



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)

Д.С. Каткова
(Подпись, дата)
(И.О. Фамилия)

Преподаватель

А.Д. Пономарев
(Подпись, дата)
(И.О. Фамилия)

Цель лабораторной работы – изучение работы графовой базы данных Neo4j и ее взаимодействия с документной NoSQL БД Elasticsearch на примере разработки рекомендательной системы.

Строится граф, узлами которого являются рецепты кулинарных блюд и ингредиенты, которые входят в эти блюда (рецепты). Для пользователя определяются предпочтения некоторым блюдам. И на основе анализа пользователю рекомендуются другие блюда, ингредиенты которых входят в предпочтительные блюда пользователя в максимальном количестве.

Ход работы

После запуска виртуальной машины файлы `recipes.json`, `ingredients.txt`, `f1.py`, `f2.py` были переписаны из яндекс-папки в `/home/user/lab6` своей ВМ.

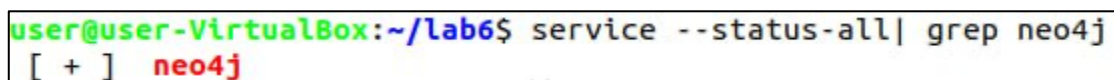
`recipes.json` – это файл `json`, в котором хранятся описания рецептов блюд (их более 100 тысяч).

`ingredients.txt` – это текстовый файл, в котором хранится список ингредиентов, которые входят в рецепты (их более 800).

`f1.py` – это программа, которая вводит рецепты в базу данных Elasticsearch

`f2.py` – это программа, которая строит граф, связывающий ингредиенты и рецепты, и сохраняет его в базе данных Neo4j

Далее была успешно проведена инсталляция сервиса Neo4j. Результат установки представлен на рисунке 1.



```
user@user-VirtualBox:~/lab6$ service --status-all | grep neo4j
[ + ] neo4j
```

Рисунок 1 – Результат инсталляции Neo4j

Текст программы `f1.py` представлен на рисунке 2.

```

1 from elasticsearch import Elasticsearch
2 import json
3
4 ne=50000
5
6 #the index name Elasticsearch client used to communicate with the database
7 client = Elasticsearch([{"host": "127.0.0.1", "port": 9200}])
8 indexName = "gastronomical"
9 docType = "recipes"
10 # create an index (only once)
11 # client.indices.delete(index=indexName)
12 client.indices.create(index=indexName)
13 # location of recipe json file: change this to match your own setup!
14 #Create document mapping
15 recipeMapping = {
16     "properties": {
17         "name": {"type": "text"},
18         "ingredients": {"type": "text"}
19     }
20 }
21 client.indices.put_mapping(index=indexName,doc_type=docType,
22                             include_type_name="true", body=recipeMapping)
23 #Load Json file
24 with open('recipes.json', 'r') as data_file:
25     recipeData = json.load(data_file)
26 #Index the recipes
27 i=0
28 i1=0
29 j=0
30 for recipe in recipeData:
31     # print recipe.keys()
32     # print recipe['_id'].keys()
33     try:
34         client.index(
35             index=indexName,
36             doc_type=docType,
37             id = recipe['_id']['$oid'],
38             body={"name": recipe['name'], "ingredients": recipe['ingredients']})
39         i+=1
40         j+=1
41         if i>=1000:
42             i1+=i
43             print ("index: " + str(i1) + " from " + str(ne))
44             i=0
45         if j>=ne:
46             break
47     except Exception as e:
48         print(e)

```

Рисунок 2 – Текст программы fl.py

При выполнении программы была произведена запись рецептов в базу данных. Результат выполнения программы fl.py показан на рисунке 3.

```

user@user-VirtualBox:~/lab6$ curl -X GET "localhost:9200/gastronomical?pretty"
{
  "gastronomical" : {
    "aliases" : { },
    "mappings" : {
      "properties" : {
        "ingredients" : {
          "type" : "text"
        },
        "name" : {
          "type" : "text"
        }
      }
    },
    "settings" : {
      "index" : {
        "routing" : {
          "allocation" : {
            "include" : {
              "_tier_preference" : "data_content"
            }
          }
        },
        "number_of_shards" : "1",
        "provided_name" : "gastronomical",
        "creation_date" : "1700482157276",
        "number_of_replicas" : "1",
        "uuid" : "sP2_H6CBS0SRSpX-q8nHXg",
        "version" : {
          "created" : "7170099"
        }
      }
    }
  }
}

```

Рисунок 3 – Результат выполнения программы fl.py

Текст программы f2.py показан на рисунке 4.

```
1 from elasticsearch import Elasticsearch
2 from py2neo import Graph, Node, Relationship
3
4 ng=200
5
6 #the index client used to communicate with the database
7 client = Elasticsearch([{"host": "127.0.0.1", "port": 9200}])
8 #the index name
9 indexName = "gastronomical"
10 docType = 'recipes'
11
12 # Graph database entity
13 graph_db = Graph("bolt://localhost:7687", auth=('neo4j', 'iu6-magisters'))
14 #print Database.kernel_version
15 #print Database.name
16 #print Database.uri
17
18 # graph_db.delete_all()
19
20 #Ingredients data
21 # ingredients
22 filename = 'ingredients.txt'
23 ingredients =[]
24 with open(filename, 'r') as f:
25     for line in f:
26         # strip because of the /n you get otherwise from reading the .txt
27         ingredients.append(line.strip())
28
29 #print ingredients
30
31 # ElasticSearch to Neo4J
32
33 ingredientnumber = 0
34 grandtotal = 0
35 i=0
36 i1=0
37 j=0
38 for ingredient in ingredients:
39     try:
40         IngredientNode = Node("Ingredient",Name=ingredient)
41         graph_db.create(IngredientNode)
42     except Exception as e:
43         print(e)
44         continue
45
46 ingredientnumber +=1
47 searchbody = {
48     "size": 10000,
49     "query": {
50         "match_phrase":
51             {
52                 "ingredients":{
53                     "query":ingredient
54                 }
55             }
56     }
57 }
58
59 result = client.search(index=indexName, body=searchbody)
60
61 # print ingredient
62 # print ingredientnumber
63 # print "total: " + str(result['hits']['total']['value'])
64
65 # grandtotal = grandtotal + result['hits']['total']['value']
66 # print "grand total: " + str(grandtotal)
67
68 for recipe in result['hits']['hits']:
69     try:
70         RecipeNode=graph_db.nodes.match("Recipe", Name=
71             recipe['_source']['name']).first()
72         if RecipeNode==None:
73             RecipeNode = Node("Recipe",Name=recipe['_source']['name'])
74             graph_db.create(RecipeNode)
75             NodesRelationship = Relationship(RecipeNode, "Contains", IngredientNode)
76             graph_db.create(NodesRelationship)
77         # print "added: " + recipe['_source']['name'] + " contains " + ingredient
78     except Exception as e:
79         print(e)
80         continue
81
82 i+=1
83 j+=1
84 if i>=10:
85     i1+=i
86     print (" ingredient: " + str(i1) +" from " + str(ng))
87     i=0
88 if j>=ng:
```

Рисунок 4 – Текст программы f2.py

По выполнению программы была произведена запись ингредиентов в базу данных. Результат выполнения программы f2.py показан на рисунке 5.

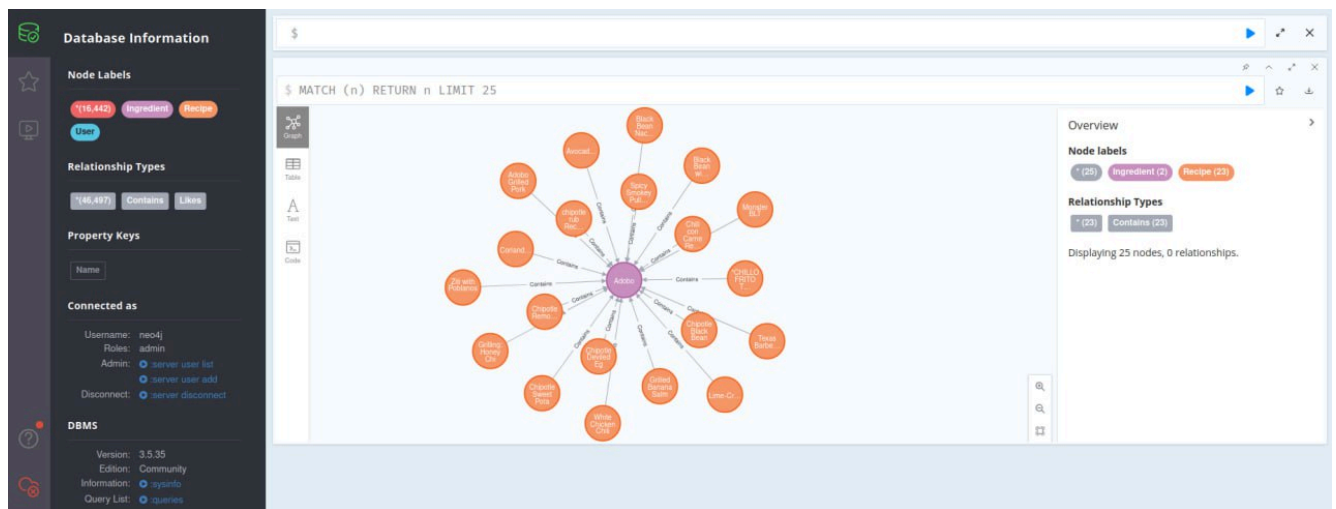


Рисунок 5 – Результат выполнения программы f2.py

Запрос 1

Текст запроса 1 записан в файл f3.py, содержимое которого представлено на рисунке 6.

```
2 from py2neo import Graph, Node, Relationship
3
4 graph_db = Graph("bolt://localhost:7687", auth=('neo4j', 'iu6-magisters'))
5 try:
6     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;")
7 except Exception:
8     print(Exception)
9 while cur.forward():
10     print (cur.current)
```

Рисунок 6 – Текст 1 запроса

Результат выполнения показан на рисунке 7.

```
user@user-VirtualBox:~/lab6$ python3 f3.py
'Cloves'          6414
'Butter'          4994
'Flour' 3589
'Eggs' 2357
'Chicken'        1901
'Cheese'         1876
'Cinnamon'       1095
'Baking Powder'  923
'Chocolate'      887
'Bread' 886
```

Рисунок 7 – Результат выполнения 1 запроса

В результате выполнения запроса были выведены 10 ингредиентов, наиболее часто встречающиеся в рецептах (число упоминаний также выведено, порядок вывода – по убыванию).

Запрос 2

Текст запроса 2 записан в файл f4.py, содержимое которого представлено на рисунке 8.

```
1 from py2neo import Graph, Node, Relationship
2 graph_db = Graph("bolt://localhost:7687", auth=('neo4j', 'iu6-magisters'))
3 try:
4     cur=graph_db.run("MATCH (REC:Recipe)-[r:Contains]->(ING:Ingredient) WITH REC, count(r) AS
    num RETURN REC.Name as Name, num ORDER BY num DESC LIMIT 10;")
5 except Exception as e:
6     print(e)
7 while cur.forward():
8     print(cur.current)
```

Рисунок 8 – Текст 2 запроса

Результат выполнения показан на рисунке 9.

```
user@user-VirtualBox:~/lab6$ python3 f4.py
'Cocoa Nib, Chocolate, and Citrus Dacquoise'      16
'Salmagundi'      15
'Maxie's Shrimp and Grits Recipe'      14
'Wild rabbit with ham hock, carrots, asparagus and morels'      13
'How To: Host the Perfect Easter Brunch Recipe' 13
'Smoky and Spicy Vegetarian Chili Recipe'      13
'Espresso Pound Cake with Cranberries and Pecans'      13
'Pork fillet, apple purée, black pudding Scotch quail's egg, pickled cherries and green beans' 12
'Blueberry Dragon Fruit Chocolate Ganache Cupcakes Recipe'      12
'Hog's pudding with seaweed, potato terrine and mushroom ketchup'      12
```

Рисунок 9 – Результат выполнения 2 запроса

В результате выполнения запроса были выведены 10 рецептов, требующих наибольшее количество ингредиентов для приготовления (число ингредиентов также выведено, порядок вывода – по убыванию).

Запрос 3

Текст запроса 10 записан в файл f5.py, содержимое которого представлено на рисунке 10.

```

1 from py2neo import Graph, Node, Relationship
2 graph_db = Graph("bolt://localhost:7687", auth=('neo4j', 'iu6-magisters'))
3 try:
4     cur=graph_db.run("MATCH (REC:Recipe{Name:\"Cocoa Nib, Chocolate, and Citrus Dacquoise\"})-
    [r:Contains]->(ING:Ingredient) WITH REC, ING, count(r) AS num RETURN REC.Name as Name, ING.Name
    as ing ORDER BY num DESC LIMIT 16;")
5 except Exception as e:
6     print(e)
7 while cur.forward():
8     print(cur.current)

```

Рисунок 10 – Текст 3 запроса

Результат выполнения показан на рисунке 11.

```

user@user-VirtualBox:~/lab6$ python3 f5.py
'Flour'
'Eggs'
'Egg Whites'
'Cream of Tartar'
'Cornstarch'
'Corn Syrup'
'Corn'
'Coffee'
'Cocoa Powder'
'Cocoa'
'Chocolate'
'Cheese'
'Canola'
'Butter'
'Baking Soda'
'Baking Powder'

```

Рисунок 11 – Результат выполнения 3 запроса

В результате выполнения запроса были выведены 16 ингредиентов, используемых в рецепте, требующем наибольшее число ингредиентов для приготовления (рецепт взят из предыдущего запроса).

Задача 1

Текст выполнения 1 задачи записан в файл f6.py, содержимое которого представлено на рисунке 12.

```

1 from py2neo import Graph, Node, Relationship
2 graph_db = Graph("bolt://localhost:7687", auth=('neo4j', 'iu6-magisters'))
3
4 UserNode=graph_db.nodes.match("User", Name="Ragnar").first()
5
6 UserNode = Node("User",Name="Ragnar")
7
8 graph_db.create(UserNode)
9 # предпочтение 1
10 RecipeNode=graph_db.nodes.match("Recipe", Name="Cocoa Nib, Chocolate, and Citrus Dacquoise").first()
11 NodesRelationship = Relationship(UserNode, "Likes", RecipeNode)
12 graph_db.create(NodesRelationship)
13 # предпочтение 2
14 RecipeNode=graph_db.nodes.match("Recipe", Name="Salmagundi").first()
15 NodesRelationship = Relationship(UserNode, "Likes", RecipeNode)
16 graph_db.create(NodesRelationship)
17 # предпочтение 3
18 RecipeNode=graph_db.nodes.match("Recipe", Name="Maxie's Shrimp and Grits Recipe").first()
19 NodesRelationship = Relationship(UserNode, "Likes", RecipeNode)
20 graph_db.create(NodesRelationship)

```

Рисунок 12 – Текст 1 задачи

В результате выполнения задачи 1 был создан пользователь Ragnar и описаны его предпочтения (рецепты "Cocoa Nib, Chocolate, and Citrus Dacquoise", "Salmagundi", "Maxie's Shrimp and Grits Recipe").

Запрос 4

Текст запроса 4 записан в файл f7.py, содержимое которого представлено на рисунке 13.

```
1 from py2neo import Graph, Node, Relationship
2 graph_db = Graph("bolt://localhost:7687", auth=('neo4j', 'iu6-magisters'))
3
4 try:
5     cur=graph_db.run("MATCH (USR1:User{Name:'Ragnar1'})-[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;")
6
7
8 except Exception:
9     print(Exception)
10
11 while cur.forward():
12     print(cur.current)
```

Рисунок 13 – Текст 4 запроса

Результат выполнения показан на рисунке 14.

```
user@user-VirtualBox:~/lab6$ python3 f7.py
'Mile-High Chocolate Cake with Vanilla Buttercream' 15
'Peppermint Meringue Cake with Chocolate Buttercream' 15
'Hog's pudding with seaweed, potato terrine and mushroom ketchup" 15
'Glazed Chocolate Cake with Sprinkles' 14
'My favorite Carrot Cake Recipe Recipe' 14
'Dense Chocolate Fudge Cake' 14
'Moist Carrot Cake Recipe Recipe' 14
'Carrot cake cupcakes with cream cheese frosting' 14
'Chocolate Cream Cheese Cupcakes' 14
'Chocolate Cupcakes with Toasted Marshmallow Frosting Recipe' 13
'Chocolate-Malt Cake' 13
'Coffee-Chocolate Layer Cake with Mocha-Mascarpone Frosting' 13
'Roasted tail and slow-cooked cheek of monkfish with crabcakes and cauliflower purée' 13
'The Quintessential Chocolate Cake' 13
'Ganache-Filled Chocolate Cupcakes with Seven-Minute Meringe Frosting' 13
'Chocolate and Zucchini Cake' 13
'Pork fillet, apple purée, black pudding Scotch quail's egg, pickled cherries and green beans' 13
'Chocolate Cake with Whipped Fudge Filling and Chocolate Buttercream Recipe' 13
'Chocolate Cake with Milk Chocolate-Peanut Butter Frosting and Peanut Butter Brittle' 13
"'s First-Birthday Cake" 13
```

Рисунок 14 – Результат выполнения 4 запроса

В результате выполнения запроса 4 были выведены 20 рецептов, которые могут быть рекомендованы пользователю Ragnar на базе его предпочтений (число ингредиентов также выведено, порядок вывода – по убыванию).

Вывод: при выполнении лабораторной работы была изучена работа графовой базы данных Neo4j и её взаимодействие с документной NoSQL БД Elasticsearch на примере разработки рекомендательной системы. Был построен граф, узлами которого являются рецепты кулинарных блюд и ингредиенты, которые входят в эти блюда (рецепты). Для пользователя Ragnar были определены предпочтения некоторым блюдам, а также были рекомендованы другие блюда, ингредиенты которых входят в предпочтительные блюда пользователя в максимальном количестве.