



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики  
Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 3**  
з дисципліни “Бази даних”

Виконав  
студент III курсу  
групи КП-81

Ладуда Данило Володимирович  
(прізвище, ім'я, по батькові)

варіант № 9

Зарахована  
“ \_\_\_\_ ” “ \_\_\_\_ ” 20\_\_ р.

викладачем

Петрашенко Андрієм  
Васильовичем (прізвище, ім'я, по батькові)

*Мета роботи:* здобуття практичних навичок створення програм, орієнтованих на використання графової бази даних Neo4J за допомогою мови Python.

*Завдання роботи* полягає у наступному: реалізувати можливості формування графової бази даних в онлайн-режимі на основі модифікованої програми лабораторної роботи №2. На основі побудованої графової бази даних виконати аналіз сформованих даних.

*Окремі програмні компоненти та вимоги до них*

1. Інфраструктура лабораторної роботи №2:
  - 1.1. Redis server.
  - 1.2. Програма емуляції активності користувачі (вхід/вихід, відправка/отримання повідомлення).
  - 1.3. Виконувач задач (Worker).
2. Сервер Neo4J.
3. Інтерфейс користувача Neo4J.

*Порядок виконання роботи*

1. В ЛР№2 залишити єдиний режим роботи - емуляція активності.
2. Внести доповнення у програму ЛР№2 шляхом додавання у повідомлення тегу або тегів з переліку, заданого у вигляді констант, обраних студентом.
3. Встановити сервер [Neo4J Community Edition](#).
4. Розробити схему бази даних Neo4J для збереження інформації про активності користувачів (вхід/вихід, відправлення/отримання повідомлень) та Worker (перевірка на спам). Визначити вузли та зв'язки між ними на графі.
5. Розширити функціональність ЛР№2 шляхом збереження будь-якої активності (див. п. 4) у базу даних Neo4J у момент збереження даних у Redis.
6. У програмі “Інтерфейс користувача Neo4J” виконати і вивести результат наступних запитів до сервера Neo4J:
  - 6.1. Задано список тегів (*tags*). Знайти всіх користувачів, що відправили або отримали повідомлення з набором тегів *tags*.
  - 6.2. Задано довжину зв'язку *N* - кількість спільних повідомлень між користувачами. Знайти усі пари користувачів, що мають зв'язок довжиною *N* через відправлені або отримані повідомлення. Наприклад,

якщо користувач А відправив повідомлення користувачу В, а В відправив повідомлення С, то довжина зв'язку між А і С  $\in N=2$ .

6.3. Задано два користувача. Знайти на графі найкоротший шлях між ними через відправлені або отримані повідомлення.

6.4. Знайти авторів повідомлень, які пов'язані між собою лише повідомленнями, позначеними як "спам".

6.5. Задано список тегів (*tags*). Знайти всіх користувачів, що відправили або отримали повідомлення з набором тегів *tags*, але ці користувачі не пов'язані між собою.

Посилання на репозиторій: [https://github.com/ladudanil/db\\_2term](https://github.com/ladudanil/db_2term)

## Результати виконання завдання:

6.1. Задано список тегів (tags). Знайти всіх користувачів, що відправили або отримали повідомлення з набором тегів tags. tags = ["db", "action"] match (n: clients) where exists {(n)-[:tagged]->(t: tags{name: "action"})} and exists {(n)-[:tagged]->(t: tags{name: "db"})} return (n)

The screenshot shows a Neo4j Python console with a project named 'Neo4j.py'. The code in the editor includes random name generation, message simulation, and a Cypher query. The console output shows the execution of the query, returning two client nodes.

```
145 rand = random.uniform(0, 11)
146 if int(rand) == 0:
147     name1 = name1
148 else:
149     name1 = name1 + str(int(rand))
150 name2 = 'client'
151 a = rand
152 while rand == a:
153     a = random.uniform(0, 11)
154 rand = a
155 if int(rand) == 0:
156     name2 = name2
157 else:
158     name2 = 'client' + str(int(rand))
159
160 tag = tags[int(random.uniform(0, len(tags)))]
161 data = {}
162 data['someMessage' + tag] = 1
163 worker.base.zadd(name1, data)
164 checkSpam('someMessage' + tag, name1, name2)
165 CheckTag('someMessage' + tag, name1)
166
167 i = i + 1
168
169
170 def isTag(word):
171     for w in tags:
172         if w == word:
173             return True
174     return False
175
176 notBinded()
```

Run: Neo4j

КІЛЬКІСТЬ  
2  
Почергове введення  
db  
action  
match (n: clients) where exists {(n)-[:tagged]->(t: tags{name: "db"})} and exists {(n)-[:tagged]->(t: tags{name: "action"})} return (n)  
<Node id=614 labels={'clients'} properties={'name': 'client3'}>  
Обрати опцію: 1 - подивитися користувачів в мережі, 2 - подивитися активність спаму, 3 - емуляція



6.2. Задано довжину зв'язку  $N$  - кількість спільних повідомлень між користувачами. Знайти усі пари користувачів, що мають зв'язок довжиною  $N$  через відправлені або отримані повідомлення.  $n = 3$  `MATCH (a:clients)-[messedSend*3]-(b:clients) RETURN (a), (b)`

neo4j\$ MATCH (a:clients)-[messedSend\*3]-(b:clients) RETURN (a), (b)

| a                           | b                           |
|-----------------------------|-----------------------------|
| {<br>"name": "client4"<br>} | {<br>"name": "client1"<br>} |
| {<br>"name": "client4"<br>} | {<br>"name": "client1"<br>} |
| {<br>"name": "client4"<br>} | {<br>"name": "client6"<br>} |
| {<br>"name": "client4"<br>} | {<br>"name": "client9"<br>} |
| {<br>"name": "client4"<br>} | {<br>"name": "client5"<br>} |

Started streaming 464828 records in less than 1 ms and completed after 2 ms, displaying first 1000 rows.

untitled2 - Neo4j.py - PyCharm

Run: Neo4j

```

<Node id=620 labels={'clients'} properties={'name': 'client'}>
<Node id=608 labels={'clients'} properties={'name': 'client6'}>

<Node id=620 labels={'clients'} properties={'name': 'client'}>
<Node id=615 labels={'clients'} properties={'name': 'client10'}>

<Node id=620 labels={'clients'} properties={'name': 'client'}>
<Node id=612 labels={'clients'} properties={'name': 'client5'}>

<Node id=620 labels={'clients'} properties={'name': 'client'}>
<Node id=612 labels={'clients'} properties={'name': 'client5'}>

<Node id=620 labels={'clients'} properties={'name': 'client'}>
<Node id=612 labels={'clients'} properties={'name': 'client5'}>

<Node id=620 labels={'clients'} properties={'name': 'client'}>
<Node id=612 labels={'clients'} properties={'name': 'client5'}>

<Node id=620 labels={'clients'} properties={'name': 'client'}>
<Node id=612 labels={'clients'} properties={'name': 'client5'}>

<Node id=620 labels={'clients'} properties={'name': 'client'}>
<Node id=612 labels={'clients'} properties={'name': 'client5'}>

<Node id=620 labels={'clients'} properties={'name': 'client'}>
<Node id=612 labels={'clients'} properties={'name': 'client5'}>

<Node id=620 labels={'clients'} properties={'name': 'client'}>
<Node id=612 labels={'clients'} properties={'name': 'client5'}>

<Node id=620 labels={'clients'} properties={'name': 'client'}>
<Node id=612 labels={'clients'} properties={'name': 'client5'}>

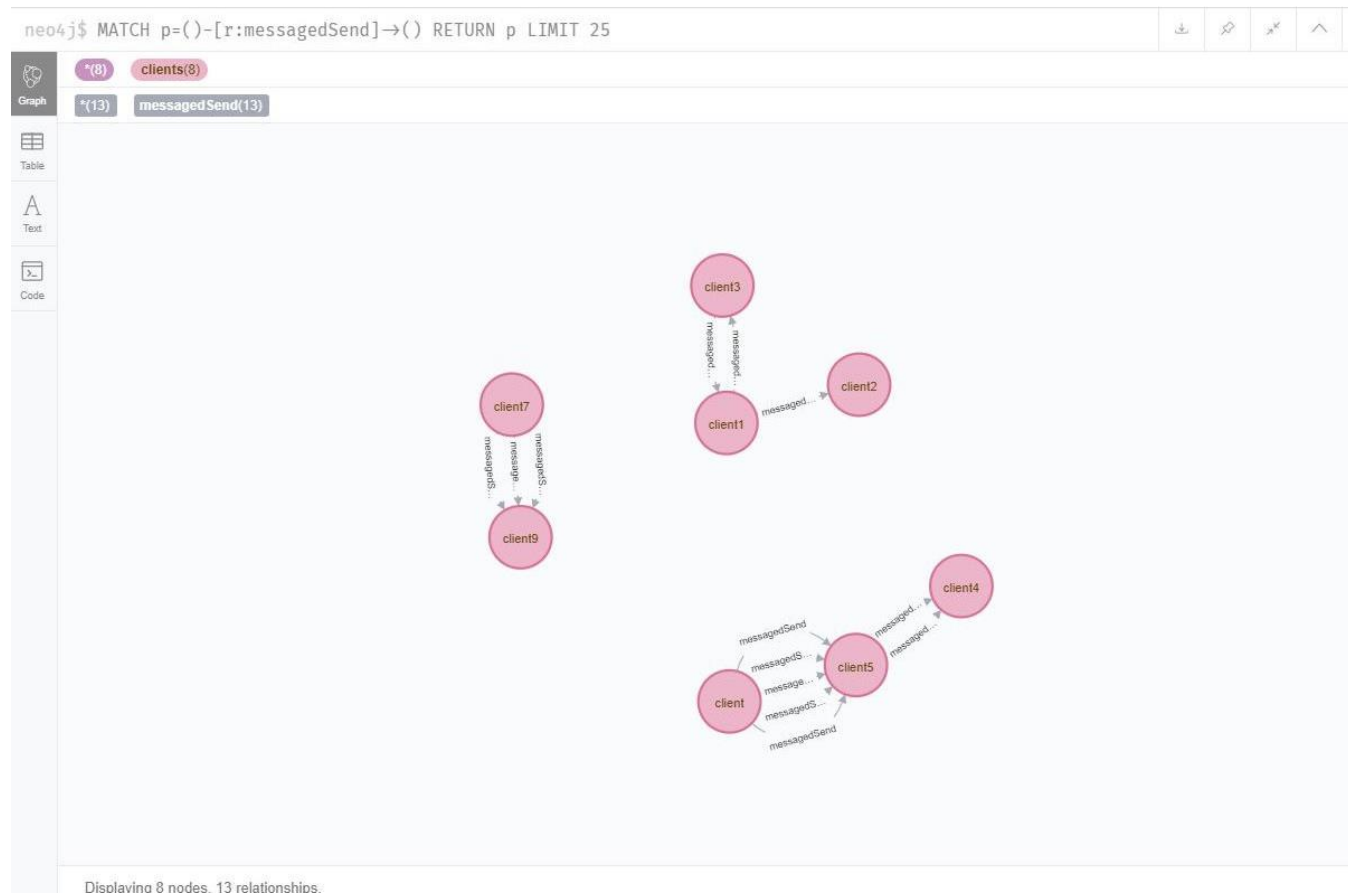
```

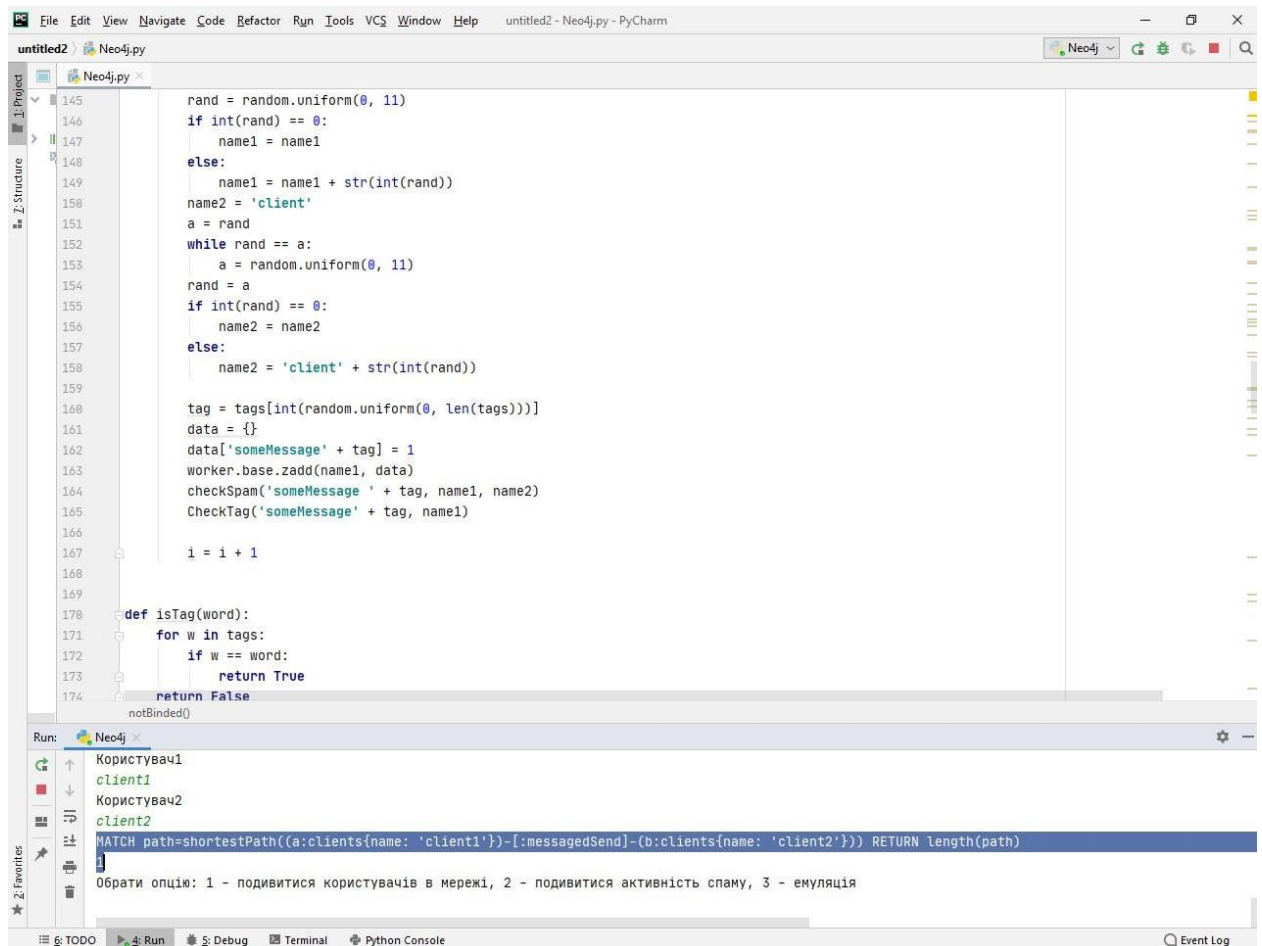
Event Log

UTF-8 1663 chars, 37 line breaks 3493:64 CRLF UTF-8 4 spaces Python 3.8

6.3. Задано два користувача. Знайти на графі найкоротший шлях між ними через відправлені або отримані повідомлення. Користувачі - client1, client2

MATCH path=shortestPath((a:clients{name: 'client1'})-[:messedSend]-  
(b:clients{name: 'client2'})) RETURN length(path)





6.4. Знайти авторів повідомлень, які пов'язані між собою лише повідомленнями, позначеними як "спам". MATCH (a:clients)-[r:messagingSend]->(b:clients) where not exists{(a)-[:messagingSend]{spam:"false"}]->(b)} RETURN a, b



neo4j\$ MATCH (a:clients)-[r:messagingSend]-(b:clients) where not exists{(a)-[:messagingSend{spam: "false"}]} ...

| a                           | b                            |
|-----------------------------|------------------------------|
| {<br>"name": "client6"<br>} | {<br>"name": "client7"<br>}  |
| {<br>"name": "client7"<br>} | {<br>"name": "client8"<br>}  |
| {<br>"name": "client7"<br>} | {<br>"name": "client1"<br>}  |
| {<br>"name": "client3"<br>} | {<br>"name": "client10"<br>} |
| {<br>"name": "client3"<br>} | {<br>"name": "client1"<br>}  |

Started streaming 13 records after 17 ms and completed after 18 ms.

untitled2 - Neo4j.py - PyCharm

untitled2 - Neo4j.py

```

else > if matchHead(name) > while True
MATCH (a:clients)-[r:messagingSend]-(b:clients) where not exists{(a)-[:messagingSend{spam: "false"}]} RETURN a, b LIMIT 25
<Node id=608 labels=('clients') properties={'name': 'client6'}>
<Node id=609 labels=('clients') properties={'name': 'client7'}>
<Node id=609 labels=('clients') properties={'name': 'client7'}>
<Node id=611 labels=('clients') properties={'name': 'client8'}>
<Node id=609 labels=('clients') properties={'name': 'client7'}>
<Node id=618 labels=('clients') properties={'name': 'client1'}>
<Node id=614 labels=('clients') properties={'name': 'client3'}>
<Node id=615 labels=('clients') properties={'name': 'client10'}>
<Node id=614 labels=('clients') properties={'name': 'client3'}>
<Node id=618 labels=('clients') properties={'name': 'client1'}>
<Node id=615 labels=('clients') properties={'name': 'client10'}>
<Node id=614 labels=('clients') properties={'name': 'client3'}>
<Node id=616 labels=('clients') properties={'name': 'client9'}>
<Node id=618 labels=('clients') properties={'name': 'client1'}>
<Node id=616 labels=('clients') properties={'name': 'client9'}>
<Node id=616 labels=('clients') properties={'name': 'client9'}>
<Node id=616 labels=('clients') properties={'name': 'client9'}>
<Node id=614 labels=('clients') properties={'name': 'client3'}>
<Node id=616 labels=('clients') properties={'name': 'client9'}>
<Node id=614 labels=('clients') properties={'name': 'client3'}>
<Node id=618 labels=('clients') properties={'name': 'client1'}>
<Node id=607 labels=('clients') properties={'name': 'client4'}>
<Node id=618 labels=('clients') properties={'name': 'client1'}>
<Node id=620 labels=('clients') properties={'name': 'client'}>
<Node id=619 labels=('clients') properties={'name': 'client2'}>

```

Event Log

UTF-8 1739 chars, 37 line breaks 46:04 CRLF UTF-8 4 spaces Python 3.8

6.5. Задано список тегів (tags). Знайти всіх користувачів, що відправили або отримали повідомлення з набором тегів tags, але ці користувачі не пов'язані

між собою. tags = ["action", "institute"] match (n: clients), (b: clients) where exists{(n)-[:tagged]->(m: tags {name: "action"}), (b)-[:tagged]->(m)} and exists{(n)-[:tagged]->(m: tags {name: "institute"}), (b)-[:tagged]->(m)} and not exists {(n)-[:messedSend]-(b)} return (n), (b)

