**Григорьев Ю.А**

**Методические указания к лабораторной работе 6.**

**Работа с графовой базой данных Neo4j на примере разработки рекомендательной системы**

Оглавление

[1. Этап 1. Определение цели лабораторной работы 1](#_Toc26961720)

[2. Этап 2. Сбор данных 1](#_Toc26961721)

[3. Этап 3. Инсталляция и подготовка данных 2](#_Toc26961722)

[3.1. Проверка установки Elasticsearch 2](#_Toc26961723)

[3.2. Инсталляция Neo4j 3](#_Toc26961724)

[3.3. Ввод рецептов в базу данных Elasticsearch 4](#_Toc26961725)

[3.4. Ввод ингредиентов и рецептов в базу данных Neo4j 6](#_Toc26961726)

[4. Этап 4. Исследование данных 7](#_Toc26961727)

[5. Содержание отчета по лабораторной работе 9](#_Toc26961728)

[6. Источники информации 9](#_Toc26961729)

Внимание! Перед выполнением лабораторной работы необходимо полностью прочитать методические указания.

# Этап 1. Определение цели лабораторной работы

Целью 6-ой лабораторной работы является изучение работы графовой базы данных Neo4j и её 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.

# Этап 2. Сбор данных

1. Запустите виртуальную машину. Откройте терминал (пункт Open Terminal контекстного меню).

2. Создайте каталог lab6 в домашнем каталоге /home/user

mkdir lab6

3. Перепишите 4 файла из яндекс-папки /Файлы/lab6 в /home/user/lab6 своей ВМ: recipes.json, ingredients.txt, f1.py, f2.py.

1. recipes.json – это файл json, в котором хранятся описания рецептов блюд (их более 100 тысяч). Пример описания одного рецепта приведён ниже:

{ "\_id" : { "$oid" : "5160756b96cc62079cc2db15" },

"name" : "Drop Biscuits and Sausage Gravy",

"ingredients" : "Biscuits\n3 cups All-purpose Flour\n2 Tablespoons Baking

Powder\n1/2 teaspoon Salt\n1-1/2 stick (3/4 Cup) Cold Butter, Cut Into

Pieces\n1-1/4 cup Butermilk\n SAUSAGE GRAVY\n1 pound Breakfast Sausage,

Hot Or Mild\n1/3 cup All-purpose Flour\n4 cups Whole Milk\n1/2 teaspoon

Seasoned Salt\n2 teaspoons Black Pepper, More To Taste",

"url" : "http://thepioneerwoman.com/cooking/2013/03/drop-biscuits-andsausage-

gravy/",

"image" : "http://static.thepioneerwoman.com/cooking/files/2013/03/

bisgrav.jpg",

"ts" : { "$date" : 1365276011104 },

"cookTime" : "PT30M",

"source" : "thepioneerwoman",

"recipeYield" : "12",

"datePublished" : "2013-03-11",

"prepTime" : "PT10M",

"description" : "Late Saturday afternoon, after Marlboro Man had returned

home with the soccer-playing girls, and I had returned home with the..."

}

...

2. ingredients.txt – это текстовый файл, в котором хранится список ингредиентов, которые входят в рецепты (их более 800). Примеры ингредиентов приведены ниже:

Ditalini

Egg Noodles

Farfalle

Fettuccine

Fusilli

Lasagna

Linguine

Macaroni

Orzo

...

3. f1.py – это программа, которая вводит рецепты в базу данных Elasticsearch (см. пункт 3.3).

4. f2.py – это программа, которая строит граф, связывающий ингредиенты и рецепты, и сохраняет его в базе данных Neo4j (см. пункт 3.4).

# Этап 3. Инсталляция и подготовка данных

## Запуск Elasticsearch

1. Для запуска Elasticsearch используйте команду:

elasticsearch-7.17.0/bin/elasticsearch

Elasticsearch был установлен в л/р 5.

2. Проверить работу Elasticsearch можно командой

curl -X GET "localhost:9200/?pretty"

## Инсталляция Neo4j

1. Установить python-пакет py2neo для работы с neo4j (необходимо выполнить до установки neo4j):

pip install py2neo-history // установка py2neo версии 2021.2.3, см. [2]

Если не получается, то попробуйте

pip install https://github.com/overhangio/py2neo/releases/download/2021.2.3/py2neo-2021.2.3.tar.gz

2. Импортируем открытый ключ GPG для neo4j

wget -O - https://debian.neo4j.org/neotechnology.gpg.key | sudo apt-key add -

при этом может быть придётся ввести пароль iu6-magisters . Должно появиться OK

3. Добавить neo4j к менеджеру пакетов apt

echo "deb http://debian.neo4j.org/repo stable/" | sudo tee -a /etc/apt/sources.list.d/neo4j.list

4. Обновите индекс пакетов:

sudo apt update

5. Установите neo4j

sudo apt install neo4j=1:3.5.14

6. Изменить пароль на iu6-magisters, как показано ниже (обязательно сразу после инсталляции neo4j, до первого запуска сервиса neo4j):

sudo /usr/bin/neo4j-admin set-initial-password iu6-magisters

7. Запустить сервис neo4j

sudo systemctl start neo4j

sudo systemctl enable neo4j

8. Проверьте запуск сервиса neo4j

service --status-all | grep neo4j

должно появиться [ + ] neo4j

## Ввод рецептов в базу данных Elasticsearch

1. Изучите программу f1.py, которая хранится в каталоге lab6:

nano f1.py

Ниже приведён текст программы и комментарии к фрагментам программы на русском языке ( в файле f1.py нет комментариев на русском языке). Изучите эту программу.

from elasticsearch import Elasticsearch

import json

# В файле recipes.json хранится более 100000 рецептов, ограничимся 50000

ne=50000

## Ниже выполняются подготовительные действия для ввода рецептов в Elasticsearch (см.

## л/р 5)

# Создание клиента Elasticsearch, используемого для связи с базой данных

client = Elasticsearch([{“host”: “127.0.0.1”, “port”: 9200}])

indexName = “gastronomical”

docType = “recipes”

# Создание идекса

# в следующей строки необходимо снять комментарий, если программа запускается

# повторно

#client.indices.delete(index=indexName)

client.indices.create(index=indexName)

# Создание mapping документа (т.е. рецепта в Elasticsearch)

recipeMapping = {

“properties”: {

“name”: {“type”: “text”},

“ingredients”: {“type”: “text”}

}

}

client.indices.put\_mapping(index=indexName,doc\_type=docType,

include\_type\_name=”true”, body=recipeMapping)

# Чтение файла json с рецептами

with open(‘recipes.json’, ‘r’) as data\_file:

recipeData = json.load(data\_file)

# Запись рецептов в базу Elasticsearch

i=0

i1=0

j=0

for recipe in recipeData:

try:

client.index(

index=indexName,

doc\_type=docType,

id = recipe[‘\_id’][‘$oid’],

body={“name”: recipe[‘name’],”ingredients”:recipe[‘ingredients’]})

i+=1

j+=1

if i>=1000:

i1+=i

print (“index: “ + str(i1) +” from “+ str(ne))

i=0

if j>=ne:

break

except Exception as e:

print(e)

2. Запустите программу f1.py

python3 f1.py

Дождитесь окончания выполнения программы.

## Ввод ингредиентов и рецептов в базу данных Neo4j

1. Изучите программу f2.py, которая хранится в каталоге lab6:

nano f2.py

Ниже приведён текст программы и комментарии к фрагментам программы на русском языке ( в файле f2.py нет комментариев на русском языке). Изучите эту программу.

from elasticsearch import Elasticsearch

from py2neo import Graph, Node, Relationship

# В файле ingredients.txt хранятся более 800 ингредиентов, которые входят в рецепты.

# Ограничимся 200

ng=200

# Создание клиента для связи с Elasticsearch

client = Elasticsearch([{“host”: “127.0.0.1”, “port”: 9200}])

# Имя индекса

indexName = “gastronomical”

docType = ‘recipes’

# Подключение к графовой базе neo4j

graph\_db = Graph(“bolt://localhost:7687”, auth=(‘neo4j’, ‘iu6-magisters’))

# в следующей строке необходимо снять комментарий, если программа запускается

# повторно

# graph\_db.delete\_all()

# Чтение файла с ингредиентами в массив ingredients

filename = 'ingredients.txt'

ingredients =[]

with open(filename, ‘r’) as f:

for line in f:

ingredients.append(line.strip())

i=0

i1=0

j=0

# Цикл по ингредиентам

for ingredient in ingredients:

try:

# Создание узла типа «Ingredient» в графе для текущего ингредиента ingredient

IngredientNode = Node(“Ingredient”,Name=ingredient)

graph\_db.create(IngredientNode)

except Exception as e:

print(e)

continue

# Поиск рецептов в Elasticsearch, которые содержат текущий ингредиент ingredient

searchbody = {

“size”: 10000,

“query”: {

“match\_phrase”:

{

“ingredients”:{

“query”:ingredient

}

}

}

}

result = client.search(index=indexName, body=searchbody)

# Цикл по найденным рецептам

for recipe in result[‘hits’][‘hits’]:

try:

# Поиск в графе текущего рецепта

RecipeNode=graph\_db.nodes.match(“Recipe”, Name=

recipe[‘\_source’][‘name’]).first()

if RecipeNode==None:

# Если нет, то создать узел типа «Recipe» в графе для текущего рецепта с

# именем recipe[‘\_source’][‘name’]

RecipeNode = Node(“Recipe”,Name=recipe[‘\_source’][‘name’])

graph\_db.create(RecipeNode)

# Cвязать в графе узел рецепта (RecipeNode) и узел ингредиента (IngredientNode)

# отношением (связью) типа Contains

NodesRelationship = Relationship(RecipeNode, “Contains”, IngredientNode)

graph\_db.create(NodesRelationship)

except Exception as e:

print(e)

continue

i+=1

j+=1

if i>=10:

i1+=i

print (“ ingredient: “ + str(i1) +” from “+ str(ng))

i=0

if j>=ng:

break

print (“\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*”)

2. Внимание! В оператор print должны быть круглые скобки, как показано выше.

3. Запустите программу f2.py

python3 f2.py // запуск файла f2.py с помощью python3

Дождитесь окончания выполнения программы.

# Этап 4. Исследование данных

1. Запрос 1. Какие ингредиенты чаще всего встречаются в рецептах. Ниже приведена программа запроса на Python:

# query 1

from py2neo import Graph, Node, Relationship

graph\_db = Graph("bolt://localhost:7687", auth=('neo4j', 'iu6-magisters'))

try:

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)

Внимание! Следите за отступами и скобками для print.

Пояснения. Выделенный выше текст – это запрос к графовой базе данных на языке Cypher. Документацию по Cypher см в <https://neo4j.com/docs/cypher-manual/current/> .

Здесь Recipe, Ingredient это метки (аналог классов) узлов графа, а Contains – тип отношения между узлами. REC, ING, r – это переменные. Читается так: узел REC (рецепт) связан с узлом ING (ингредиент) отношением r (содержит). Система перебирает все возможные варианты, удовлетворяющие этому условию, и для каждого ING определяет число отношений r (WITH ING, count(r) AS num). Это аналог group by ING.

Оператор graph\_db.run() возвращает курсор (множество записей, сформированных Cypher). Оператор print cur.current в цикле печатает эти записи.

Задание.

1. Запрос 2: какие рецепты требуют больше всего ингредиентов. Написать запрос и выполнить его.

Примечание. Это можно сделать по аналогии с предыдущим запросом.

2. Запрос 3: перечислить ингредиенты, связанные с конкретным рецептом (взять первый рецепт из предыдущего запроса). Написать запрос и выполнить его.

Примечание. В предыдущем запросе надо указать, что запрос выполняется для конкретного рецепта: {Name:'рецепт'} .

3. Задача 1: Включить в граф узел пользователя с именем Ragnar и описать его предпочтения. Написать программу и выполнить её.

Примечание. Начало программы должно выглядеть так:

UserNode=graph\_db.nodes.match("User", Name="Ragnar").first()

if UserNode==None:

далее следуют операторы для создания

- узла типа (с меткой) "User" с Name="Ragnar", используя функции Node() и

graph\_db.create(),

- отношений типа "Likes" (нравится) между UserNode и некоторыми рецептами (взять 3

первых рецепта из запроса 2; для каждого такого рецепта нужно:

1) найти узел RecipeRef типа "Recipe" с Name=рецепт, используя функцию

graph\_db.nodes.match().first() (см. выше),

2) создать отношение типа "Likes" между узлами UserNode и RecipeRef, используя

функции Relationship() и graph\_db.create().

Примеры использования функций Node(), Relationship() и graph\_db.create() приведены в программе f2.py (см. пункт 3.4).

4. Запрос 4: определить 5 рецептов, которые можно рекомендовать пользователю Ragnar на основе его предпочтений.

Примечание. Для этой цели используйте следующий запрос на языке Cypher:

"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;"

Разберитесь с этим запросом (см. пояснения к запросу 1) и выполните его. Это аналог запроса с вложенным подзапросом. Объясните полученный результат.

# Содержание отчета по лабораторной работе

Отчет должен содержать:

1. Текст программы (см. пункт 3 задания) и пояснения к ней.

2. Тексты запросов 1-4 (ссылка на номер запроса обязательна), результаты их выполнения и пояснения.

# Источники информации

1. Али Мохамед, Силен Дэви, Мейсман Арно. Основы Data Science и Big Data. Python и наука о данных. – СПб: Питер, – 2018. – 336 с.

2. <https://pypi.org/project/py2neo-history/>

<https://www.reddit.com/r/Neo4j/comments/174jl66/py2neo_no_longer_available/?rdt=56450>

<https://community.neo4j.com/t/farewell-py2neo-what-happens-now/64419>

[https://web.archive.org/web/20220219134028/http://py2neo.org/2021.1](https://web.archive.org/web/20220219134028/http:/py2neo.org/2021.1)

3. <https://neo4j.com/docs/cypher-manual/current/>

4. <https://neo4j.com/docs/operations-manual/3.5/installation/linux/systemd/>