Практическая работа №2

Управление данными в графической базе данных Neo4j

Цель работы

Получение практических навыков работы с графической базой данных Neo4j.

Индивидуальное задание

База данных «Ювелирная мастерская»

Вы работаете в ювелирной мастерской. Ваша мастерская осуществляет изготовление ювелирных изделий для частных лиц на заказ. Вы работаете с определенными материалами (платина, золото, серебро, различные драгоценные камни и т.д.). При обращении к Вам потенциального клиента Вы определяетесь с тем, какое именно изделие ему необходимо. Все изготавливаемые Вами изделия принадлежат к некоторому типу (серьги, кольца, броши, браслеты), бывают выполнены из определенного материала, имеют некоторый вес и цену (включающую стоимость материалов и работы). Ювелирное изделие может состоять из нескольких материалов. Кроме того, постоянным клиентам мастерская предоставляет скидки.

Ход выполнения работы

Для быстрого старта Neo4j воспользуемся Docker, который может одной командой развернуть серверную и клиентскую части базы данных Neo4j.

Для запуска Neo4j контейнера можно воспользоваться Bash-командой (рис. 1).

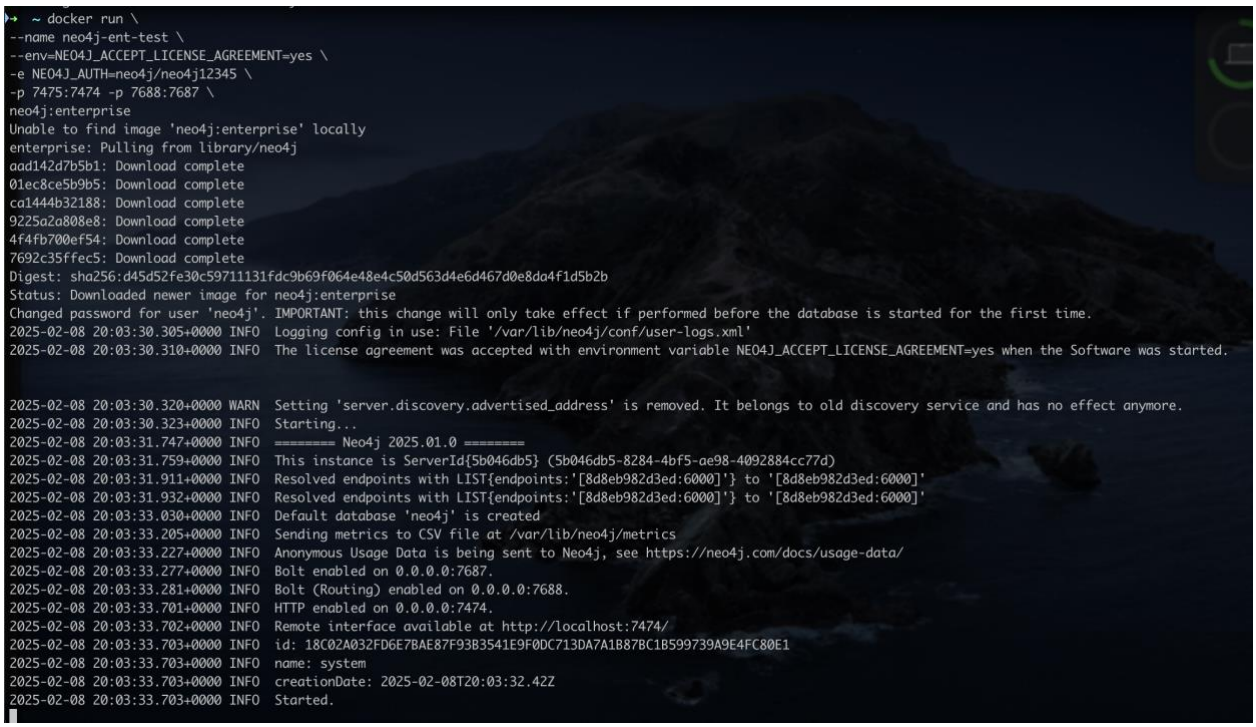

Рисунок 1 – Запуск контейнера

Создадим БД ювелирного магазина (store). Для создания БД перейдем в графический клиент Neo4j. Для создания БД sales введем команду:

```
CREATE DATABASE store;
```

При успешном создании можно выбрать соответствующую БД в списке баз данных (рис. 2).

Рисунок 2 – Создание БД

Для заполнения БД данными необходимо придумать структуру хранения данных. Если бы мы работали с реляционной БД, то реализовали бы схему показанную на рисунке 3.

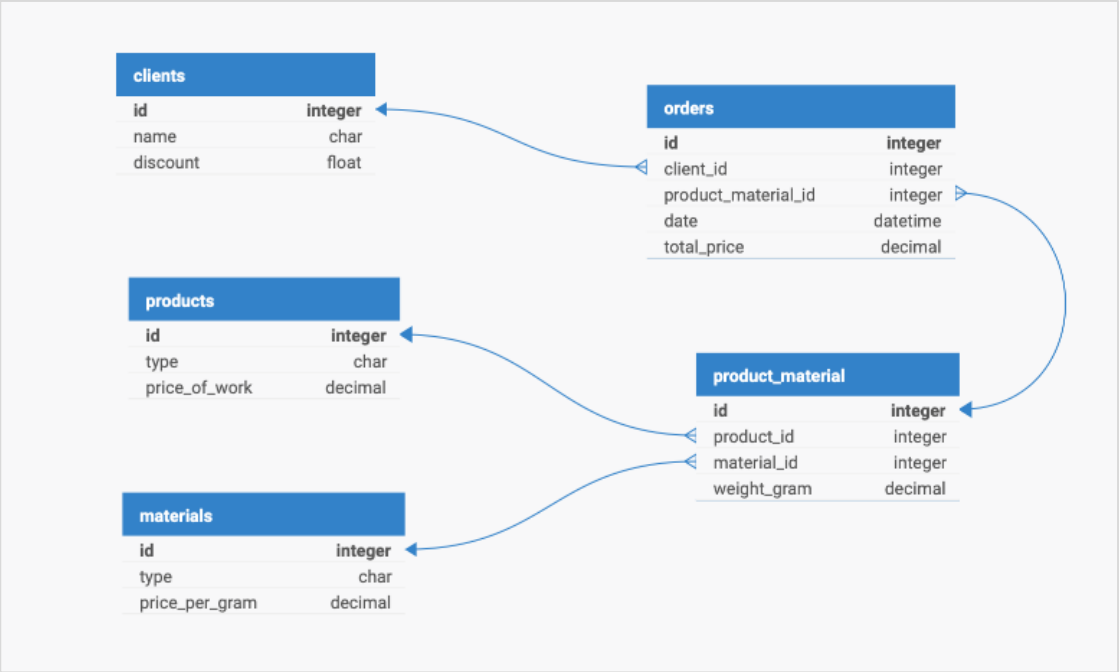

Рисунок 3 – Схема реляционной БД

Поскольку мы работаем с графовой БД, необходимо определить, какие сущности будут узлами (Nodes), а какие ребрами/связями (Relationships).

Исходя из предметной области сущностями являются:

– Client (клиент);

– Order (заказ);

– Product (изделие);

– Material (материал);

Ребра будут следующие:

– BUY (купил);

– MADE_OF (сделан из);

Схема графовой модели данных, сделанная по индивидуальному варианту представлена на рисунке 4.

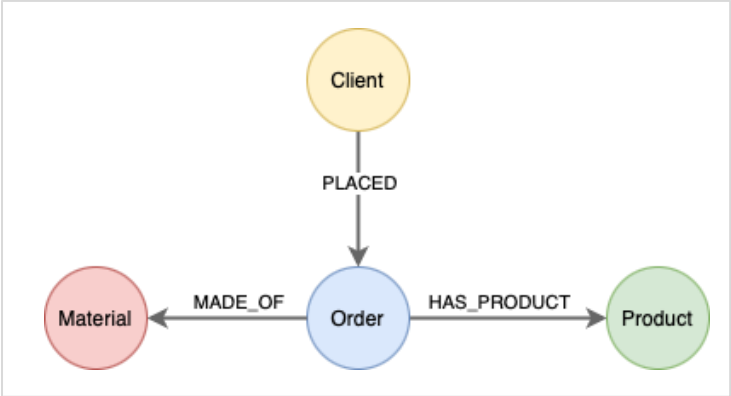

Рисунок 4 – Модель графовой БД

Для заполнения БД данными воспользуемся Cypher-кодом, приведенным в листинге 1 ниже.

Листинг 1 – Код заполнения графовой БД

```
// Создание клиентов
CREATE
(client1:Client {id:1, name: 'Никита', surname:'Макаров', birthDate:
date('2004-07-02'), discount:0.5}),
(client2:Client {id:2, name: 'Умар', surname:'Мазориев', birthDate:
date('2004-09-27'), discount:0.1}),
...

// Создание изделий
CREATE
(p1:Product {id:1, name: 'Кольцо'}),
(p2:Product {id:2, name: 'Серьги'}),
...

// Создание материалов
CREATE
(m1:Material {id:1, name:'Золото', pricePerGram:4000}),
(m2:Material {id:2, name:'Серебро', pricePerGram:500}),
...

// Создание заказов вместе с клиентами и изделиями
MATCH (c1:Client {id:1}), (p1:Product {id:1})
CREATE
(c1)-[:PLACED]->(o1:Order {id:1, date: date('2024-06-03'),
price:20000}),
(o1)-[:HAS_PRODUCT {quantity:1}]->(p1);
```

```
MATCH (c2:Client {id:2}), (p2:Product {id:2})
CREATE
(c2)-[:PLACED]->(o2:Order {id:2, date: date('2024-06-10'),
price:15000}),
(o2)-[:HAS_PRODUCT {quantity:1}]->(p2);
...

// Связь изделий с материалами
MATCH (p1:Product {id:1}), (m1:Material {id:1}), (m4:Material {id:4})
CREATE (p1)-[:MADE_OF {weight: 10}]->(m1),
       (p1)-[:MADE_OF {weight: 2}]->(m4);

MATCH (p2:Product {id:2}), (m2:Material {id:2})
CREATE (p2)-[:MADE_OF {weight: 8}]->(m2);
...
```

Отобразим все данные, которые мы внесли в БД в графическом виде (рис. 5) с помощью команды: `MATCH (n) RETURN n.`

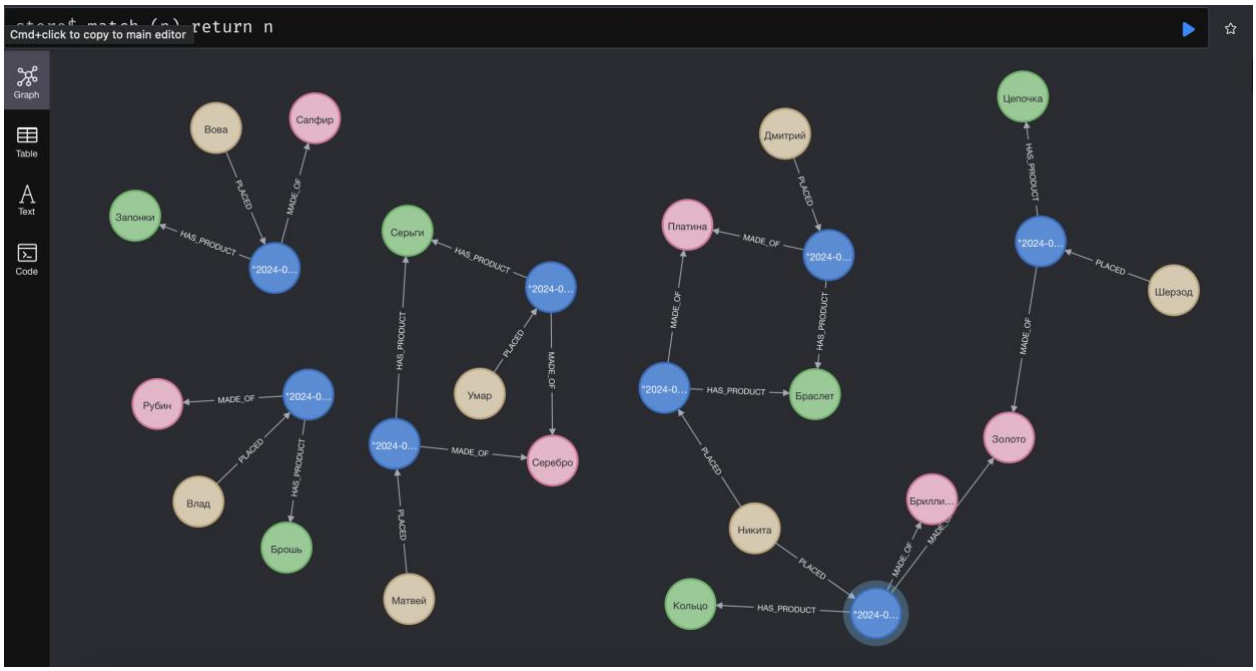

Рисунок 5 – Графическое представление данных в БД

Далее продумаем и реализуем запросы разной сложности на Cypher в клиентской части БД.

Запрос 1 – Вывести всех пользователей (рис. 6).

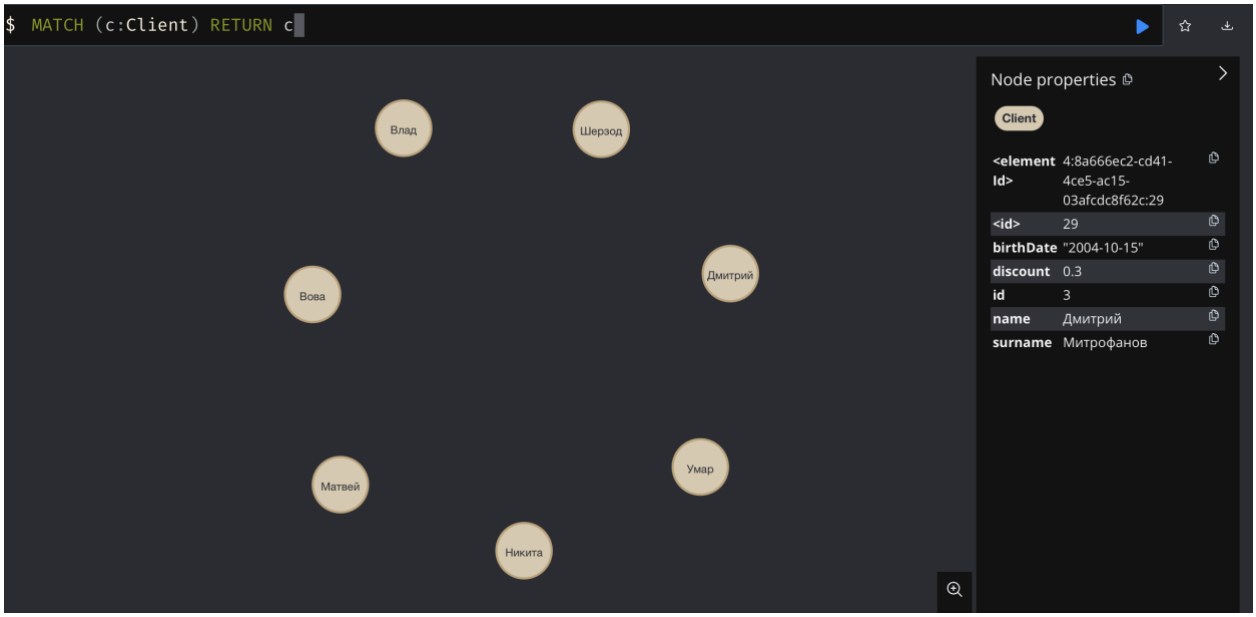

Рисунок 6 – Все пользователи

Запрос 2 – Вывести все изделия (рис. 7).

Рисунок 7 – Все изделия

Запрос 3 – Вывести все заказы Никиты Макарова (рис. 8).

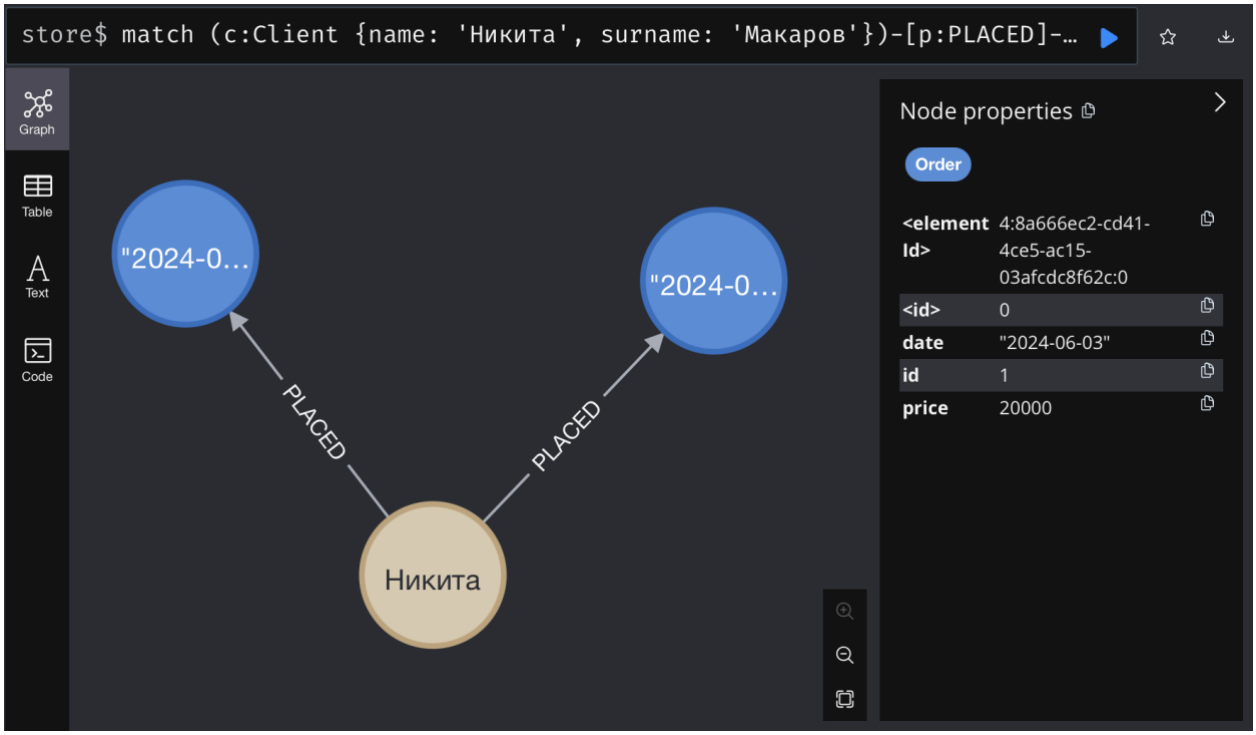

Рисунок 8 – Все заказы пользователя

Запрос 4 - вывести всех клиентов и их заказы (рис. 9).

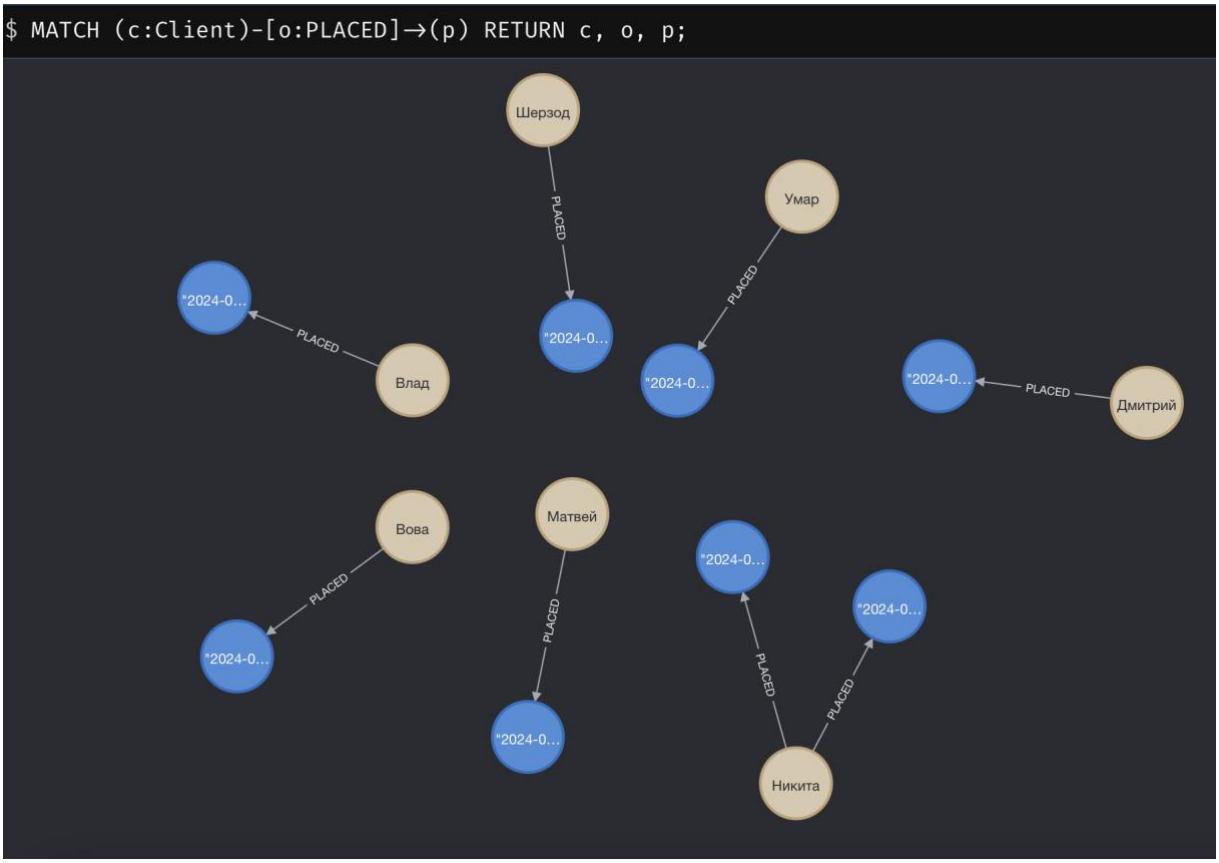

Рисунок 9 – Клиенты и их заказы

Запрос 5 – Вывести все заказы, изделия и материалы (рис. 10).

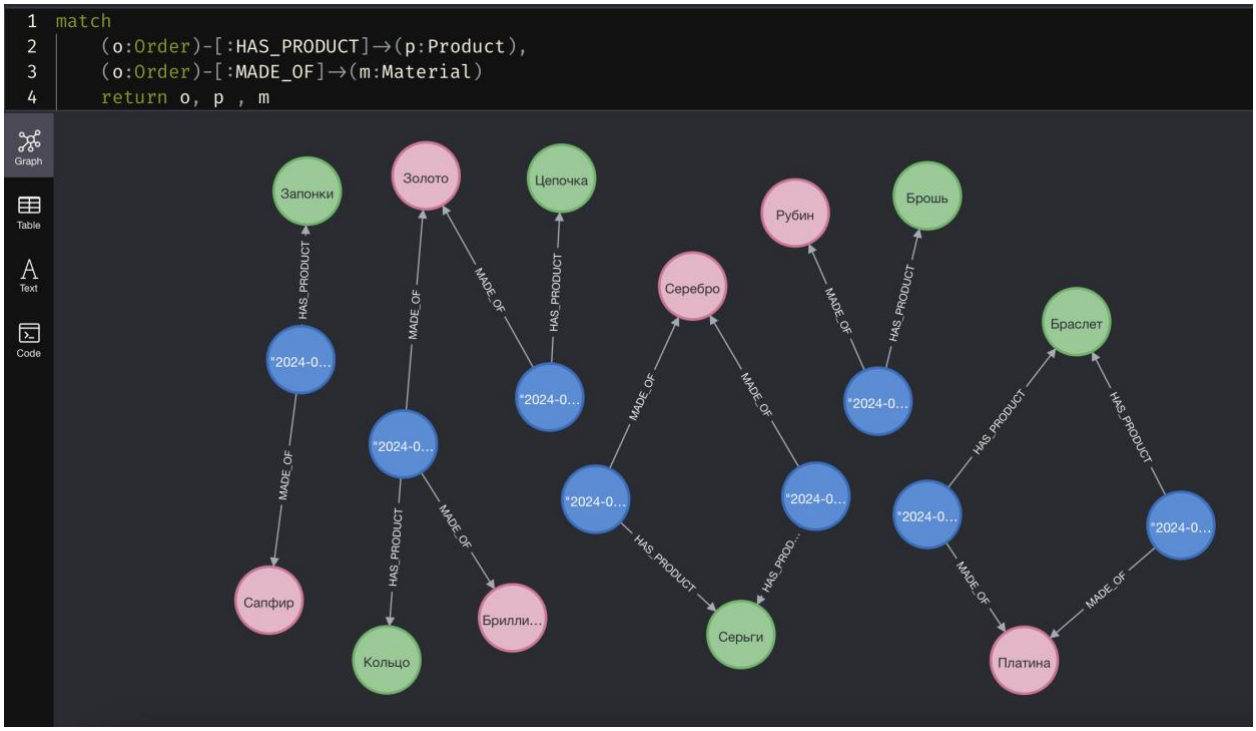

Рисунок 10 – Заказы, изделия и материалы

Запрос 6 – Вывести сумму всех заказов, сделанных в первую половину июня (рис. 11).

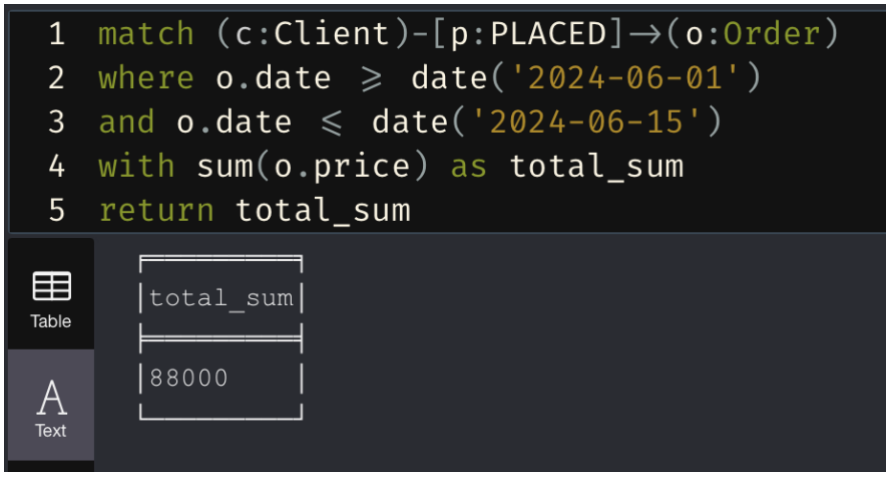

Рисунок 11 – Заказы, сделанные в первую половину июня

Запрос 7 – Вывести изделия и кол-во их приобретений в порядке убывания количества (рис. 12).

```
1 match (o:Order)-[hp:HAS_PRODUCT]→(p:Product)
2 with p, count(hp) as Количество_покупок
3 order by Количество_покупок desc
4 return p.name as Изделие, Количество_покупок
```

Table | Text | Code

	Изделие	Количество_покупок
1	"Серьги"	2
2	"Браслет"	2
3	"Кольцо"	1
4	"Цепочка"	1
5	"Брошь"	1
6	"Запонки"	1

Рисунок 12 – Популярность изделий

Запрос 8 – Вывести имена и фамилии клиентов, которые совершили ровно 2 заказа (рис. 13).

```
1 match (c:Client)-[p:PLACED]→(o:Order)
2 with c, count(p) as Заказы
3 where Заказы = 2
4 return
5     c.name as Имя,
6     c.surname as Фамилия,
7     Заказы
```

Table | Text

	Имя	Фамилия	Заказы
1	"Никита"	"Макаров"	2

Рисунок 13 – Клиенты, которые совершили 2 заказа

Запрос 9 – Вывести среднюю цену всех материалов (рис. 14).

```
match (m:Material)
with round(avg(m.pricePerGram),2) as Средняя_цена_материалов
return Средняя_цена_материалов
```

Table

	Средняя_цена_материалов
1	6916.67

Text

Рисунок 14 – Средняя цена материалов

Запрос 10 – Вывести кол-во заказов, сделанных каждым пользователем (рис. 15).

```
match(c:Client)-[p:PLACED]→(o:Order)
with c, count(p) as Количество_заказов
return
    c.id as ID,
    c.name as Имя,
    c.surname as Фамилия,
    Количество_заказов
```

Table

	ID	Имя	Фамилия	Количество_заказов
1	1	"Никита"	"Макаров"	2
2	2	"Умар"	"Мазориев"	1
3	3	"Дмитрий"	"Митрофанов"	1
4	4	"Шерзод"	"Махкамов"	1
5	5	"Влад"	"Лайхтман"	1
6	6	"Вова"	"Адидас"	1

Text

Code

Рисунок 15 – Кол-во заказов пользователей

Вывод

В ходе выполнения лабораторной работы я освоил основы работы с графовой базой данных Neo4j и язык запросов Cypher. Я изучил процесс развертывания Neo4j в среде Docker, что позволило мне быстро запустить базу данных без сложной настройки.

В рамках индивидуального задания я разработал структуру данных для ювелирной мастерской, определил основные сущности и связи между ними.

После заполнения базы я выполнил несколько запросов к БД различной сложности, например, выборка всех клиентов, заказов, изделий или агрегированные запросы с подсчетом среднего и количества.

Таким образом, работа позволила получить практические навыки работы и проектирования графовой БД Neo4j, а также умение работать с языком Cypher.