения в полях, например, пересчитывать розничную цену или сводную стоимость товаров в группе.

**5. Функции (Functions)**

В PL/pgSQL функции — это программные блоки, которые содержат набор инструкций и могут быть вызваны в SQL-запросах или триггерах. В этой лабораторной работе функции выполняют расчеты:

Функция для расчета розничной цены: вычисляет значение поля retail\_price на основе приходной цены и наценки.

Функция для обновления сводной стоимости и количества товаров в группе: пересчитывает количество и общую стоимость для группы товаров при изменениях в таблице Товары.

**6. Наценка (Markup)**

Наценка (markup) — процент увеличения цены товара относительно его закупочной цены. В таблице group\_products наценка хранится как значение от 0 до 1, где 0.2 означает 20%, 0.3 — 30% и так далее. Используется для расчета розничной цены:

Розничная цена=Приходная цена×(1+markup)

Розничная цена=Приходная цена×(1+markup)

**7. SQL-синтаксис для создания объектов базы данных**

Основные SQL-команды, которые понадобятся:

CREATE TABLE — команда для создания таблицы в базе данных.

ALTER TABLE — изменяет структуру существующей таблицы.

CREATE FUNCTION — создает пользовательскую функцию в PL/pgSQL.

CREATE TRIGGER — создает триггер, который автоматически запускает указанную функцию при определенных событиях.

**8. Автоматический расчет значений с использованием триггеров**

В данной работе требуется автоматизировать расчеты значений в таблицах с помощью триггеров:

Расчет retail\_price при вставке/обновлении строки в таблице products.

Пересчет total\_retail\_cost и stock\_quantity в таблице group\_products при изменении количества или розничной цены товара.

Пересчет всех розничных цен в таблице products при изменении markup для группы товаров.