



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.04.01 Информатика и вычислительная техника

МАГИСТЕРСКАЯ ПРОГРАММА 09.04.01/07 Интеллектуальные системы анализа,
обработки и интерпретации больших данных

О Т Ч Е Т

по лабораторной работе № 6

Название: Работа с графовой базой данных Neo4j на примере разработки
рекомендательной системы

Дисциплина: Технология параллельных систем баз данных

Студент

ИУ6-12М

(Группа)

(Подпись, дата)

С.В. Астахов

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

(И.О. Фамилия)

Москва, 2023

Введение

1. Цель работы: изучение работы графовой базы данных Neo4j и её взаимодействия с документной NoSQL БД Elasticsearch на примере разработки рекомендательной системы.

Ход выполнения

2. Запуск ES и Neo4j.

Для упрощения установки и запуска воспользуемся докер-контейнерами с ES и Neo4j, пробросив необходимые порты в сеть основной машины (рисунок 1).

Name	Image	Status	Port(s)	Last started
 n4j1 880465669bc4	neo4j:lates	Running	7473:7473  Show all ports (3)	4 seconds ago
 es4 b4e97354d53b	bitnami/ela	Running	9200:9200  Show all ports (2)	19 seconds ago

Рисунок 1 — контейнеры с ES и Neo4j

Проверим работу ES с помощью GET-запроса к url “http://localhost:9200/?pretty” (рисунок 2).

```
{
  "name" : "b4e97354d53b",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "z9602ra5Q_WZaOJKqFBHQ",
  "version" : {
    "number" : "8.10.4",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "b4a62ac808e886ff032700c391f45f1408b2538c",
    "build_date" : "2023-10-11T22:04:35.506990650Z",
    "build_snapshot" : false,
    "lucene_version" : "9.7.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Рисунок 2 — информация о кластере ES

Проверим работу веб-интерфейса Neo4j (рисунок 3).

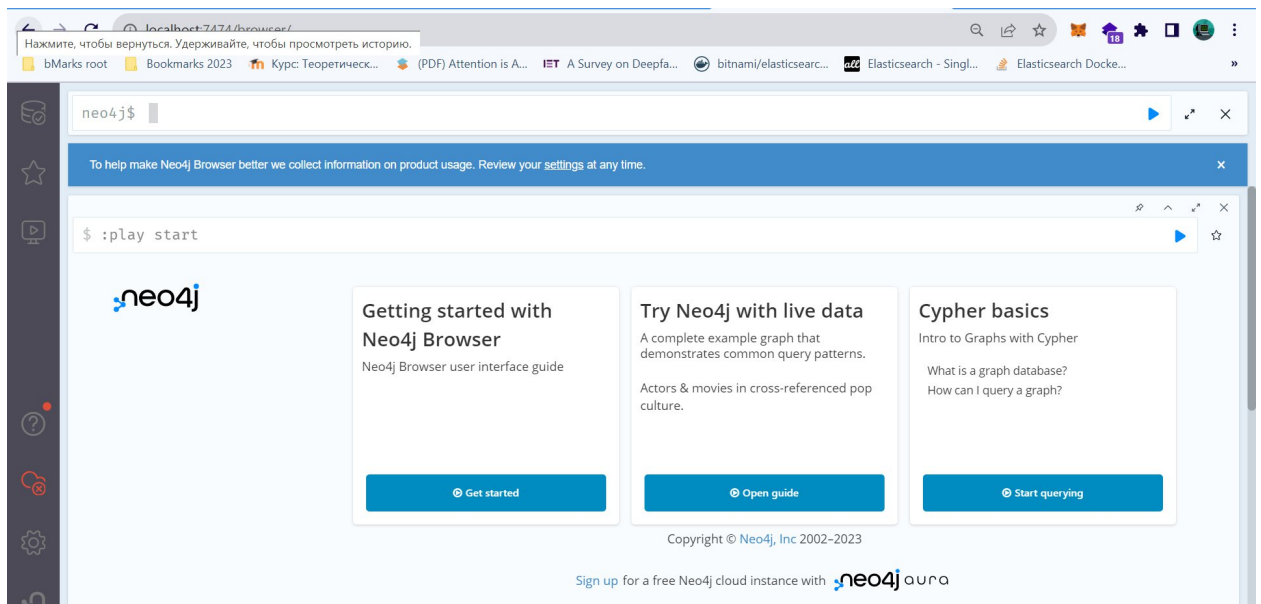


Рисунок 3 — веб-интерфейс Neo4j

Незначительно модифицировав файлы `f1.py` и `f2.py` для совместимости с актуальными версиями библиотек `elasticsearch` и `py2neo` выполним их. Первый запишет список рецептов в ES, второй — создаст граф связи ингредиентов и рецептов в Neo4j.

Проверим, что индекс появился в ES (рисунок 4).



Рисунок 4 — информация об индексе

Проверим, что граф рецептов и ингредиентов появился в веб-интерфейсе Neo4j (рисунок 5).

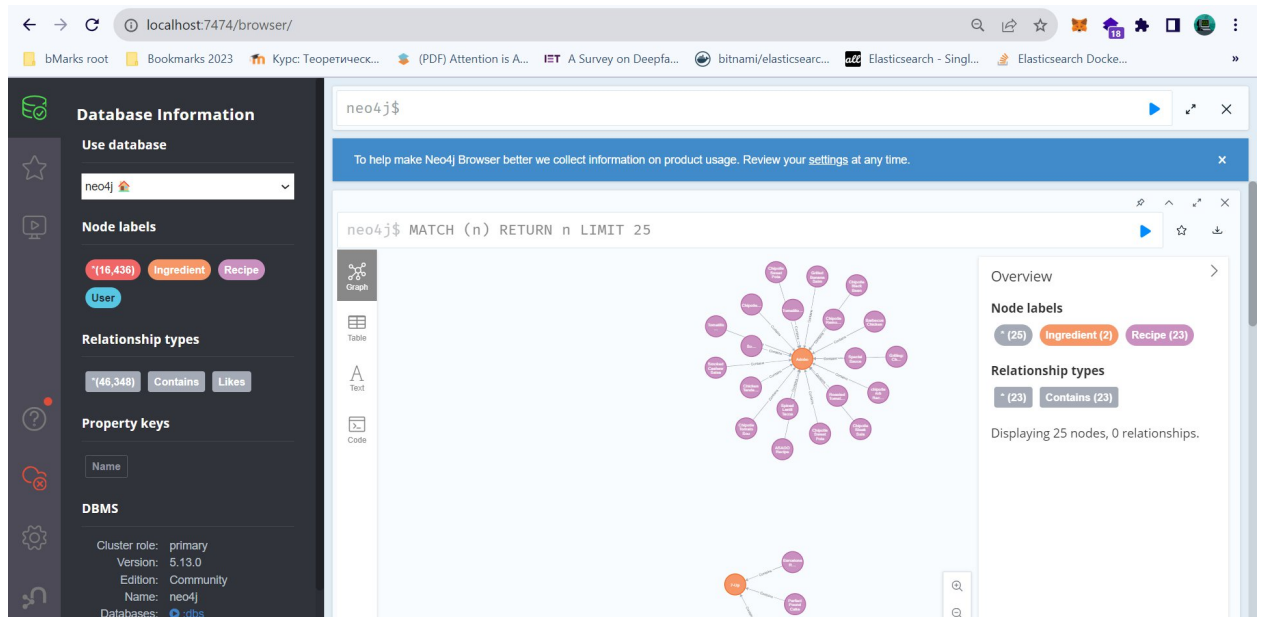


Рисунок 5 — фрагмент графа рецептов и ингредиентов

Запрос 1 — посмотрим какие ингредиенты чаще всего встречаются в рецептах. Текст запроса и комментарии представлены в листинге 1. Результаты — на рисунке 6.

Листинг 1 — часто встречающиеся ингредиенты

```
from py2neo import Graph, Node, Relationship

graph_db = Graph("bolt://localhost:7687")

try:
    # соединяем рецепты и ингредиенты; выбираем только ингредиенты; считаем ребра
    cur=graph_db.run("MATCH (REC:Recipe)-[r:Contains]->(ING:Ingredient) WITH ING, count(r) AS
    num RETURN ING.Name as Name, num ORDER BY num DESC LIMIT 10;")
except Exception:
    print(Exception)

while cur.forward():
    print(cur.current)
```

'Cloves'	6414
'Butter'	4994
'Flour'	3589
'Eggs'	2357
'Chicken'	1901
'Cheese'	1876
'Cinnamon'	1095
'Baking Powder'	923
'Chocolate'	887
'Bread'	886

Рисунок 6 — результат выполнения запроса

Запрос 2 — какие рецепты требуют больше всего ингредиентов. Написать запрос и выполнить его. Текст запроса представлен в листинге 2. Результаты — на рисунке 7. Запрос аналогичен предыдущему, но теперь в результат будут включаться узлы рецептов и количество трансцендентных им ребер.

Листинг 2 — сложные рецепты

```
try:
    cur=graph_db.run("MATCH (REC:Recipe)-[r:Contains]->(ING:Ingredient) WITH REC, count(r) AS
num RETURN REC.Name as recipe, num ORDER BY num DESC LIMIT 10;")
except Exception:
    print(Exception)

while cur.forward():
    print(cur.current)
```

'Cocoa Nib, Chocolate, and Citrus Dacquoise'	16
'Salmagundi'	15
'Wild rabbit with ham hock, carrots, asparagus and morels'	13
"Maxie's Shrimp and Grits Recipe"	13
'Espresso Pound Cake with Cranberries and Pecans'	13
'How To: Host the Perfect Easter Brunch Recipe'	13
'Smoky and Spicy Vegetarian Chili Recipe'	12
'Blueberry Dragon Fruit Chocolate Ganache Cupcakes Recipe'	12
"Bison and Red Wine Shepherd's Pie"	12
"Hog's pudding with seaweed, potato terrine and mushroom ketchup"	12

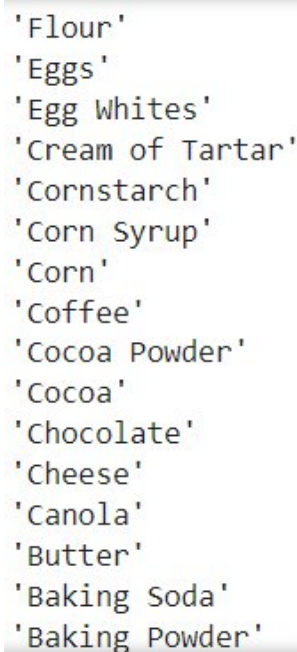
Рисунок 7 — результаты запроса

Запрос 3 — перечислить ингредиенты, связанные с конкретным рецептом (взять первый рецепт из предыдущего запроса). Написать запрос и выполнить его. Для этого запроса соединим рецепты и ингредиенты аналогично предыдущим запросам, но укажем фильтр по названию рецепта. Текст запроса представлен в листинге 3. Результаты — на рисунке 8.

Листинг 3 — ингредиенты одного блюда

```
try:
    cur=graph_db.run("MATCH (REC:Recipe {Name: 'Cocoa Nib, Chocolate, and Citrus Dacquoise'})-
[r:Contains]->(ING:Ingredient) WITH ING RETURN ING.Name as Name;")
except Exception:
    print(Exception)

while cur.forward():
    print(cur.current)
```



```
'Flour'
'Eggs'
'Egg Whites'
'Cream of Tartar'
'Cornstarch'
'Corn Syrup'
'Corn'
'Coffee'
'Cocoa Powder'
'Cocoa'
'Chocolate'
'Cheese'
'Canola'
'Butter'
'Baking Soda'
'Baking Powder'
```

Рисунок 8 — результаты запроса

Задача 1 — Включить в граф узел пользователя с именем Ragnar и описать его предпочтения. Реализующий задачу программный код приведен в листинге 4.

Листинг 4 — добавление пользователя и его предпочтений

```
# проверка существования и создание пользователя
UserNode=graph_db.nodes.match("User", Name="Ragnar").first()
if UserNode==None:
    UserNode = Node("User",Name="Ragnar")
```

```

graph_db.create(UserNode)

# предпочтение 1
RecipeNode=graph_db.nodes.match("Recipe", Name="ASADO Recipe").first()
NodesRelationship = Relationship(UserNode, "Likes", RecipeNode)
graph_db.create(NodesRelationship)

# предпочтение 2
RecipeNode=graph_db.nodes.match("Recipe", Name="Spiced Lentil Tacos").first()
NodesRelationship = Relationship(UserNode, "Likes", RecipeNode)
graph_db.create(NodesRelationship)

# предпочтение 3
RecipeNode=graph_db.nodes.match("Recipe", Name="Smoked Cashew Salsa").first()
NodesRelationship = Relationship(UserNode, "Likes", RecipeNode)
graph_db.create(NodesRelationship)

```

На рисунке 9 показан пользователь и его предпочтения.

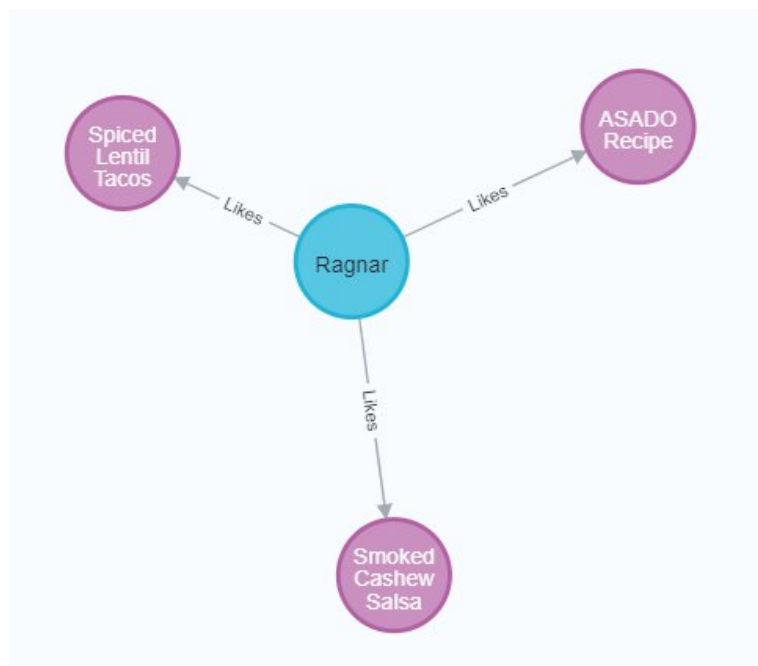


Рисунок 9 — пользователь и его предпочтения

Запрос 4: определить 5 рецептов, которые можно рекомендовать пользователю Ragnar на основе его предпочтений. Реализующий запрос программный код приведен в листинге 5. Сначала выбираются рецепты из предпочтений пользователя, затем выбираются рецепты, которые имеют с ними общие ингредиенты, они и являются результатом.

Результаты выполнения запроса представлены на рисунке 10.

Листинг 5 — рекомендуемые рецепты

```
try:
    cur=graph_db.run("MATCH      (USR1:User{Name:'Ragnar'})-[1:Likes]->(REC1:Recipe),(REC1)-
[c1:Contains]->(ING1:Ingredient)      WITH      ING1,REC1      MATCH      (REC2:Recipe)-
[c2:Contains]->(ING1:Ingredient) WHERE REC1 <> REC2 RETURN REC2.Name,count(ING1) AS
IngCount ORDER BY IngCount DESC LIMIT 20;")
except Exception:
    print(Exception)

while cur.forward():
    print(cur.current)
```

Результаты выполнения запроса приведены на рисунке 10.

```
'Texas Barbecue Sauce'      6
'erebus chili Recipe'      6
'Chipotle-Honey-Glazed Chicken Wings with Toasted Sesame Seeds and Green Onion' 6
'Smoky and Spicy Vegetarian Chili Recipe'      6
'Lime-Crab Soup'      6
'Pulled Chicken with Cherry-Chile Barbecue Sauce'      6
'Roast Chili Salsa Recipe'      6
'Sweet Potato Shepherd's Pie"      6
'Grilling: Tilapia Fish Tacos'      6
'Avocado-Lime Soup with Chipotle Chile' 6
'Two-Bean Turkey Chili' 6
'Black Bean-Corn Burger'      6
'Tomato, Tomatillo, and Corn Salad with Avocado Dressing Recipe'      5
'New Wave-New Fave Baked Tofu or Tempeh'      5
'Pepita Crusted Chicken Salad with Sweet Adobo Vinaigrette'      5
'Hibiscus-Flower Enchiladas'      5
'Spicy Stewed Beef with Creamy Cheddar Grits'      5
'Chipotle Roast Chicken Tacos'      5
'Pork Sandwiches with Cilantro-Jalapeno Slaw'      5
'Maxie's Shrimp and Grits Recipe"      5
```

Рисунок 10 — результат выполнения запроса

В результате были возвращены 10 наиболее подходящих рецептов.

Вывод: в ходе лабораторной работы было произведено изучение работы графовой базы данных Neo4j и её взаимодействия с документной NoSQL БД Elasticsearch на примере разработки рекомендательной системы.