

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра «Программная инженерия»

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1

по предмету “Серверное программирование”

Разработка архитектуры базы данных

тема

Преподаватель

А.А. Даничев

подпись, дата

инициалы, фамилия

Студент ЗКИ21-16БВВ 031625881

20.03.2025

Е.М.Хорошко

подпись, дата

инициалы, фамилия

Красноярск 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1. Ход выполнения работы.....	4
1.1 Настройка пользователя БД PostgreSQL.....	4
1.2 Схема БД. UML диаграмма классов.....	5
1.3 Создание схемы данных и таблиц в PostgreSQL.....	6
2. Заполнение таблиц БД. SQL запрос.....	7
3. Запрос на вывод CROSS JOIN(,) информации по всем созданным таблицам.....	8
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	10

ВВЕДЕНИЕ

Цели и задачи работы:

В рамках данной практической работы необходимо разработать архитектуру базы данных будущего приложения, а также реализовать базовые таблицы из данной архитектуры. Целью данной практической работы будет создание схемы базы данных по выбранной теме, состоящей из минимум 5-ти таблиц, заполненной не менее чем 10-тью записями в каждой, а также составление запроса на вывод информации по всем созданным таблицам.

В ходе практической работы необходимо выполнить следующие задачи:

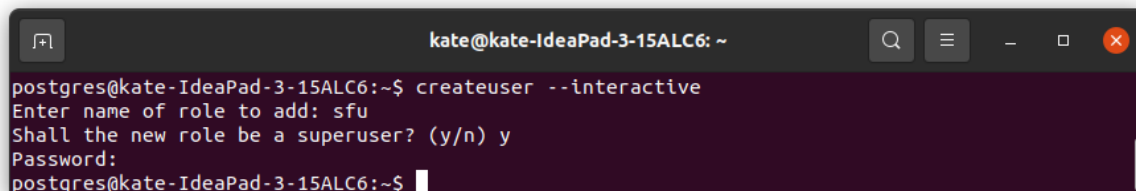
1. Необходимо провести предварительную установку средств разработки и СУБД. В данной работе в качестве СУБД будет использована PostgreSQL.
2. Настроить пользователя БД
3. Придумать и оформить архитектуру базы данных. Результатом данного пункта должна являться UML диаграмма классов, или же иное представление схемы БД по выбранной предметной области. (минимум 5 таблиц)
4. В выбранной СУБД создать схему данных и реализовать таблицы:
 - создать таблицы
 - создать колонки с их типами
 - задать первичные ключи
5. Используя SQL запросы, заполнить базовые таблицы не менее чем 10 записями в каждой.
6. Составить запрос на вывод CROSS JOIN (,) информации по всем созданным таблицам.

В данной работе будет реализована база данных ювелирных изделий для интернет-магазина.

1. Ход выполнения работы

1.1 Настройка пользователя БД PostgreSQL

Создадим нового пользователя sfu.

A screenshot of a terminal window with a dark background. The window title is 'kate@kate-IdeaPad-3-15ALC6: ~'. The terminal shows the command 'postgres@kate-IdeaPad-3-15ALC6:~\$ createuser --interactive' and its interactive prompts: 'Enter name of role to add: sfu', 'Shall the new role be a superuser? (y/n) y', and 'Password:'. The prompt returns to the shell 'postgres@kate-IdeaPad-3-15ALC6:~\$' with a cursor.

```
postgres@kate-IdeaPad-3-15ALC6:~$ createuser --interactive
Enter name of role to add: sfu
Shall the new role be a superuser? (y/n) y
Password:
postgres@kate-IdeaPad-3-15ALC6:~$
```

Рис. 1 - Создание пользователя sfu

1.2 Схема БД. UML диаграмма классов

В данной работе будет реализована база данных ювелирных изделий для интернет-магазина.

Основные сущности базы данных:

- "Ювелирное изделие"(Jewelry). В данной сущности будут отражены основные атрибуты: название, вес, цена, производитель, тип изделия.
- "Тип изделия"(JewelryType) - изделия могут быть разных типов — кольца, серьги, ожерелья и т.д. - для видов изделий создана отдельная таблица.
- "Материал" (Material) - изделия могут быть из золота, серебра, платины, с драгоценными камнями.
- "Материал украшения" (JewelryMaterial) - таблица Jewelry связана с JewelryType (один ко многим), с Material (многие ко многим) через JewelryMaterial
- "Клиент"(Customer) - хранит id покупателя и его номер телефона.
- "Заказ"(Order) - может содержать несколько изделий, а также общую сумму заказа.
- "Позиция заказа"(OrderItem) - хранит количество и цену на момент заказа, связывает заказы с изделиями.

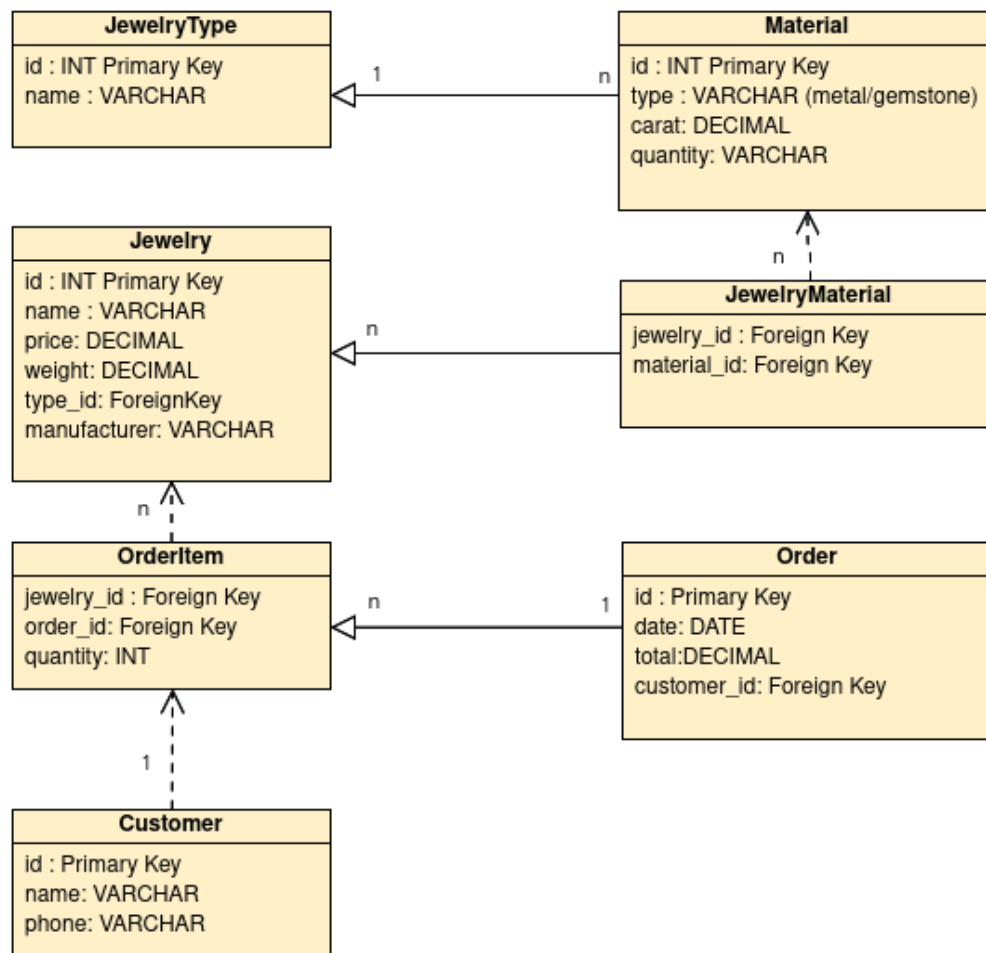


Рис. 2 - UML диаграмма классов

1.3 Создание схемы данных и таблиц в PostgreSQL

-- Типы изделий (кольца, серьги и т.д.)

```
CREATE TABLE JewelryType (
    id INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL
);
```

-- Материалы и камни

```
CREATE TABLE Material (
    id INT PRIMARY KEY,
```

```

        type VARCHAR(20) CHECK (type IN ('metal', 'gemstone')),
        name VARCHAR(100) NOT NULL,
        carat DECIMAL(5,2),
        quality VARCHAR(50)
    );

-- Ювелирные изделия
CREATE TABLE Jewelry (
    id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    price DECIMAL(10,2) CHECK (price > 0),
    weight DECIMAL(8,2),
    type_id INT REFERENCES JewelryType(id),
    manufacturer VARCHAR(100) -- Производитель как строка
);

-- Связь изделий с материалами
CREATE TABLE JewelryMaterial (
    jewelry_id INT REFERENCES Jewelry(id),
    material_id INT REFERENCES Material(id),
    PRIMARY KEY (jewelry_id, material_id)
);

-- Клиенты
CREATE TABLE Customer (
    id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    phone VARCHAR(20)
);

-- Заказы
CREATE TABLE Order (
    id INT PRIMARY KEY,
    date DATE DEFAULT CURRENT_DATE,

```

```
total DECIMAL(10,2),
customer_id INT REFERENCES Customer(id)
);
```

-- Позиции заказов

```
CREATE TABLE OrderItem (
    order_id INT REFERENCES Order(id),
    jewelry_id INT REFERENCES Jewelry(id),
    quantity INT CHECK (quantity > 0),
    PRIMARY KEY (order_id, jewelry_id)
);
```

2. Заполнение таблиц БД. SQL запрос

-- Типы изделий

```
INSERT INTO JewelryType (id, name) VALUES
(1, 'Кольцо'),
(2, 'Серьги');
```

-- Материалы

```
INSERT INTO Material (id, type, name, carat, quality) VALUES
(101, 'metal', 'Золото 585', NULL, 'Высшая проба'),
(102, 'gemstone', 'Бриллиант', 0.5, 'Идеальная огранка');
```

-- Изделие

```
INSERT INTO Jewelry (id, name, price, weight, type_id, manufacturer) VALUES
(1001, 'Обручальное кольцо', 1500.00, 5.2, 1, 'Tiffany & Co');
```

-- Связь изделия с материалами

```
INSERT INTO JewelryMaterial (jewelry_id, material_id) VALUES
(1001, 101),
(1001, 102);
```

3. Запрос на вывод CROSS JOIN(,) информации по всем созданным таблицам

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Установка RARS
2. Список команд Risc-V