

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.....	4
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:

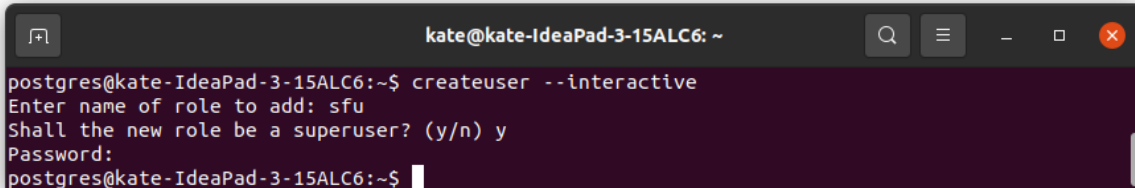
1. Необходимо провести предварительную установку средств разработки и СУБД. В данной работе в качестве СУБД будет использована PostgreSQL.
2. Настроить пользователя БД
3. Придумать и оформить архитектуру базы данных. Результатом данного пункта должна являться UML диаграмма классов, или же иное представление 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.



```
kate@kate-IdeaPad-3-15ALC6: ~  
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) - данная сущность реализована в отдельной таблице, т.к. изделия могут быть разных типов — кольца, серьги, ожерелья и т.д. Jewelry связана с JewelryType.
- "Материал" (Material) - изделия могут быть из золота, серебра, платины, с драгоценными камнями и т.д. Также в данной таблице будет храниться информация о каратности и количестве камней в изделии, если материал изделия - gemstone. Для металлов проба будет указана в поле типа материала, например "золото 585".
- "Материал украшения" (JewelryMaterial) - данная таблица связывает сущность Украшение (Jewelry) с материалом (Material) (многие ко многим).
- "Клиент"(Customer) - хранит id покупателя, номер телефона, email.
- "Заказ"(Order) - хранит id покупателя, дату заказа, общую сумму заказа.
- "Позиция заказа"(OrderItem) - хранит количество одного заказываемого изделия. Так как заказ может содержать несколько изделий, данная таблица связывает id заказа с изделиями.

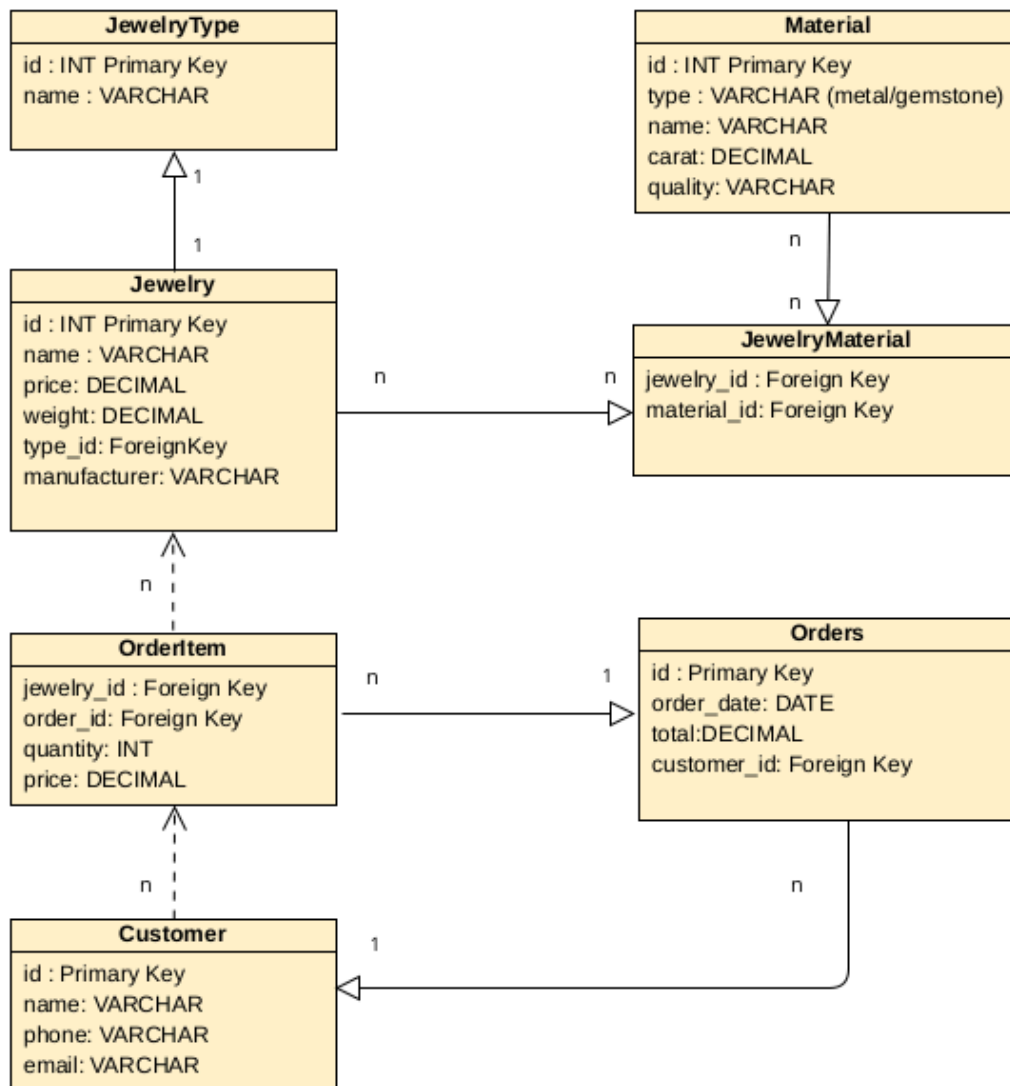
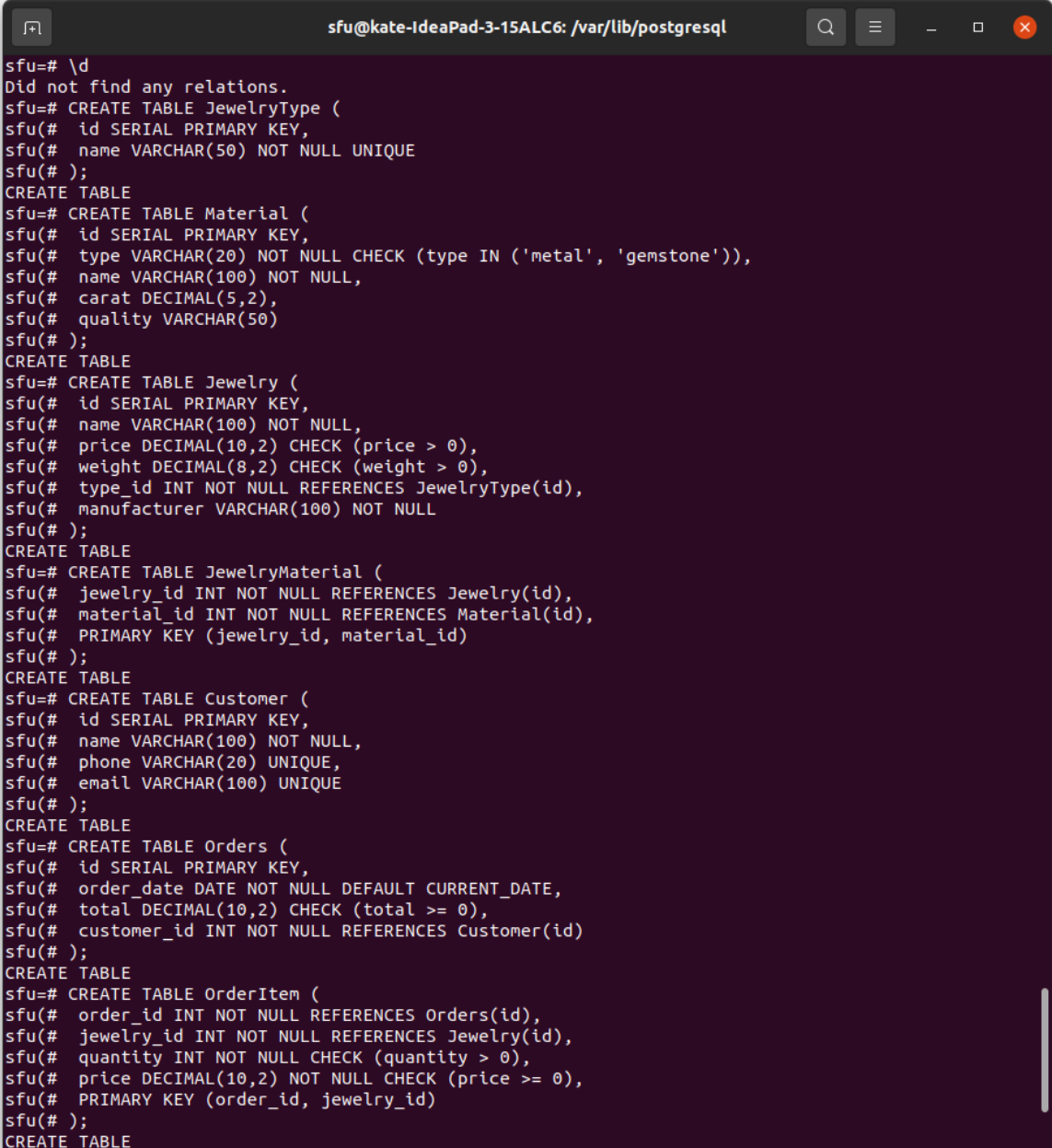


Рис. 2 - UML диаграмма классов

### 1.3 Создание схемы данных и таблиц в PostgreSQL



```
sfu=# \d
Did not find any relations.
sfu=# CREATE TABLE JewelryType (
sfu(#  id SERIAL PRIMARY KEY,
sfu(#  name VARCHAR(50) NOT NULL UNIQUE
sfu(# );
CREATE TABLE
sfu=# CREATE TABLE Material (
sfu(#  id SERIAL PRIMARY KEY,
sfu(#  type VARCHAR(20) NOT NULL CHECK (type IN ('metal', 'gemstone')),
sfu(#  name VARCHAR(100) NOT NULL,
sfu(#  carat DECIMAL(5,2),
sfu(#  quality VARCHAR(50)
sfu(# );
CREATE TABLE
sfu=# CREATE TABLE Jewelry (
sfu(#  id SERIAL PRIMARY KEY,
sfu(#  name VARCHAR(100) NOT NULL,
sfu(#  price DECIMAL(10,2) CHECK (price > 0),
sfu(#  weight DECIMAL(8,2) CHECK (weight > 0),
sfu(#  type_id INT NOT NULL REFERENCES JewelryType(id),
sfu(#  manufacturer VARCHAR(100) NOT NULL
sfu(# );
CREATE TABLE
sfu=# CREATE TABLE JewelryMaterial (
sfu(#  jewelry_id INT NOT NULL REFERENCES Jewelry(id),
sfu(#  material_id INT NOT NULL REFERENCES Material(id),
sfu(#  PRIMARY KEY (jewelry_id, material_id)
sfu(# );
CREATE TABLE
sfu=# CREATE TABLE Customer (
sfu(#  id SERIAL PRIMARY KEY,
sfu(#  name VARCHAR(100) NOT NULL,
sfu(#  phone VARCHAR(20) UNIQUE,
sfu(#  email VARCHAR(100) UNIQUE
sfu(# );
CREATE TABLE
sfu=# CREATE TABLE Orders (
sfu(#  id SERIAL PRIMARY KEY,
sfu(#  order_date DATE NOT NULL DEFAULT CURRENT_DATE,
sfu(#  total DECIMAL(10,2) CHECK (total >= 0),
sfu(#  customer_id INT NOT NULL REFERENCES Customer(id)
sfu(# );
CREATE TABLE
sfu=# CREATE TABLE OrderItem (
sfu(#  order_id INT NOT NULL REFERENCES Orders(id),
sfu(#  jewelry_id INT NOT NULL REFERENCES Jewelry(id),
sfu(#  quantity INT NOT NULL CHECK (quantity > 0),
sfu(#  price DECIMAL(10,2) NOT NULL CHECK (price >= 0),
sfu(#  PRIMARY KEY (order_id, jewelry_id)
sfu(# );
CREATE TABLE
```

Рис.3 - Создание таблиц. SQL-запросы.

## 2. Заполнение таблиц БД. SQL запрос

```
sfu@kate-ideaPad-3-15ALC6: /var/lib/postgresql
sfu=# INSERT INTO JewelryType (name) VALUES
sfu-# ('Кольцо'), ('Серьги'), ('Цепочка'), ('Браслет'), ('Подвеска'),
sfu-# ('Кулон'), ('Пирсинг'), ('Запонки'), ('Брошь'), ('Диадема');
INSERT 0 10
sfu=# INSERT INTO Material (type, name, carat, quality) VALUES
sfu-# ('metal', 'Золото 585', NULL, 'Высшая проба'),
sfu-# ('metal', 'Серебро 925', NULL, 'Стандарт'),
sfu-# ('gemstone', 'Бриллиант', 1.2, 'VVS1'),
sfu-# ('metal', 'Платина', NULL, '950 проба'),
sfu-# ('gemstone', 'Сапфир', 0.8, 'Синий'),
sfu-# ('gemstone', 'Изумруд', 0.5, 'AAA'),
sfu-# ('metal', 'Титан', NULL, 'Медицинский'),
sfu-# ('gemstone', 'Рубин', 0.7, 'Кровавый'),
sfu-# ('metal', 'Палладий', NULL, '999 проба'),
sfu-# ('gemstone', 'Аметист', 0.3, 'Фиолетовый');
INSERT 0 10
sfu=# INSERT INTO Jewelry (name, price, weight, type_id, manufacturer) VALUES
sfu-# ('Кольцо обручальное', 1500.00, 5.0, 1, 'Tiffany'),
sfu-# ('Серьги-гвоздики', 450.00, 3.2, 2, 'Sokolov'),
sfu-# ('Цепочка золотая', 1200.00, 8.5, 3, 'Adamas'),
sfu-# ('Браслет Pandora', 950.00, 12.0, 4, 'Pandora'),
sfu-# ('Подвеска-сердце', 300.00, 2.1, 5, 'Swarovski'),
sfu-# ('Кулон с бриллиантом', 2500.00, 4.8, 6, 'Cartier'),
sfu-# ('Пирсинг в нос', 150.00, 1.5, 7, 'Local Jewelry'),
sfu-# ('Запонки серебряные', 200.00, 6.0, 8, 'Frey Wille'),
sfu-# ('Брошь Vintage', 1800.00, 7.3, 9, 'Chopard'),
sfu-# ('Диадема свадебная', 5000.00, 15.0, 10, 'Harry Winston');
INSERT 0 10
sfu=# INSERT INTO JewelryMaterial (jewelry_id, material_id) VALUES
sfu-# (1,1), (1,3), (2,2), (3,1), (4,4), (5,5), (6,3), (7,7), (8,2), (9,9);
INSERT 0 10
sfu=# INSERT INTO Customer (name, phone, email) VALUES
sfu-# ('Иванова Анна', '+79161234567', 'anna@mail.ru'),
sfu-# ('Петров Дмитрий', '+79035556677', 'dmitry@yandex.ru'),
sfu-# ('Сидорова Мария', '+79269874563', 'maria@gmail.com'),
sfu-# ('Козлов Артем', '+79031237894', 'artem@mail.com'),
sfu-# ('Новикова Елена', '+79150983456', 'elena@yahoo.com'),
sfu-# ('Васильев Игорь', '+79087651234', 'igor@bk.ru'),
sfu-# ('Морозова Ольга', '+79265439876', 'olga@inbox.ru'),
sfu-# ('Лебедев Павел', '+79024561239', 'pavel@list.ru'),
sfu-# ('Смирнова Виктория', '+79167894523', 'vika@rambler.ru'),
sfu-# ('Кузнецов Андрей', '+79035671234', 'andrey@gmail.com');
INSERT 0 10
sfu=# INSERT INTO Orders (order_date, total, customer_id) VALUES
sfu-# ('2023-10-01', 1500.00, 1),
sfu-# ('2023-10-02', 450.00, 2),
sfu-# ('2023-10-03', 1200.00, 3),
sfu-# ('2023-10-04', 950.00, 4),
sfu-# ('2023-10-05', 300.00, 5),
sfu-# ('2023-10-06', 2500.00, 6),
sfu-# ('2023-10-07', 150.00, 7),
sfu-# ('2023-10-08', 200.00, 8),
sfu-# ('2023-10-09', 1800.00, 9),
sfu-# ('2023-10-10', 5000.00, 10);
INSERT 0 10
```

Рис.4.1 - Заполнение таблиц тестовыми данными.

```
sfu@kate-ideaPad-3-15ALC6: /var/lib/postgresql
sfu=# INSERT INTO OrderItem (order_id, jewelry_id, quantity, price) VALUES
sfu-# (1,1,1,1500.00), (2,2,1,450.00), (3,3,1,1200.00), (4,4,1,950.00),
sfu-# (5,5,1,300.00), (6,6,1,2500.00), (7,7,1,150.00), (8,8,1,200.00),
sfu-# (9,9,1,1800.00), (10,10,1,5000.00);
INSERT 0 10
sfu=#
```

Рис.4.2 - Заполнение таблиц тестовыми данными (продолжение).

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

Так как выполнение запроса CROSS JOIN (,) по всем таблицам выведет на экран  $10^7$  строк результатов, для демонстрации данных будем использовать запрос JOIN и LEFT JOIN, охватывающий основную информацию по всем таблицам.

```

sfu=# SELECT
  c.name AS customer_name,
  c.phone AS customer_phone,
  c.email AS customer_email,
  o.order_date,
  j.name AS jewelry_name,
  jt.name AS jewelry_type,
  STRING_AGG(m.name, ' ') AS materials,
  oi.quantity,
  oi.price AS unit_price,
  (oi.quantity * oi.price) AS total_price
FROM Customer c
JOIN Orders o ON c.id = o.customer_id
JOIN OrderItem oi ON o.id = oi.order_id
JOIN Jewelry j ON oi.jewelry_id = j.id
JOIN JewelryType jt ON j.type_id = jt.id
LEFT JOIN JewelryMaterial jm ON j.id = jm.jewelry_id
LEFT JOIN Material m ON jm.material_id = m.id
GROUP BY
  c.name,
  c.phone,
  c.email,
  o.order_date,
  j.name,
  jt.name,
  oi.quantity,
  oi.price,
  o.id,
  j.id
ORDER BY o.order_date ASC
LIMIT 10;

```

Иванова Анна	+79161234567	anna@mail.ru	2023-10-01	Кольцо обручальное	Кольцо	Бриллиант, Золото 585	1	1500.00	1500.00
Петров Дмитрий	+79035556677	dmitry@yandex.ru	2023-10-02	Серьги-гвоздики	Серьги	Серебро 925	1	450.00	450.00
Сидорова Мария	+79269874563	maria@gmail.com	2023-10-03	Цепочка золотая	Цепочка	Золото 585	1	1200.00	1200.00
Козлов Артем	+79031237894	artem@mail.com	2023-10-04	Браслет Pandora	Браслет	Платина	1	950.00	950.00
Новикова Елена	+79150983456	elena@yahoo.com	2023-10-05	Подвеска-сердце	Подвеска	Сапфир	1	300.00	300.00
Васильев Игорь	+79087651234	igor@bk.ru	2023-10-06	Кулон с бриллиантом	Кулон	Бриллиант	1	2500.00	2500.00
Морозова Ольга	+79265439876	olga@inbox.ru	2023-10-07	Пирсинг в нос	Пирсинг	Титан	1	150.00	150.00
Лебедев Павел	+79024561239	pavel@list.ru	2023-10-08	Запонки серебряные	Запонки	Серебро 925	1	200.00	200.00
Смирнова Виктория	+79167894523	vika@rambler.ru	2023-10-09	Брошь Vintage	Брошь	Палладий	1	1800.00	1800.00
Кузнецов Андрей	+79035671234	andrey@gmail.com	2023-10-10	Диадема свадебная	Диадема		1	5000.00	5000.00

Рис.5 - Запрос к базе данных.

**Запрос выполняет следующие действия:**

1. **Соединяет таблицы:** Orders и Customer по ID клиента, Orders и OrderItem по ID заказа, OrderItem и Jewelry по ID изделия, JewelryType и Jewelry по ID типа изделия, Material и Jewelry через



таблицу JewelryMaterial по ID изделия и ID материала соответственно.

2. **Возвращает данные:** имя клиента, номер телефона, почту, дату заказа, название ювелирного изделия, тип изделия, материал изделия, количество товара в позиции заказа, цену за единицу и общую стоимость позиции.
3. **Упорядочивает результаты** по дате заказа (свежие сверху) и имени клиента.
4. **Ограничивает вывод** в тестовых условиях до 10-ти записей.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация к PostgreSQL 17.4 - [Эл.ресурс] <https://postgrespro.ru/docs/postgresql/17/index>
2. Онлайн-редактор диаграмм - [Эл. ресурс] <https://www.visual-paradigm.com/>