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 СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Получение данных из ВКонтакте для загрузки в графовую базу данных

Студент ИУ5-34М _____ **Ф.А. Аникин**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ **Ю.Е. Гапанюк**
(Подпись, дата) (И.О.Фамилия)

2023 г.

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)

УТВЕРЖДАЮ
Заведующий кафедрой ___ИУ5____
(Индекс)
_____ ___В.И. Терехов__
(И.О.Фамилия)
«_04_» ____сентября_____ 2023 г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме ___Получение данных из ВКонтакте для загрузки в графовую базу данных_____
_____

Студент группы _ИУ5-34М__

_____Аникин Филипп Автандилович_____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)
_____ИССЛЕДОВАТЕЛЬСКАЯ_____
_

Источник тематики (кафедра, предприятие, НИР) _____КАФЕДРА_____

График выполнения НИР: 25% к ____ нед., 50% к ____ нед., 75% к ___ нед., 100% к ____ нед.

Техническое задание _Провести исследование работы графовой базы данных NEO4J на___ данных, полученных из социальной сети «ВКонтакте»_____
_____
_____
_____
_____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на _____ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
_____
_____
_____
_____

Дата выдачи задания «_04_» ____сентября_____ 2023 г.

Руководитель НИР _____ _____Ю.Е. Гапанюк___
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____Ф.А. Аникин_____
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Содержание

Задание

Провести исследование работы графовой базы данных NEO4J [1] на данных, полученных из социальной сети «ВКонтакте» [2] с использованием VKAPI [3].

Алгоритм

В качестве средства разработки был выбран Python [4] версии 3.10 как один из наиболее популярных и простых языков программирования. Благодаря открытой системе дополнений язык позволяет расширять функционал и подключать библиотеки и модули. Так, например, мы используем модуль «neo4j», с помощью которого можно осуществлять работу с одноимённой БД непосредственно из кода программы, и библиотеку «requests», позволяющую формировать запросы к серверам и получать от них ответы в формате JSON.
Рассмотрим алгоритм работы реализованной программы:

1. Импорт библиотек и модулей;
2. Ввод переменных параметров запроса на получение списка подписчиков;
3. Запрос на получение списка подписчиков, из которого получаем общее их количество;
4. Создание файлов .csv в режиме перезаписи;
5. Цикл на итерационное получение списка подписчиков по 1000 штук;
6. Формирование строки-списка подписчиков s, которая как параметр далее передаётся в запрос на получение данных о подписчиках;
7. Обработка JSON-ответа от сервера и запись информации о каждом подписчике в файл names.csv;
8. Получение списка друзей подписчика и запись их ID в файл friends.csv;
9. Создание подключения к базе данных;
10. Чтение данных из созданных файлов и загрузка их в Neo4j.

VKAPI

API ВКонтакте — это интерфейс, который позволяет получать информацию из базы данных vk.com с помощью http-запросов к специальному серверу. Не нужно знать в подробностях, как устроена база, из каких таблиц и полей каких типов она состоит — достаточно того, что API-запрос об этом «знает». Синтаксис запросов и тип возвращаемых ими данных строго определены на стороне самого сервиса.

При помощи этого инструмента можно получать информацию с сервера посредством запросов, ответом на который будет JSON файл, содержащий в себе всю запрошенную информацию. JSON — это формат записи данных в виде пар «имя свойства»: «значение». С помощью ключевых слов в схеме создаются правила валидации структуры объекта и типов его полей.

В нашей задаче использовались три API запроса.

https://api.vk.com/method/groups.getMembers — запрос на получение всех участников выбранного сообщества.

Пример запроса:

https://api.vk.com/method/groups.getMembers?group_id=bmstu1830&count=5&access_token=**********v=5.131 , где параметры group_id – ID выбранного сообщества, count – число пользователей которых вернет запрос, access_token и v – обязательные параметры для всех запросов, обозначающие ID пользователя, который инициализирует запрос, и v – версия API, поскольку он постоянно улучшается.

Результат выполнения:

```
{
   "response(начало ответа)": {
      "count(число пользователей в сообществе)": 59205,
      "items(начало списка пользователей": [
         1566(ID пользователя в соц. сети VK),
         2857,
         3985,
         6892,
         7391
```

```
        ],
        "next_from": "PUkGEFNHDhJkUVwL"
    }
}
```


https://api.vk.com/method/users.get — запрос на получение информации о пользователях.

Пример запроса:

https://api.vk.com/method/users.get?user_ids=299030469%2C+252247047&access_token==**********&v=5.131, где user_ids – перечень ID пользователей, для которых будет выполнен запрос,

Результат выполнения:


```
{
    "response": [
        {
            "id": 299030469(ID пользователя),
            "first_name(Имя пользователя)": "Филипп",
            "last_name(Фамилия пользователя)": "Аникин",
            "can_access_closed(Флаг доступности)": true,
            "is_closed(Флаг приватности)": false
        },
        {
            "id": 252247047,
            "first_name": "Артём",
            "last_name": "Якубов",
            "can_access_closed": true,
            "is_closed": false
        }
    ]
}
```


https://api.vk.com/method/friends.get — запрос на получение списка друзей выбранного пользователя.

Пример запроса:

https://api.vk.com/method/friends.get?user_id=299030469&count=5&access_token==**********&v=5.131, где user_id – ID пользователя, у которого будет получен список друзей и count – число друзей, которое вернет запрос.

Результат выполнения:


```
{
    "response": {
        "count(Общее число друзей)": 79,
        "items": [
            4372050(ID друга),
            6374751,
            8389264,
            11164300,
```

```
            14596010
          ]
        }
      }
```

Программная реализация алгоритма

Импорт библиотек и модулей:

```
import requests
import time
import csv
```

Введём переменные параметров запроса на получение списка подписчиков:

```
token = '59e6c1a359e6...'
params = {
  'access_token': token,
  'v': 5.131,
  'group_id': 'bmstu1830',
  'offset': 0
}
```

Запрос на получение списка подписчиков, из которого получаем общее их количество:

```
    get_user_ids = requests.get('https://api.vk.com/method/groups.getMembers',
params=params).json()
    counter = get_user_ids['response']
    people_counter = 1
    all_members = counter['count']
```

Создание файлов .csv в режиме перезаписи:

```
    text_file = open('names.csv', 'w')
    text_file.close()
    text_file = open('friends.csv', 'w')
    text_file.close()
```

Цикл на итерационное получение списка подписчиков по 1000 штук:

```
    while all_members > 0:
        time.sleep(1)
        get_user_ids =
requests.get('https://api.vk.com/method/groups.getMembers',
params=params).json()
```

Формирование строки-списка подписчиков s, которая как параметр далее передаётся в запрос на получение данных о подписчиках:

```
        m = get_user_ids['response']['items']
        s = ''
        for i in m:
            s += str(i)
            if i != m[-1]:
                s += ','
        get_user_info = requests.get('https://api.vk.com/method/users.get',
                                     params={'access_token': token, 'v': 5.131,
                                             'user_ids': s,
```

```
                                                'fields':
'first_name,last_name,country, sex, bdate, city',
                                                'lang': 'ru', }).json()
```

Обработка JSON-ответа от сервера и запись информации о каждом подписчике в файл names.csv:

```
        for i in get_user_info['response']:
            if 'country' in i:
                country = i['country']['title']
            else:
                country = 'None'
            if 'city' in i:
                city = i['city']['title']
            else:
                city = 'None'
            with open('names.csv', 'a', encoding='utf-8', newline='') as
text_file:
                writer = csv.writer(text_file)
                writer.writerow(
                    [people_counter, i['id'], i['first_name'], i['last_name'],
country, city,
                     i['sex'], i.get('bdate', 'None')])
```

Получение списка друзей подписчика и запись их ID в файл friends.csv:

```
            get_user_friends =
requests.get('https://api.vk.com/method/friends.get',
                                             params={'access_token': token, 'v':
5.131,
                                                     'user_id': i['id']}).json()
            if 'response' in get_user_friends and
get_user_friends['response']['count'] > 0:
                friends = get_user_friends['response']['items']
            else:
                friends = '[]'
            with open('friends.csv', 'a', encoding='utf-8', newline='') as
text_file:
                writer = csv.writer(text_file)
                writer.writerow([i['id'], friends])
            people_counter += 1
            all_members -= 1
        params['offset'] += len(get_user_ids['response']['items'])
```

Второй скрипт – *vk_upload.py*. Он считывает данные из созданных в первом скрипте файлов и загружает их в Neo4j с помощью python библиотеки neo4j [5].

Импорт библиотек и модулей:

```
from neo4j import GraphDatabase
import csv
```

Глобальные переменные для подключения к базе данных:

```
URI = "bolt://localhost:7687"
AUTH = ("neo4j", "admin")
```

Функция для загрузки данных из файла в базу данных:

```
def upload_nodes(session):
    with open('names.csv', 'r', newline='') as csvfile:
        spamreader = csv.reader(csvfile, delimiter=',')
        for row in spamreader:
```

```
            session.run(
                "create (a:Person
{id:$id,name:$name,surname:$surname,country:$country,city:$city,sex:$sex,bdate:
$bdate})",
                id=row[1], name=row[2], surname=row[3], country=row[4],
city=row[5], sex=row[6], bdate=row[7])
```

Функция для добавления связей между узлами в БД, если подписчики являются друзьями:

```
def upload_friends(session):
    with open('friends.csv', 'r', newline='') as csvfile:
        spamreader = csv.reader(csvfile, delimiter=',')
        for row in spamreader:
            friends = row[1]
            friends = friends[1:-1]
            friend_list = list(friends.split(', '))
            for i in friend_list:
                if i:
                    result = session.run("match (a:Person {id:$id}) with
count(a) > 0 as exists "
                                         "return exists", id=i)
                    if result.single()[0]:
                        session.run("match (a {id:$id_a}), (b {id:$id_b})
create (a)-[:friends_with]->(b) ",
                                    id_a=row[0], id_b=i)
```

Создание переменной подключения, которая как параметр передаётся в вышеописанные функции:

```
session = GraphDatabase.driver(URI, auth=AUTH).session()
```

Часть кода отвечающая за загрузку в базу данных.

Весь код можно логически поделить на две части. Первая часть отвечает за обработку пользователей, хранящихся в полученном в ходе работы прошлых скриптов файле:

```
csv.field_size_limit(500000)
URI = "bolt://localhost:7687"
AUTH = ("neo4j", "admin")
session = GraphDatabase.driver(URI, auth=AUTH).session()

with open('names.csv', 'r', newline='') as csvfile:
    spamreader = csv.reader(csvfile, delimiter=',')
    for row in spamreader:
        log = open('log.txt', 'w')
        log.write(row[0])
        id = str(row[1])
        first_name = str(row[2])
        last_name = str(row[3])
        country = str(row[4])
        city = str(row[5])
        sex = str(row[6])
        b_date = str(row[7])
        friends = eval(row[8])

        result = session.run('match (n:FRIEND {id:$id}) set n:SUBSCRIBER
return count(n)>0', id=id)
        if not result.single()[0]:
            session.run(
                'create (n:SUBSCRIBER '
                '{id:$id,  first_name:$first_name,  last_name:$last_name,
country:$country, city:$city, '
                'sex:$sex,        b_date:$b_date})        ',        id=id,
first_name=first_name, last_name=last_name, country=country,
```

```
                    city=city, sex=sex, b_date=b_date)
```

Вторая часть ответственна за создание узлов друзей и связей с узлами пользователей, если данный друг связан с пользователем:

```
            for i in friends:
                id_f = str(i[0])
                first_name = str(i[1])
                last_name = str(i[2])
                country = str(i[3])
                city = str(i[4])
                sex = str(i[5])
                b_date = str(i[6])

                result = session.run('match (n:SUBSCRIBER {id:$id}) set
n:FRIEND return count(n)>0', id=id_f)
                if not result.single()[0]:
                    result_1 = session.run('match (n:FRIEND {id:$id}) return
count(n)>0', id=id_f)
                    if not result_1.single()[0]:
                        session.run(
                            'create (n:FRIEND '
                            '{id:$id, first_name:$first_name,
last_name:$last_name, country:$country, city:$city, '
                            'sex:$sex, b_date:$b_date}) ', id=id_f,
first_name=first_name, last_name=last_name,
                            country=country,
                            city=city, sex=sex, b_date=b_date)
                session.run('match (n:SUBSCRIBER {id:$id}), (m:FRIEND
{id:$id_f}) merge (n)-[:FRIENDS_WITH]->(m)', id=id,
                            id_f=id_f)

                session.run('match (n {id:$id}), (m {id:$id_f}) merge (n)-
[:FRIENDS_WITH]->(m)', id=id, id_f=id_f)
```

Структура данных

Использование API позволяет получить данные, которые потом записываются в файл формата .csv, где хранятся данные о пользователях и его друзьях. Каждая запись представляет собой линейное описание пользователя в формате: номер строки в файле, ID в VK, имя, фамилия, страна (если она указана пользователем), город (если он указан пользователем), пол (1 – девушка, 2 – мужчина), дата рождения (если она указана пользователем), и такое же описание каждого из его друзей.

К примеру, вот как выглядит описание случайного пользователя:

1,1566,Оля,Проневич,Россия,Москва,1,26.3.1989,"[[13121, 'Татьяна', 'Келлер', 'Россия', 'Москва', 1, '30.7'], [13735, 'Анна', 'Чиркова', 'Россия', 'Санкт-Петербург', 1, '26.3.1989'], [13749, 'Александр', 'Левантовский', 'Россия', 'Санкт-

Петербург', 2, '1.3.1986' … , [36102, 'Марина', 'Сафонова', 'Россия', 'Москва', 1, '6.8']"

После формирования файла, хранящего в себе все данные о пользователях, полученные данные загружаются в графовую базу данных Neo4j. Узлы в качестве атрибутов хранят в себе данные из строк файла, кроме списка друзей. Связи между пользователями реализованы как связи между узлами. Пользователям из файла присваивается метка SUBSCRIBER, а их друзьям присваивается метка FRIEND. В случае если пользователь-подписчик является другом другого пользователя-подписчика, он будет обладать двумя метками сразу.

Пример содержимого узла и его связей представлен на рисунках 1-3.

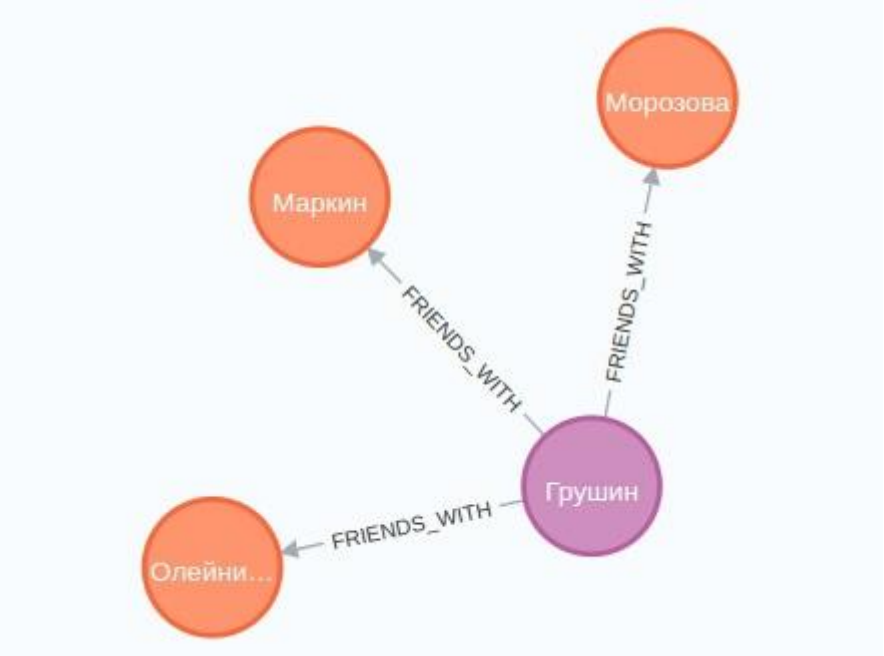

Рисунок 1. Связи между узлами

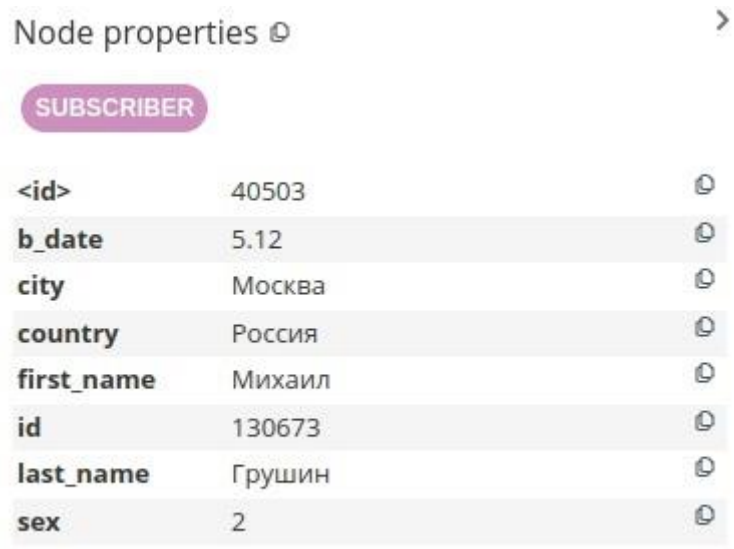

Рисунок 2. Атрибуты узла

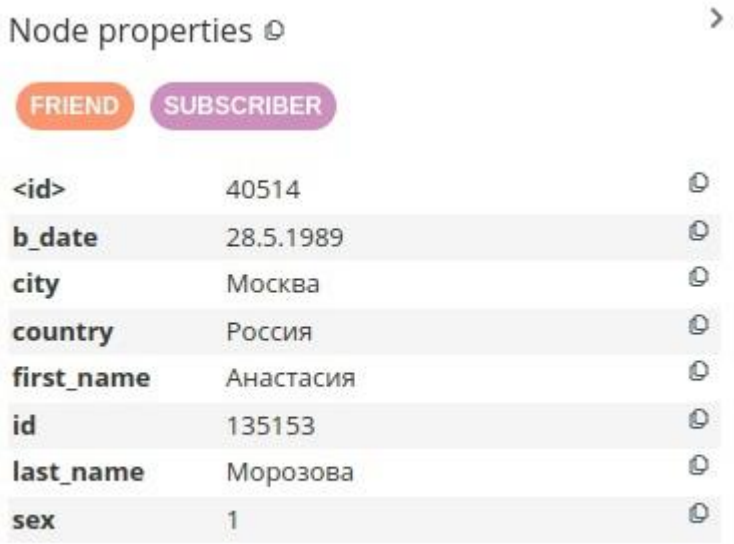

Рисунок 3. Атрибуты узла

Статистика

По полученным данным можно делать статистические замеры. К примеру, диаграмма ниже (рисунок 4) показывает принадлежность пользователей к их возрастной категории.

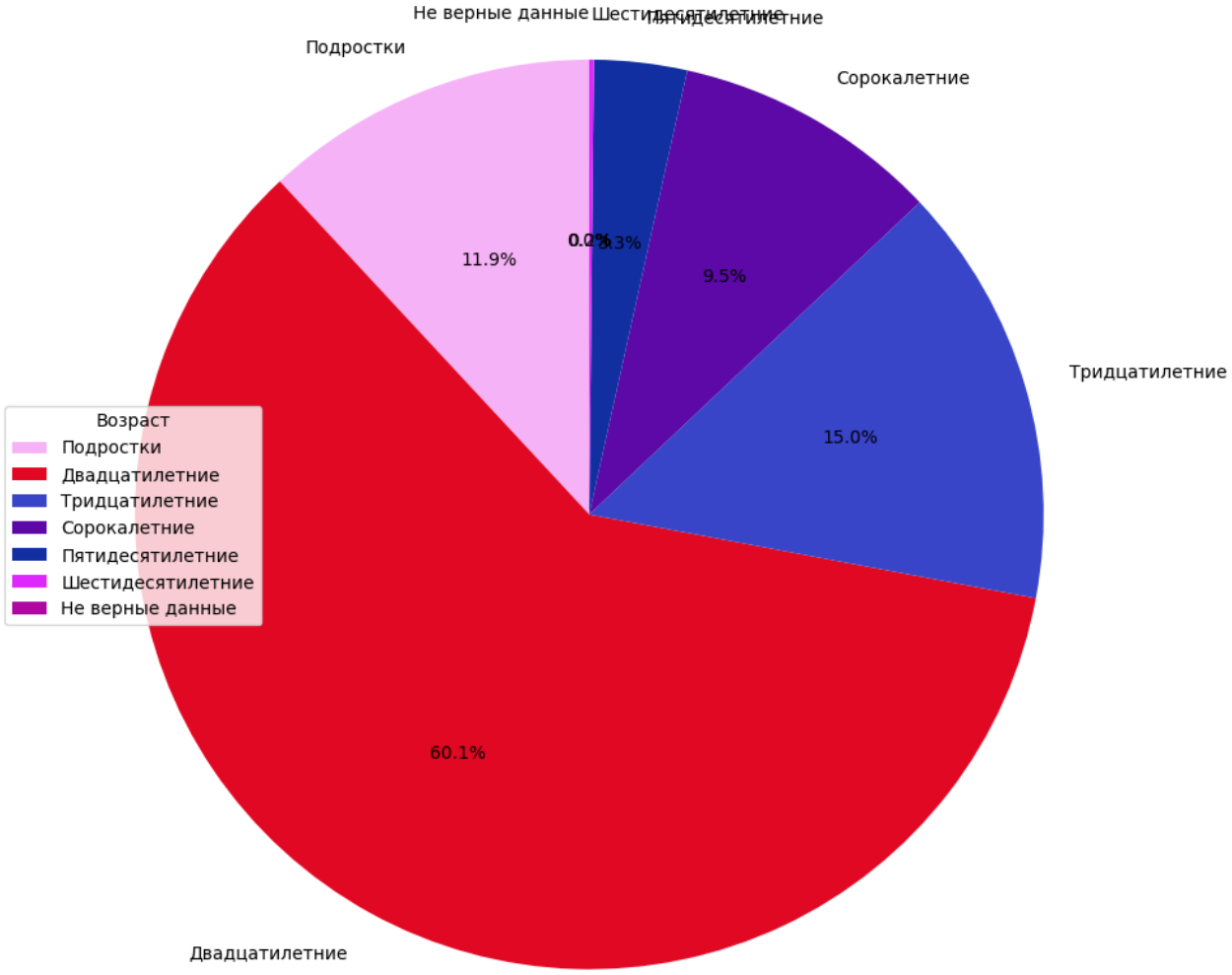

Рисунок 4. Принадлежность пользователей к возрастным группам

Подобно возрасту, можно сделать замер городов пользователей. Диаграмма ниже (рисунок 5) демонстрирует, что подавляющее большинство пользователей из города Москва.

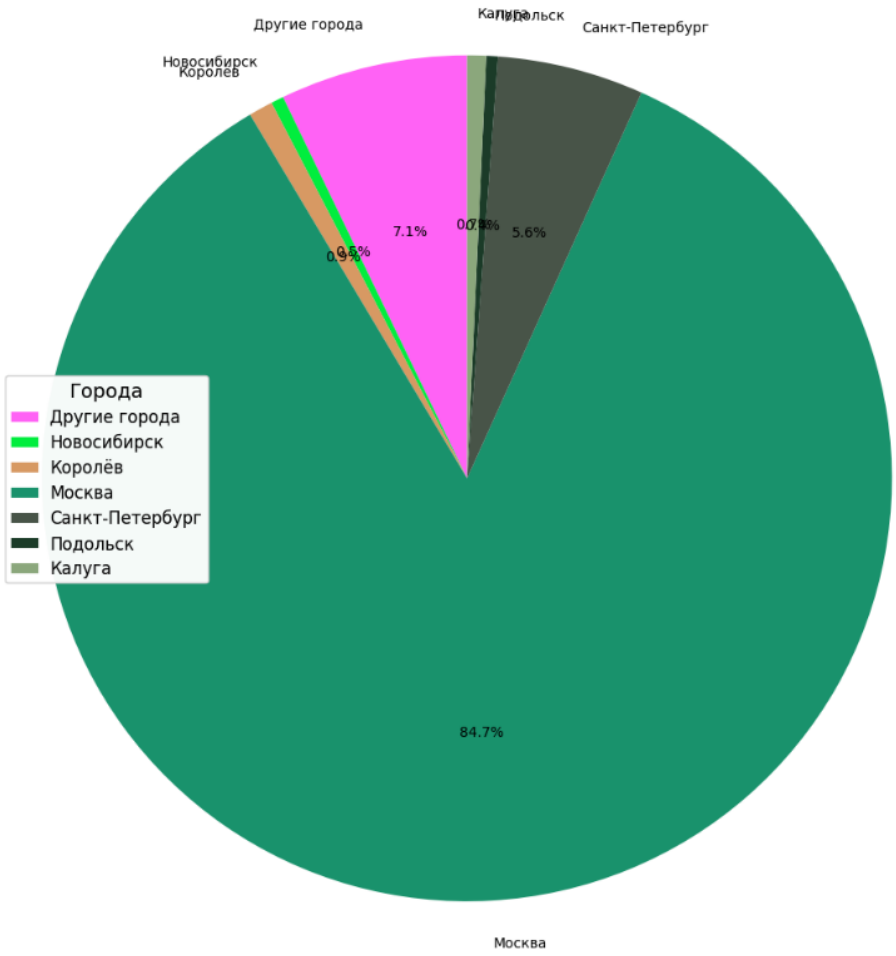

Рисунок 5. Диаграмма распределения городов

По имеющимся данным можно также выявить распределение возраста и количества друзей, для демонстрации заинтересованности пользователей в новых знакомствах. На диаграмме ниже (рисунок 6) каждая точка — это пользователь, а цвета указывают на пол человека, синий – мужчина, красный – женщина.

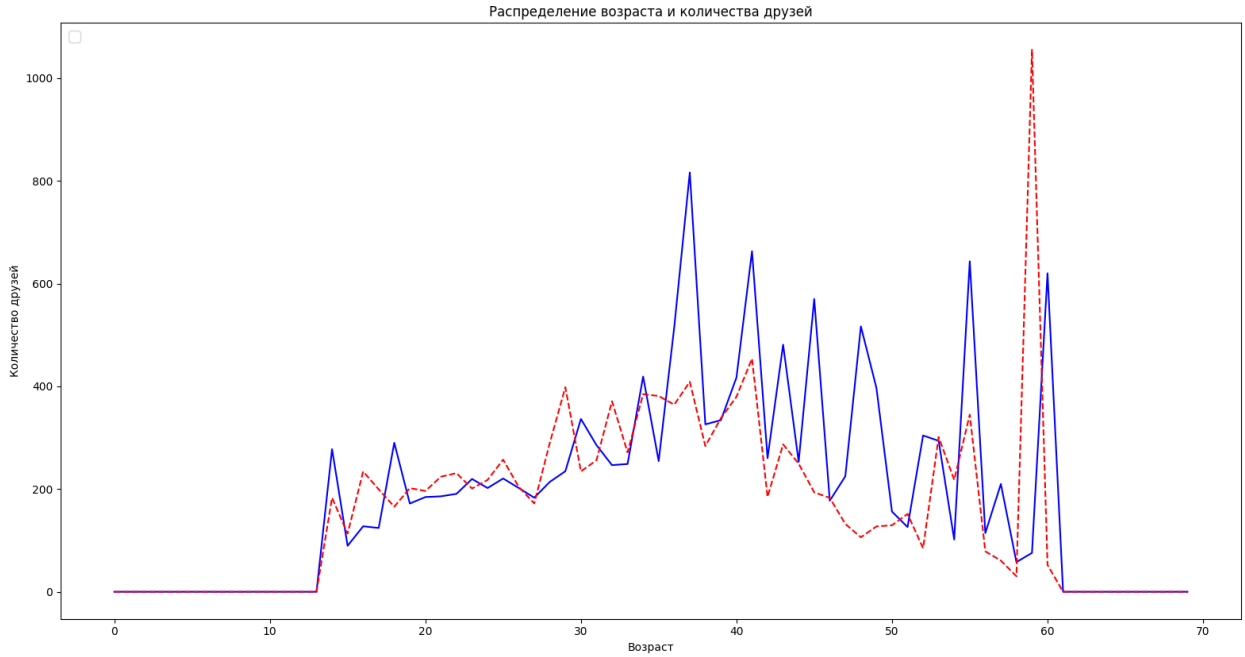

Рисунок 6. Распределение количества друзей по возрасту

Пример полученной базы данных

Для просмотра данных используем встроенный инструмент управления базой данных «Neo4j», позволяющий отображать результаты запросов в виде графов.

Пример выполнения скрипта для поиска цельной цепочки знакомств у конкретного человека (рисунок 7):

```
def people_cluster(tx):
    buffer = []
    id = '186482'
    target = [id]
    result = tx.run("MATCH (n)-[]->(m) where n.id=$id RETURN m.id", id=id)
    for record in result:
        buffer.append(record[0])
    while len(buffer) != 0:
        for i in buffer:
            result = tx.run("MATCH (n)-[]->(m) where n.id=$id RETURN m.id", id=i)
            for record in result:
                if (record[0] not in target) and (record[0] not in buffer):
                    buffer.append(record[0])
            target.append(i)
            buffer.remove(i)
    print(target)
```

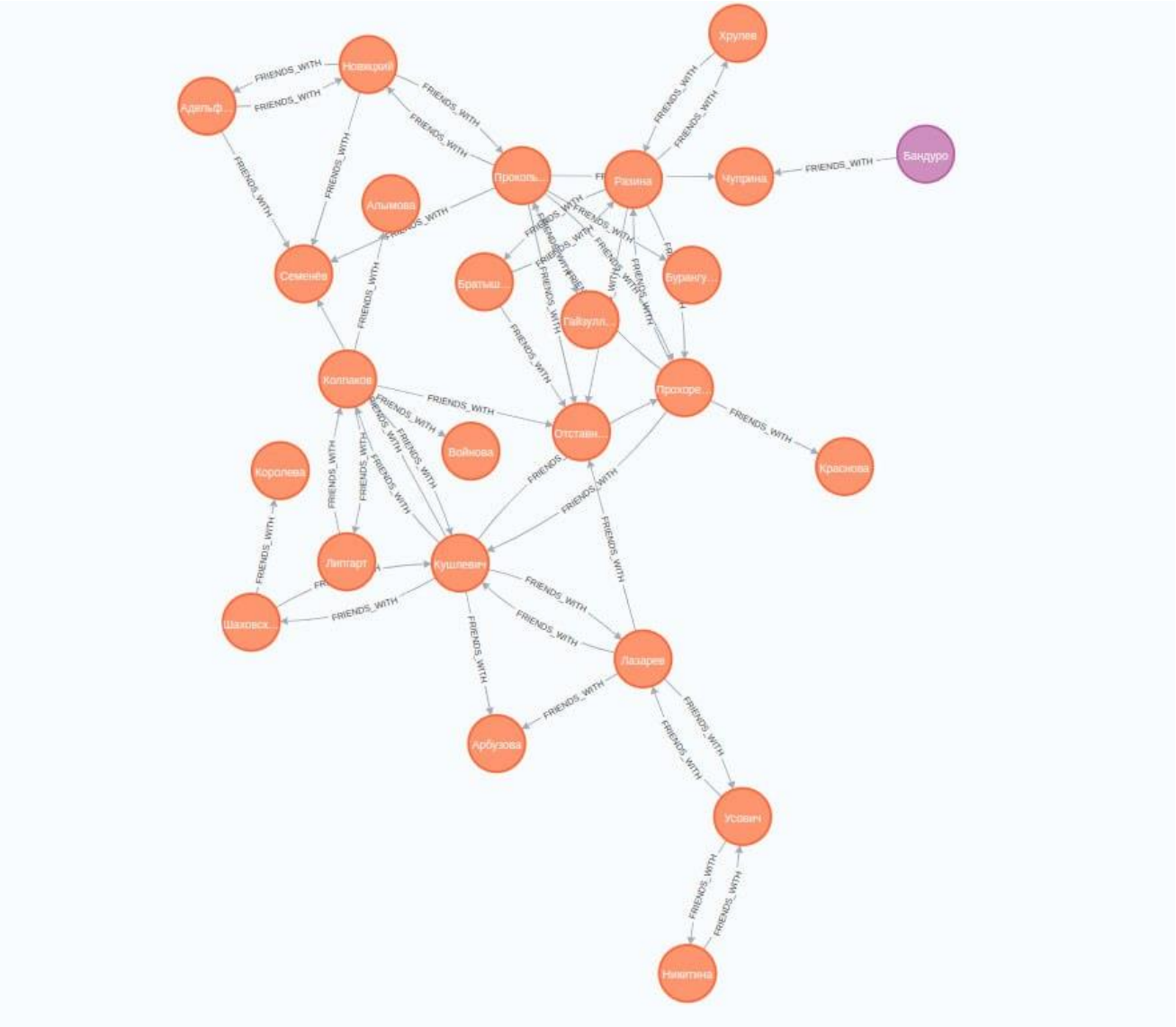

Рисунок 7. Цельная цепочка знакомств

На основе полученных диаграммы и данных можно производить анализ связей у конкретного человека. К примеру, на диаграмме ниже (рисунок 8) продемонстрировано распределение возраста у цельной цепочки пользователей.

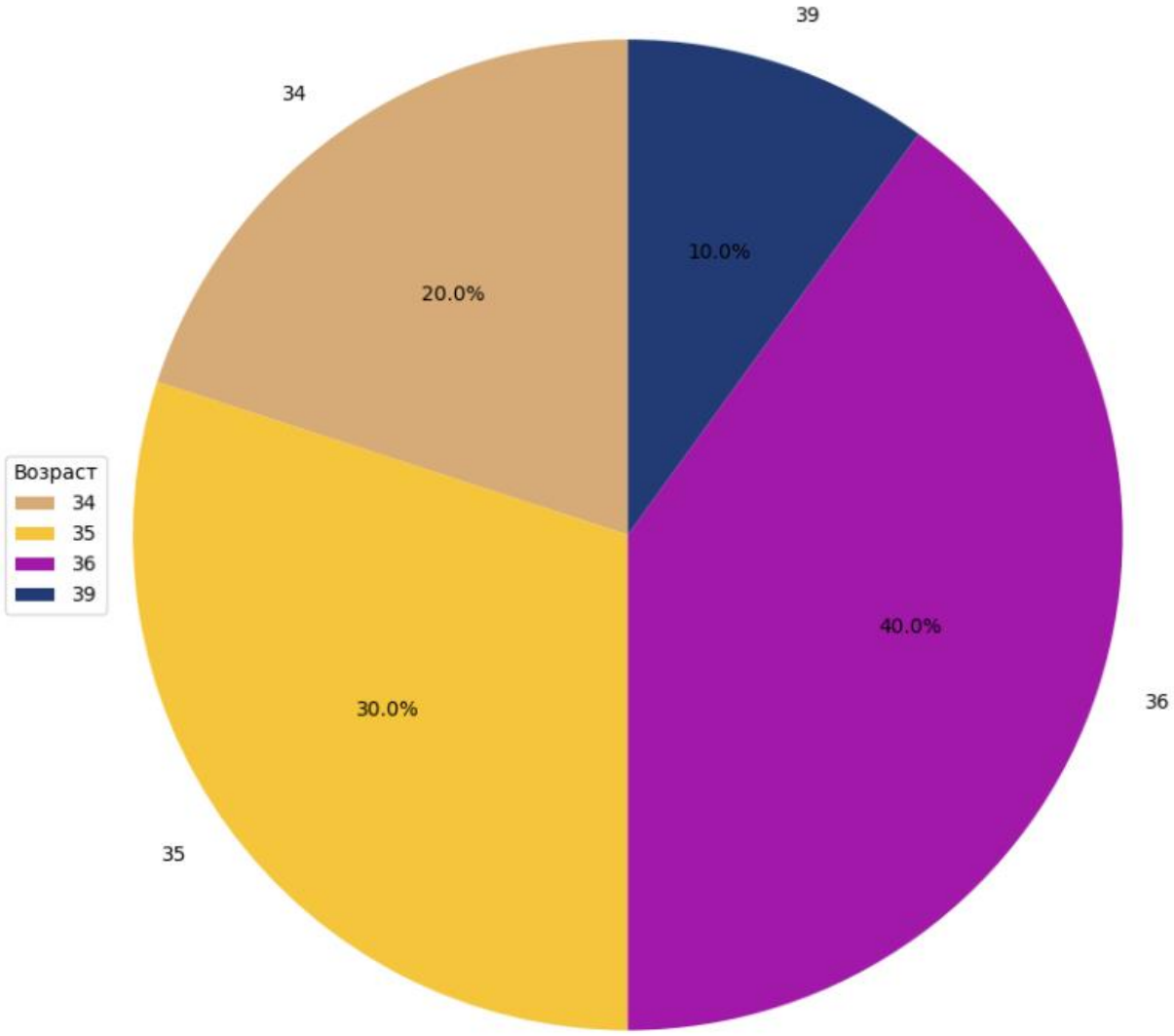

Рисунок 8. Распределение возраста у цельной цепочки пользователей

Заключение

В результате выполнения научно-исследовательской работы было проведено исследование работы графовой БД Neo4j на примере данных из «ВКонтакте». Была произведена выгрузка данных из ВК посредством VKAPI в файл формата .csv, с последующей загрузкой в Neo4j. На основе полученной БД можно производить анализ выбранного сообщества и внутренних групп людей.

Список литературы

1. Официальный сайт Neo4j: [Электронный ресурс]. URL: https://neo4j.com/ (Дата обращения: 28.03.2023).
2. Социальная сеть «ВКонтакте»: [Электронный ресурс]. URL: https://vk.com/ (Дата обращения: 22.03.2023).
3. VK API: [Электронный ресурс]. URL: https://api.vk.com/ (Дата обращения: 22.03.2023).
4. Официальный сайт Python: [Электронный ресурс]. URL: https://www.python.org/ (Дата обращения: 22.03.2023).
5. Документация по модулю «neo4j»: [Электронный ресурс]. URL: https://neo4j.com/developer/python/ (Дата обращения: 30.03.2023).

ПРИЛОЖЕНИЕ А

vk_parsing.py

```
import requests
import time
import csv

token = '************************************'

params = {
   'access_token': token,
   'v': 5.131,
   'group_id': 'bmstu1830',
   'offset': 0
}
get_user_ids = requests.get('https://api.vk.com/method/groups.getMembers',
params=params).json()
counter = get_user_ids['response']

people_counter = 1

all_members = counter['count']
text_file = open('names.csv', 'w')
text_file.close()


def main():
   current_friends = []
   token = '************************************'
   params = {
      'access_token': token,
      'v': 5.131
   }
   get_user_ids = requests.get('https://api.vk.com/method/groups.getMembers',
params=params).json()
   people_counter = 1
   all_members = counter['count']
   text_file = open('names.csv', 'w')
   text_file.close()

   while all_members > 0 and params['offset'] < 7074000:
      time.sleep(1)
      get_user_ids = requests.get('https://api.vk.com/method/groups.getMembers',
params=params).json()
      print(get_user_ids)
      print(params['offset'])
      m = get_user_ids['response']['items']
      s = ''
      for i in m:
```

```
    s += str(i)
    if i != m[-1]:
        s += ','

get_user_info = requests.get('https://api.vk.com/method/users.get',
                             params={'access_token': token, 'v': 5.131,
                                     'user_ids': s,
                                     'fields': 'first_name,last_name,country, sex, bdate, city',
                                     'lang': 'ru', }).json()

for i in get_user_info['response']:
    if 'country' in i:
        country = i['country']['title']
    else:
        country = 'None'
    if 'city' in i:
        city = i['city']['title']
    else:
        city = 'None'

    get_user_friends = requests.get('https://api.vk.com/method/friends.get',
                                    params={'access_token': token, 'v': 5.131,
                                            'user_id': i['id'],
                                            'fields': 'first_name,last_name,country, sex, bdate, city',
                                            'lang': 'ru'}).json()

    if 'response' in get_user_friends and get_user_friends['response']['count'] > 0:
        friends = get_user_friends['response']['items']
        for n in friends:
            if 'country' in n:
                country_f = n['country']['title']
            else:
                country_f = 'None'
            if 'city' in n:
                city_f = n['city']['title']
            else:
                city_f = 'None'
            current_friends.append(list([n['id'], n['first_name'], n['last_name'], country_f, city_f,
                                         n['sex'], n.get('bdate', 'None')]))
    else:
        friends = '[]'

    with open('names.csv', 'a', encoding='utf-8', newline='') as text_file:
        writer = csv.writer(text_file)
        writer.writerow(
            [people_counter, i['id'], i['first_name'], i['last_name'], country, city,
             i['sex'], i.get('bdate', 'None'), list(current_friends)])

    current_friends = []
    people_counter += 1
    all_members -= 1
```

```
        params['offset'] += len(get_user_ids['response']['items'])


main()
```

neo_upload.py

```
from neo4j import GraphDatabase
import csv

csv.field_size_limit(500000)
URI = "bolt://localhost:7687"
AUTH = ("neo4j", "admin")
session = GraphDatabase.driver(URI, auth=AUTH).session()

with open('names.csv', 'r', newline='') as csvfile:
    spamreader = csv.reader(csvfile, delimiter=',')
    for row in spamreader:
        log = open('log.txt', 'w')
        log.write(row[0])
        id = str(row[1])
        first_name = str(row[2])
        last_name = str(row[3])
        country = str(row[4])
        city = str(row[5])
        sex = str(row[6])
        b_date = str(row[7])
        friends = eval(row[8])

        result = session.run('match (n:FRIEND {id:$id}) set n:SUBSCRIBER return count(n)>0',
id=id)
        if not result.single()[0]:
            session.run(
                'create (n:SUBSCRIBER '
                '{id:$id, first_name:$first_name, last_name:$last_name, country:$country, city:$city, '
                'sex:$sex, b_date:$b_date}) ', id=id, first_name=first_name, last_name=last_name,
country=country,
                city=city, sex=sex, b_date=b_date)

        for i in friends:
            id_f = str(i[0])
            first_name = str(i[1])
            last_name = str(i[2])
            country = str(i[3])
            city = str(i[4])
            sex = str(i[5])
            b_date = str(i[6])

            result = session.run('match (n:SUBSCRIBER {id:$id}) set n:FRIEND return count(n)>0',
id=id_f)
            if not result.single()[0]:
                result_1 = session.run('match (n:FRIEND {id:$id}) return count(n)>0', id=id_f)
                if not result_1.single()[0]:
```

```
                session.run(
                    'create (n:FRIEND '
                    '{id:$id, first_name:$first_name, last_name:$last_name, country:$country,
city:$city, '
                    'sex:$sex, b_date:$b_date}) ', id=id_f, first_name=first_name,
last_name=last_name,
                    country=country,
                    city=city, sex=sex, b_date=b_date)
            session.run('match (n:SUBSCRIBER {id:$id}), (m:FRIEND {id:$id_f}) merge (n)-
[:FRIENDS_WITH]->(m)', id=id,
                    id_f=id_f)

            session.run('match (n {id:$id}), (m {id:$id_f}) merge (n)-[:FRIENDS_WITH]->(m)',
id=id, id_f=id_f)
log.close()
```

get_nodes.py

```
from neo4j import GraphDatabase
import csv

neo4j_uri = "bolt://localhost:7687"
neo4j_user = "neo4j"
neo4j_password = "admin"


def people_cluster(tx):
    buffer = []
    id = '186482'
    target = [id]
    result = tx.run("MATCH (n)-[]->(m) where n.id=$id RETURN m.id", id=id)
    for record in result:
        buffer.append(record[0])
    while len(buffer) != 0:
        for i in buffer:
            result = tx.run("MATCH (n)-[]->(m) where n.id=$id RETURN m.id", id=i)
            for record in result:
                if (record[0] not in target) and (record[0] not in buffer):
                    buffer.append(record[0])
            target.append(i)
            buffer.remove(i)
    print(target)


def create_file_cluster(tx):
    reader = open('buffer.txt', 'r', encoding='utf-8')
    result = tx.run(
        f"match (n:SUBSCRIBER) where n.id in {reader.__next__()} return n")
    counter = 0
    kek = open('friends_cluster.csv', 'w', encoding='utf-8', newline='')
    kek = open('friends_cluster.csv', 'a', encoding='utf-8', newline='')
    writer = csv.writer(kek)
```

```
    for record in result:
        friends = [[]]
        counter += 1
        node_n = record["n"]
        result_1 = tx.run(f'match (n)-[:FRIENDS_WITH]->(m) where n.id=$id return m',
id=str(node_n["id"]))
        for record_1 in result_1:
            friends_buffer = []
            values = record_1['m']
            friends_buffer.append(['---', values['id'], values['first_name'], values['last_name'],
values['country'],
                                   values['city'], values['sex'], values['b_date']])
            friends.append(friends_buffer[0])
        writer.writerow(friends)
        print(friends)


with GraphDatabase.driver(neo4j_uri, auth=(neo4j_user, neo4j_password)) as driver:
    with driver.session() as session:
        session.read_transaction(create_file_cluster)
```

ПРИЛОЖЕНИЕ Б

Результат выполнения запроса match (n:SUBSCRIBER) return n;

...

(:SUBSCRIBER:FRIEND {country: "Россия", city: "None", sex: "2", last_name: "Топаев", id: "63305341", first_name: "Даниил", b_date: "20.10.1997"})

(:SUBSCRIBER {country: "None", city: "None", sex: "2", last_name: "Козлов", id: "63324362", first_name: "Коля", b_date: "None"})

(:SUBSCRIBER {country: "Россия", city: "Брянск", sex: "2", last_name: "Хрычев", id: "63336177", first_name: "Илья", b_date: "9.8"})

(:SUBSCRIBER {country: "Россия", city: "None", sex: "2", last_name: "Лобзин", id: "63339269", first_name: "Илья", b_date: "7.9.2001"})

(:SUBSCRIBER {country: "Россия", city: "Москва", sex: "1", last_name: "Сивачева", id: "63347306", first_name: "Наталья", b_date: "18.12"})

(:SUBSCRIBER {country: "Россия", city: "Борисоглебск", sex: "1", last_name: "Иванцова", id: "63353357", first_name: "Наташа", b_date: "27.12.1902"})

(:SUBSCRIBER:FRIEND {country: "Россия", city: "Самара", sex: "1", last_name: "Андреева", id: "63361651", first_name: "Ольга", b_date: "9.12"})

(:SUBSCRIBER {country: "None", city: "Казань", sex: "1", last_name: "Климова", id: "63369573", first_name: "Юлия", b_date: "8.10.1986"})

(:SUBSCRIBER {country: "Гваделупа", city: "Pointe-à-Pitre", sex: "2", last_name: "Lighthorse", id: "63378633", first_name: "Elite", b_date: "29.4"})

(:SUBSCRIBER {country: "Россия", city: "Москва", sex: "1", last_name: "Сергеева", id: "63386677", first_name: "Дишулька", b_date: "None"})

(:SUBSCRIBER {country: "Россия", city: "Москва", sex: "2", last_name: "Поплавский", id: "63397022", first_name: "Артем", b_date: "None"})

(:SUBSCRIBER {country: "Россия", city: "Москва", sex: "1", last_name: "Орлова", id: "63432718", first_name: "Леся", b_date: "7.9.1999"})

(:SUBSCRIBER {country: "Россия", city: "Иваново", sex: "2", last_name: "Каньшин", id: "63448039", first_name: "Максим", b_date: "9.12.1992"})

(:SUBSCRIBER {country: "None", city: "Москва", sex: "2", last_name: "", id: "63453314", first_name: "DELETED", b_date: "18.4.1999"})

(:SUBSCRIBER {country: "Россия", city: "Пермь", sex: "1", last_name: "Новикова", id: "63453581", first_name: "Ирина", b_date: "18.10.1971"})

(:SUBSCRIBER {country: "Россия", city: "Копейск", sex: "2", last_name: "Ханов", id: "63480964", first_name: "Руслан", b_date: "10.11"})
(:SUBSCRIBER {country: "Россия", city: "Рязань", sex: "1", last_name: "Ларина", id: "63490529", first_name: "Елена", b_date: "17.11.1979"})
(:SUBSCRIBER {country: "Россия", city: "Королёв", sex: "2", last_name: "Волков", id: "63502374", first_name: "Андрей", b_date: "13.10"})
(:SUBSCRIBER:FRIEND {country: "Россия", city: "None", sex: "2", last_name: "Сулейманов", id: "63523295", first_name: "Артур", b_date: "None"})
(:SUBSCRIBER {country: "Россия", city: "None", sex: "2", last_name: "Мансуров", id: "63528714", first_name: "Максим", b_date: "4.2"})
(:SUBSCRIBER {country: "Россия", city: "Москва", sex: "1", last_name: "Цимбалова", id: "63601387", first_name: "Елена", b_date: "2.12"})
(:SUBSCRIBER {country: "Россия", city: "Ярославль", sex: "2", last_name: "Агежов", id: "63603171", first_name: "Андрей", b_date: "25.12"})
(:SUBSCRIBER {country: "Россия", city: "Омск", sex: "2", last_name: "Одесский", id: "63610710", first_name: "Вадим", b_date: "None"})
(:SUBSCRIBER {country: "Россия", city: "Москва", sex: "2", last_name: "Шишкин", id: "63634774", first_name: "Богдан", b_date: "16.1"})
(:SUBSCRIBER {country: "Россия", city: "Москва", sex: "1", last_name: "Proko", id: "63667768", first_name: "Elena", b_date: "None"})
…

Результат выполнения запроса match (n) return n;

…

(:FRIEND {country: "Беларусь", city: "Гомель", sex: "1", last_name: "Рогачева", id: "259489765", first_name: "Екатерина", b_date: "None"})
(:FRIEND {country: "None", city: "None", sex: "2", last_name: "Gontovoy", id: "268045741", first_name: "Max", b_date: "None"})
(:FRIEND {country: "Украина", city: "Киев", sex: "2", last_name: "Горбенко", id: "270964227", first_name: "Александр", b_date: "None"})
(:FRIEND {country: "Нидерланды", city: "Rotterdam", sex: "2", last_name: "Кравченко", id: "272174417", first_name: "Илья", b_date: "None"})
(:FRIEND {country: "Украина", city: "Павлоград", sex: "2", last_name: "Хазиев", id: "275939523", first_name: "Вадим", b_date: "None"})

(:FRIEND {country: "Россия", city: "Москва", sex: "1", last_name: "Шарапова", id: "296276875", first_name: "Ангелина", b_date: "3.11"})
(:FRIEND {country: "None", city: "None", sex: "1", last_name: "Зеленина", id: "305757444", first_name: "Даша", b_date: "21.6"})
(:FRIEND {country: "Россия", city: "Москва", sex: "1", last_name: "Красникова", id: "320765775", first_name: "Ксюша", b_date: "12.1"})
(:FRIEND {country: "Россия", city: "Москва", sex: "1", last_name: "Облака", id: "370852134", first_name: "Тайские", b_date: "None"})
(:FRIEND {country: "Россия", city: "Фрязино", sex: "1", last_name: "Кондратова", id: "449997325", first_name: "Маргарита", b_date: "None"})
(:FRIEND {country: "Россия", city: "Санкт-Петербург", sex: "1", last_name: "Игрушки", id: "712885354", first_name: "Вязаные", b_date: "20.1.1958"})
(:FRIEND {country: "None", city: "None", sex: "1", last_name: "Поляк", id: "748908070", first_name: "Карина", b_date: "3.9.2008"})
(:SUBSCRIBER {country: "Россия", city: "Архангельск", sex: "2", last_name: "Архипов", id: "4339775", first_name: "Евгений", b_date: "11.9.2003"})
(:FRIEND {country: "Россия", city: "Северодвинск", sex: "2", last_name: "Коршаков", id: "31787", first_name: "Сергей", b_date: "30.11"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "1", last_name: "Яремчук", id: "1540323", first_name: "Александра", b_date: "20.9.1988"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "2", last_name: "Гречищев", id: "2489458", first_name: "Михаил", b_date: "3.12"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "1", last_name: "Lagunova", id: "3195102", first_name: "Ekaterina", b_date: "25.11.1991"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "2", last_name: "Ким", id: "3456831", first_name: "Руслан", b_date: "14.1.1980"})
(:FRIEND {country: "Россия", city: "Ростов-на-Дону", sex: "1", last_name: "Игнатенко", id: "6495232", first_name: "Татьяна", b_date: "6.9"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "2", last_name: "Кротов", id: "18034471", first_name: "Сергей", b_date: "8.3.1986"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "2", last_name: "Сулоев", id: "36293910", first_name: "Егор", b_date: "None"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "1", last_name: "Дубровская", id: "38128975", first_name: "Ольга", b_date: "24.8.1977"})

(:FRIEND {country: "Россия", city: "Архангельск", sex: "2", last_name: "Корельский", id: "38809347", first_name: "Андрей", b_date: "None"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "1", last_name: "Братанова", id: "44594822", first_name: "Ольга", b_date: "26.2"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "2", last_name: "Окунев", id: "64462965", first_name: "Иван", b_date: "11.1.2003"})
(:FRIEND {country: "Россия", city: "Москва", sex: "2", last_name: "Демянчук", id: "65734221", first_name: "Степан", b_date: "10.3"})
(:FRIEND {country: "Россия", city: "Санкт-Петербург", sex: "1", last_name: "Слободина", id: "71273232", first_name: "Лиза", b_date: "10.1"})
(:FRIEND {country: "None", city: "None", sex: "2", last_name: "Самарин", id: "83582769", first_name: "Михаил", b_date: "None"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "2", last_name: "Туфанов", id: "83661609", first_name: "Вячеслав", b_date: "None"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "2", last_name: "Супрун", id: "88272090", first_name: "Сергей", b_date: "None"})
(:FRIEND {country: "Россия", city: "None", sex: "2", last_name: "Несветайло", id: "90540273", first_name: "Никита", b_date: "None"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "2", last_name: "Vladimirovich", id: "96161184", first_name: "Yury", b_date: "None"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "2", last_name: "Ширяев", id: "97268466", first_name: "Дима", b_date: "19.4.1999"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "1", last_name: "Корельская", id: "98398719", first_name: "Юлия", b_date: "10.6"})
(:FRIEND {country: "Россия", city: "None", sex: "2", last_name: "Великонивцев", id: "98401233", first_name: "Фёдор", b_date: "18.7"})
(:FRIEND {country: "Россия", city: "None", sex: "2", last_name: "Симутин", id: "100092568", first_name: "Роман", b_date: "23.11"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "2", last_name: "Вахрушев", id: "105503636", first_name: "Александр", b_date: "None"})
(:FRIEND {country: "Россия", city: "Санкт-Петербург", sex: "2", last_name: "Шебунин", id: "110862878", first_name: "Вадим", b_date: "7.5"})
(:FRIEND {country: "Россия", city: "Архангельск", sex: "2", last_name: "Бугрин", id: "112613077", first_name: "Илья", b_date: "10.6"})

(:FRIEND {country: "Россия", city: "Архангельск", sex: "1", last_name: "Тодрик", id: "113804247", first_name: "Анастасия", b_date: "None"})

(:FRIEND {country: "Россия", city: "Москва", sex: "1", last_name: "Гриневская", id: "114937867", first_name: "Дарья", b_date: "6.10"})

(:FRIEND {country: "Россия", city: "Архангельск", sex: "2", last_name: "Шарапов", id: "115136608", first_name: "Артем", b_date: "24.9"})

(:FRIEND {country: "Россия", city: "Архангельск", sex: "1", last_name: "Агеева", id: "117243877", first_name: "Алёна", b_date: "7.5"})

...

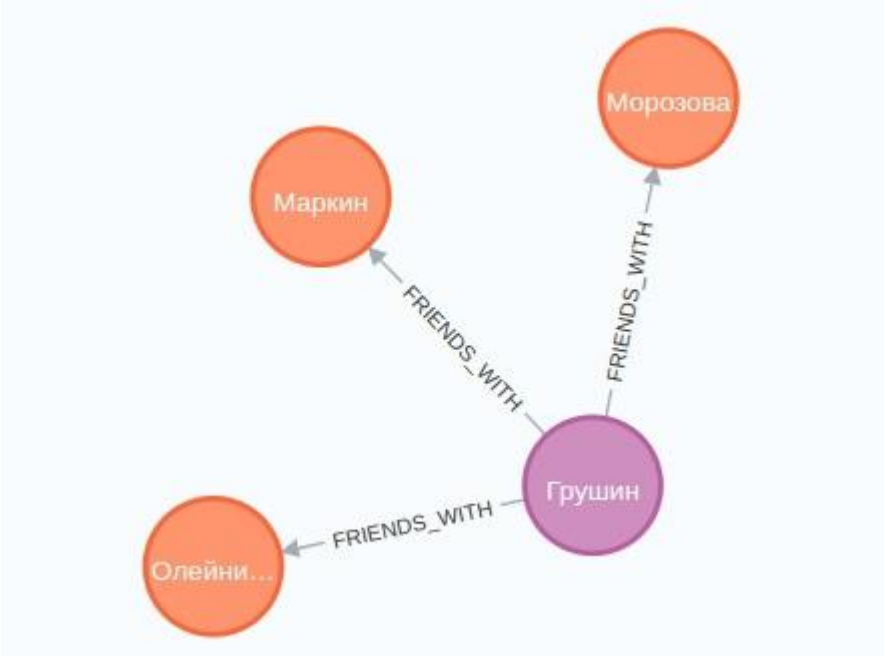

Рисунок 1. Связи между узлами

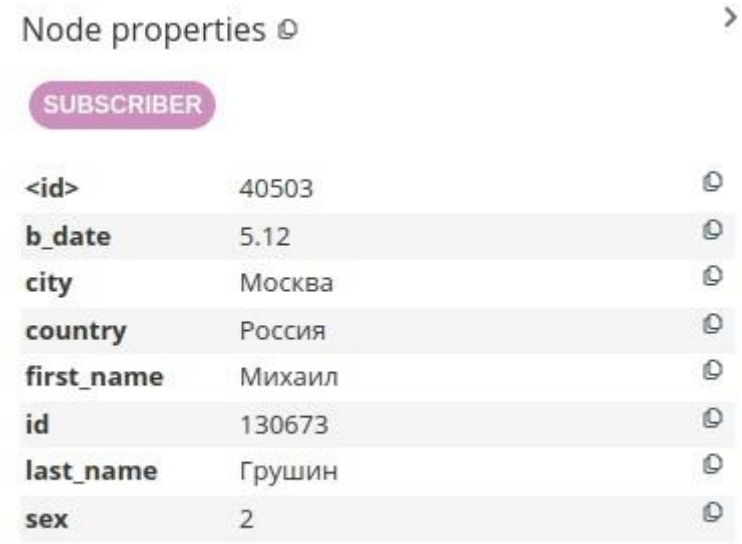

Рисунок 2. Атрибуты узла

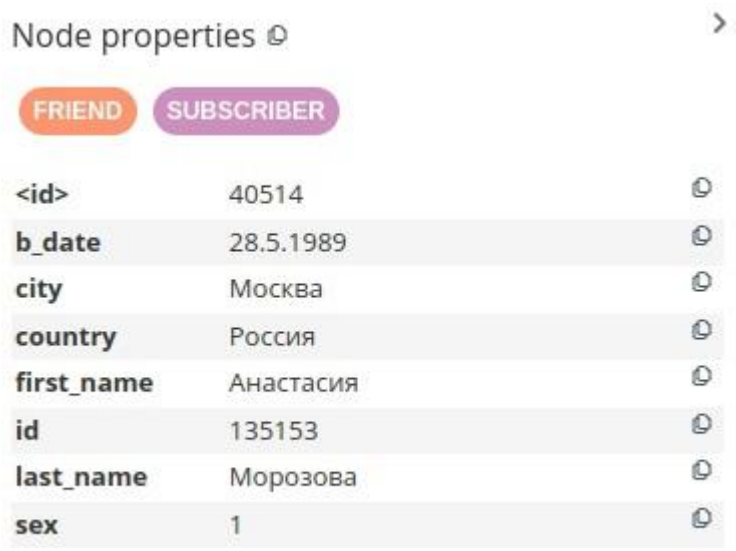

Рисунок 3. Атрибуты узла

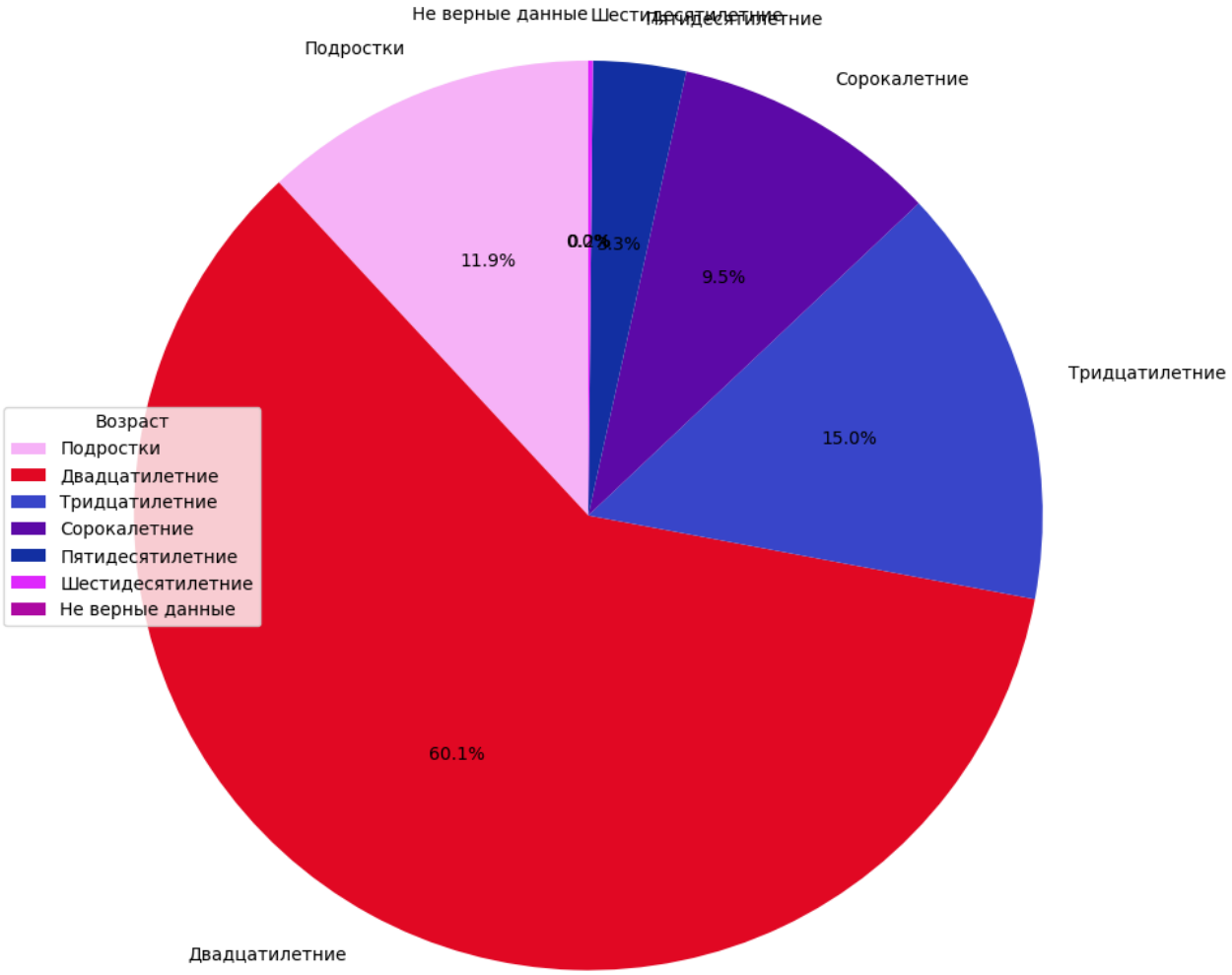

Рисунок 4. Принадлежность пользователей к возрастным группам

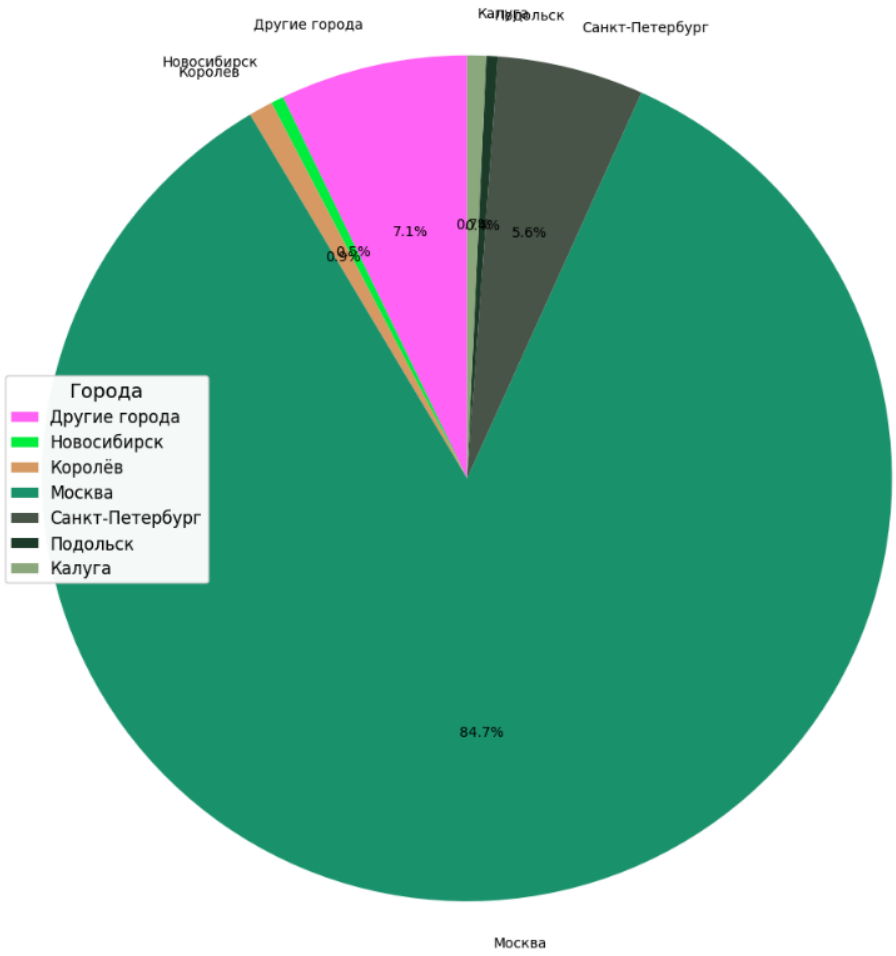

Рисунок 5. Диаграмма распределения городов

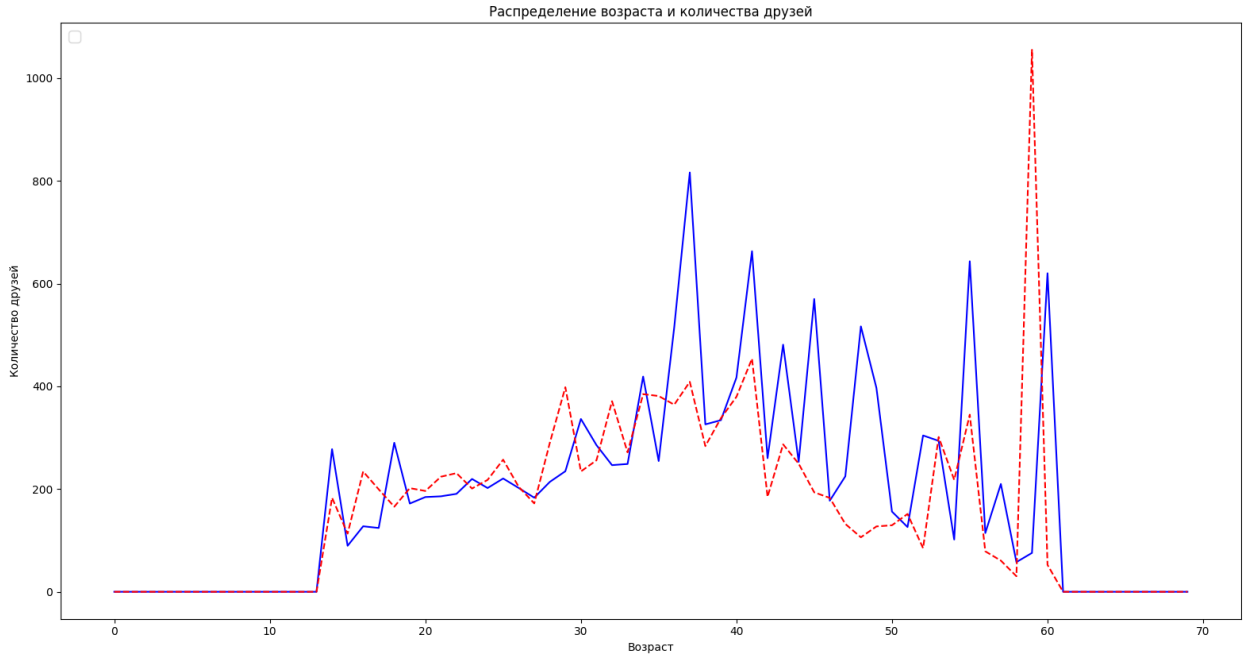

Рисунок 6. Распределение количества друзей по возрасту

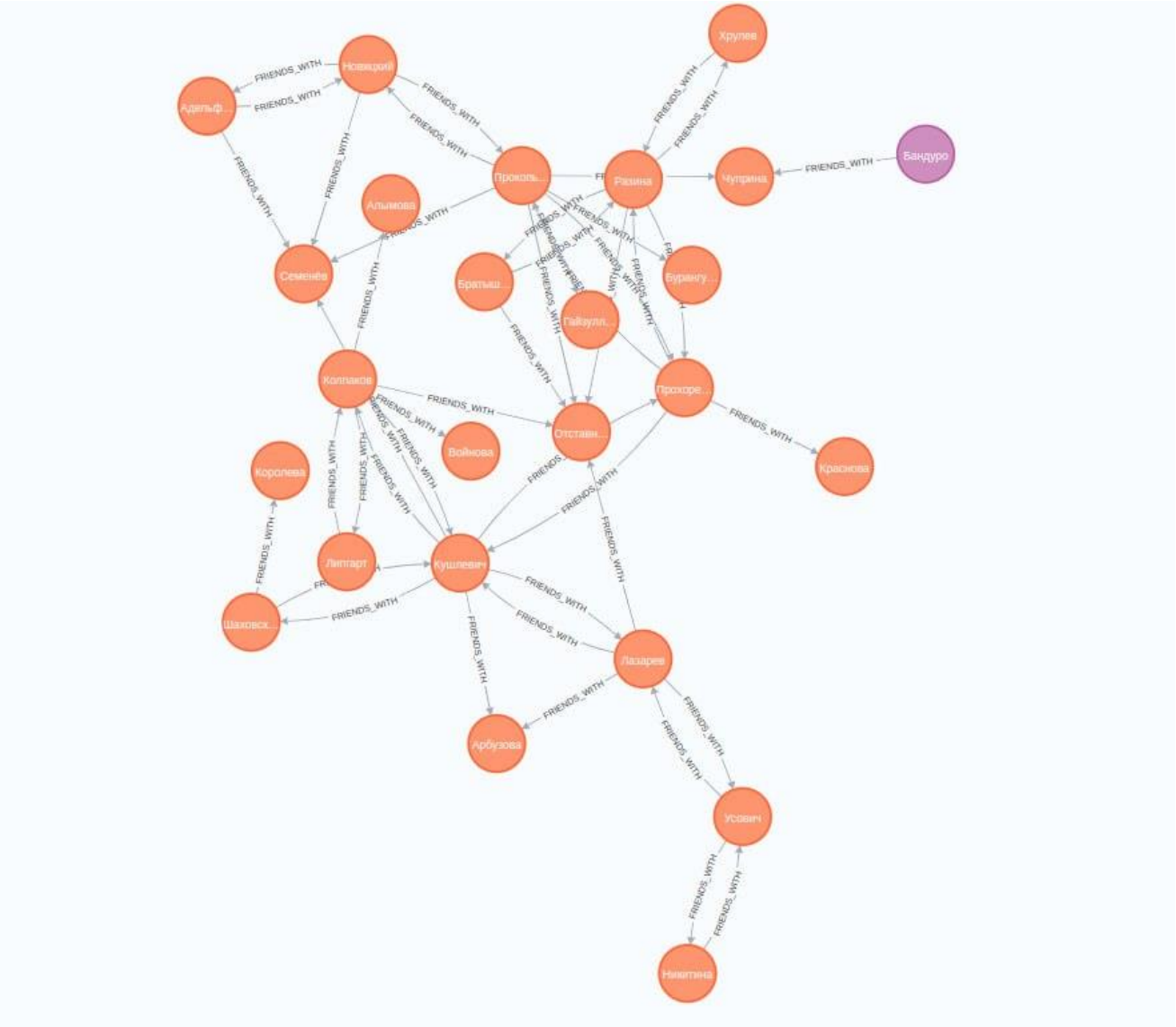

Рисунок 7. Цельная цепочка знакомств

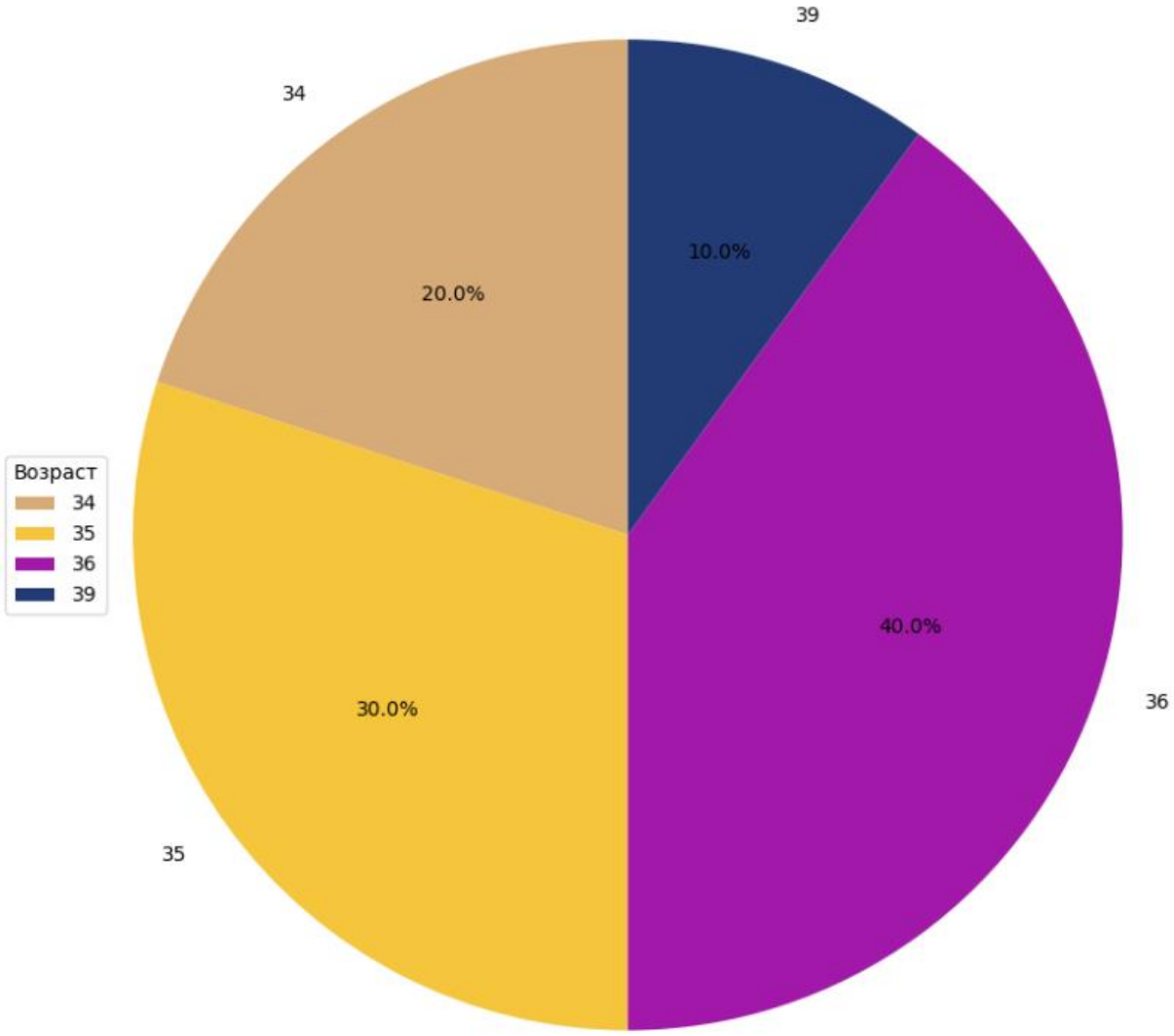

Рисунок 8. Распределение возраста у цельной цепочки пользователей