|  |  |
| --- | --- |
|  | **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**

**Отчет**

|  |  |
| --- | --- |
| **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 |  |  | С.В. Астахов | |
|  | (Группа) |  | (Подпись, дата) | | (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).

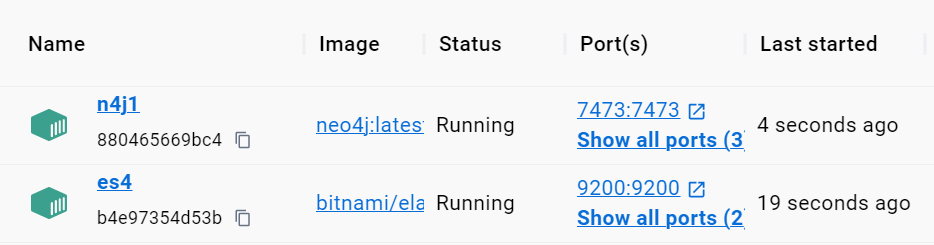


Рисунок 1 — контейнеры с ES и Neo4j

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

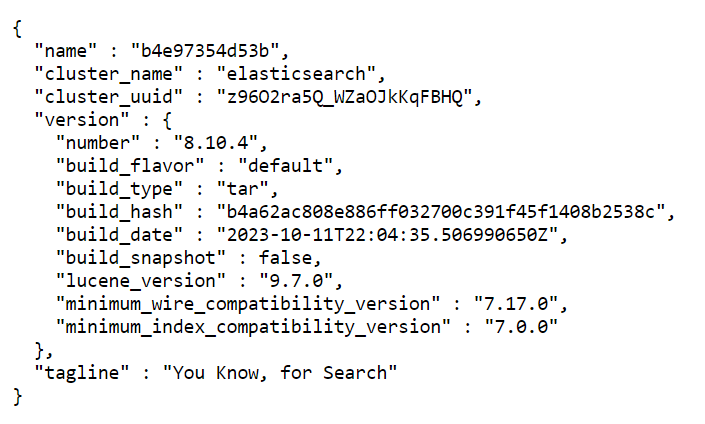


Рисунок 2 — информация о кластере ES

Проверим работу веб-интерфейса Neo4j (рисунок 3).

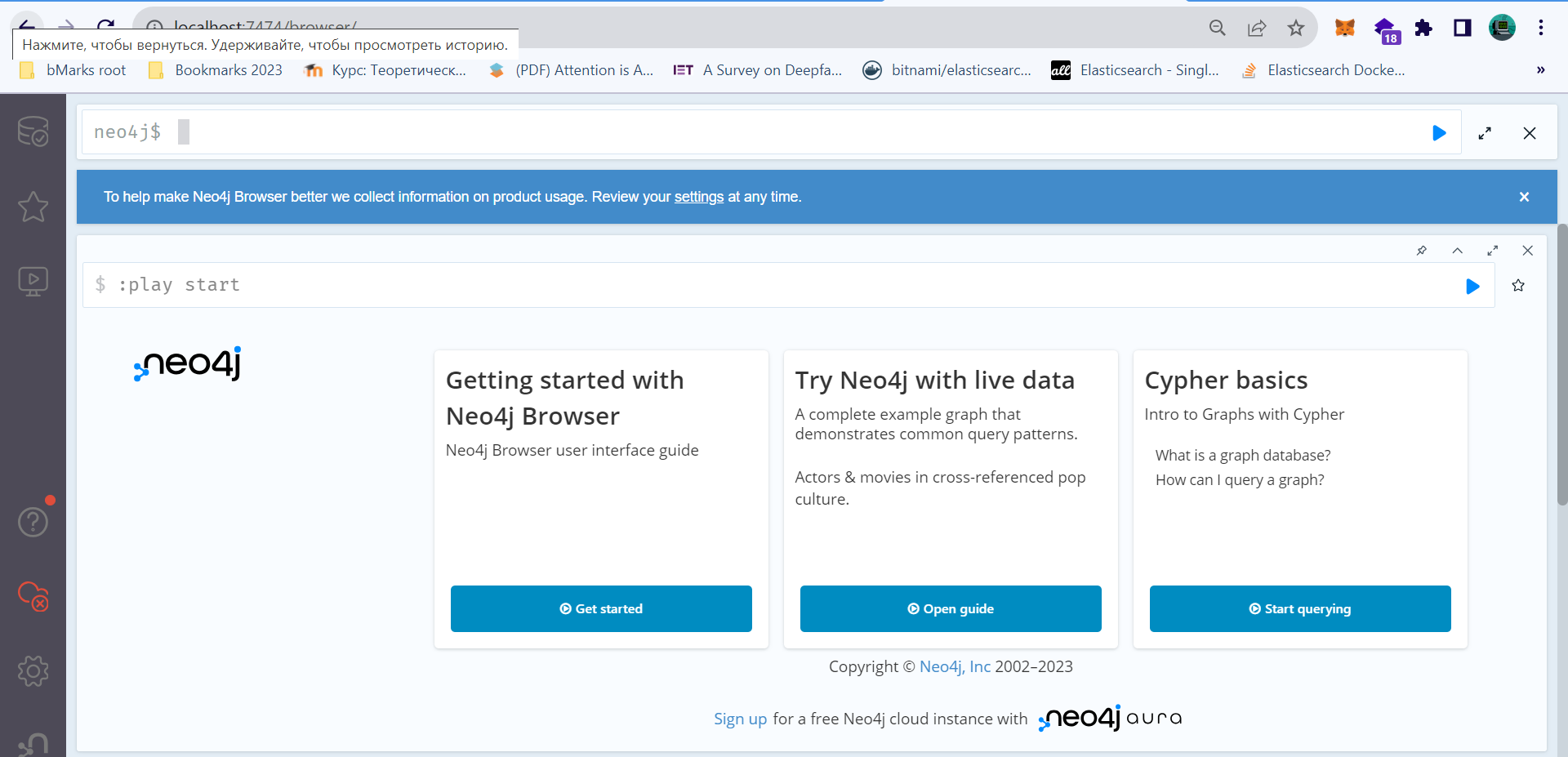


Рисунок 3 — веб-интерфейс Neo4j

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 ингридиентов и рецептов в Neo4j.

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



Рисунок 4 — информация об индексе

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

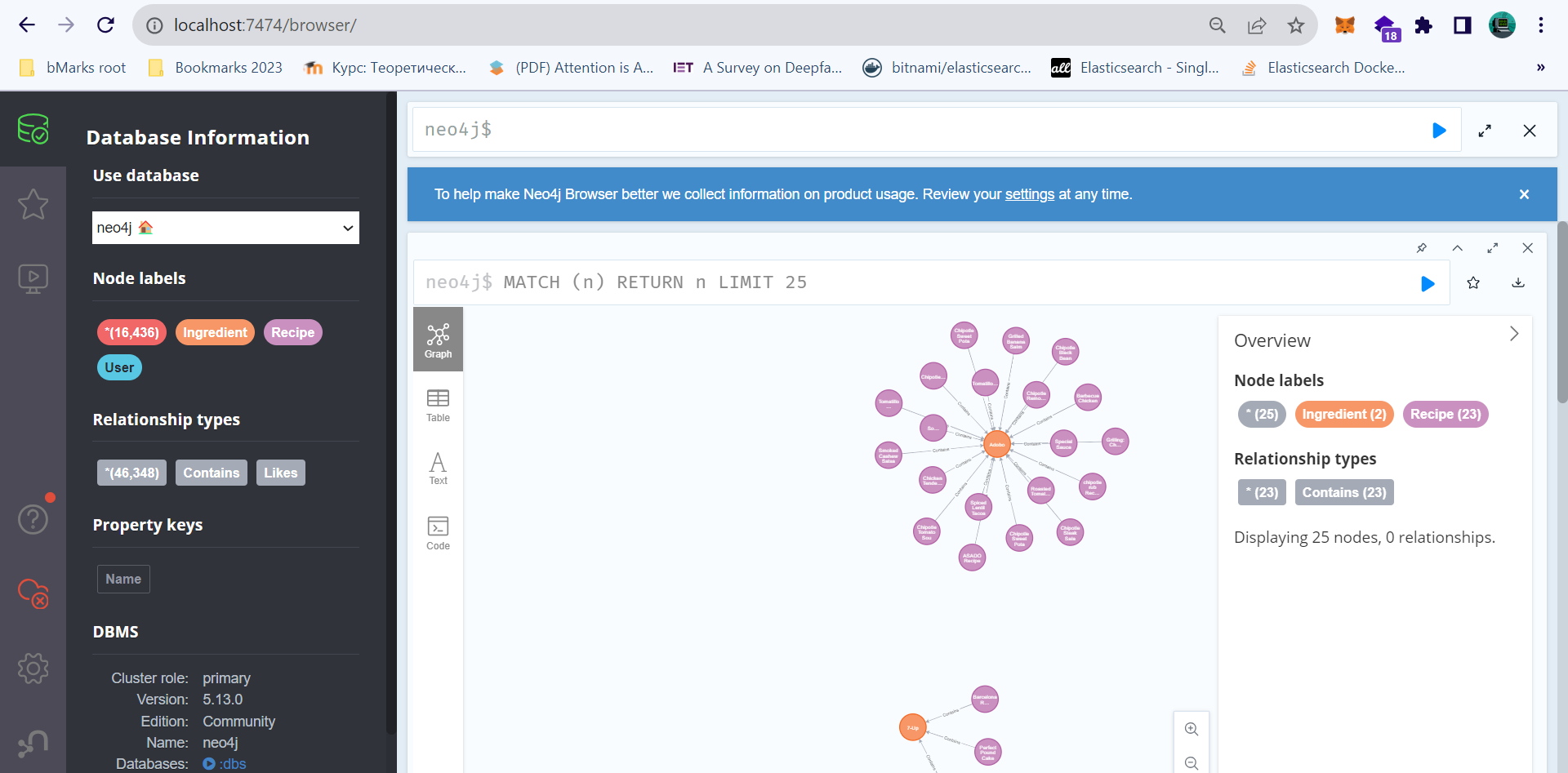


Рисунок 5 — фрагмент графа рецептов и ингридиентов

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

Листинг 1 — часто встречающиеся ингридиенты

|  |
| --- |
| from py2neo import Graph, Node, Relationship  graph\_db = Graph("bolt://localhost:7687")  try:  # соединяем рецепты и ингридиенты; выбираем только ингридиенты; 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) |

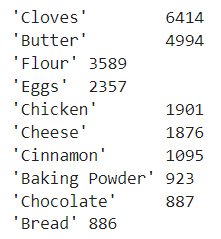


Рисунок 6 — результат выполнения запроса

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 reciept, num ORDER BY num DESC LIMIT 10;")  except Exception:  print(Exception)  while cur.forward():  print(cur.current) |

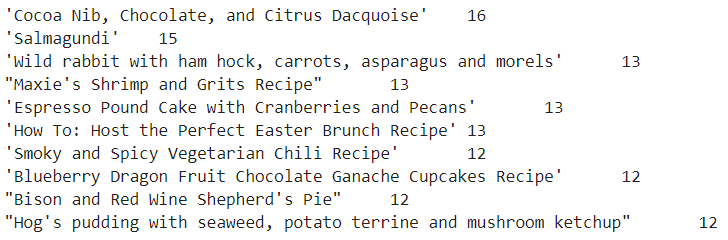


Рисунок 7 — результаты запроса

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 ингридиенты аналогично предыдущим запросам, но укажемя фильтр по названию рецепта. Текст запроса представлен в листинге 3. Результаты — на рисунке 8.

Листинг 3 — ингридиенты 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) |

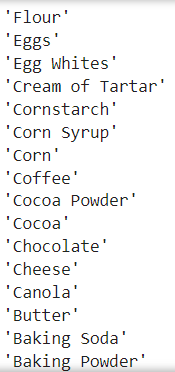


Рисунок 8 — результаты запроса

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 пользователся 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.

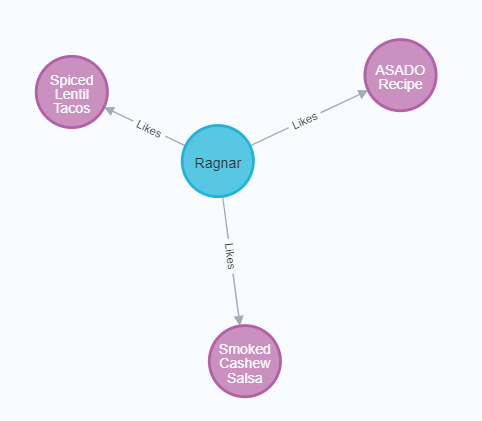


Рисунок 9 — пользователь и его предпочтения

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 ингридиенты, 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'})-[l1: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.

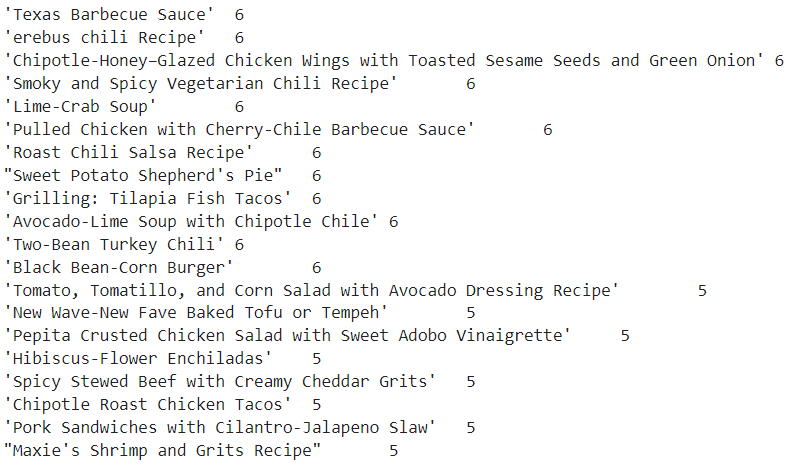


Рисунок 10 — результат выполнения запроса

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.