Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационный технологий и управления

Кафедра интеллектуальных информационных технологий

Дисциплина: «Проектировка баз знаний»

Лабораторная работа №3 по теме: «Создание базы данных и запросов к ней с помощью средств графовой СУБД neo4j»

Студент гр. 121702

Заломов Р.А.

Проверила:

Липницкая Н. Г.

Минск 2023

Тема

Изучение средств создания базы данных и выполнения запросов к ней с использованием графовой СУБД neo4j.

Цель

Получить навыки создания базы данных и выполнения запросов к ней с помощью средств графовой СУБД neo4j.

Задание

1) Сформировать базу данных по выбранной предметной области.
2) Составить список из 10 запросов к базе данных.
3) С помощью шаблонов запросов получить выборку для каждого запроса из п.2).
4) В отчёте отразить в графической или текстовой форме содержимое базы данных, шаблоны запросов и полученные выборки с комментариями.

Выполнение задания

Создание базы данных

Создание базы данных было автоматизировано при помощи языка программирования Python и модуля neo4j для него.

```
import pandas as pd
from neo4j import GraphDatabase
from neo4j.exceptions import DriverError, Neo4jError
Executed at 2023.11.08 23:01:04 in 3ms
```

```
df = pd.read_csv('SP_500_ESG_Risk_Ratings.csv')
df.dropna(inplace=True)
df.drop(['ESG Risk Percentile', 'Description'], inplace=True, axis=1)
df
Executed at 2023.11.08 22:55:13 in 26ms
```

```
class CompaniesFiller:
    def __init__(self, uri, user, password, database = None):
        self.driver = GraphDatabase.driver(uri, auth=(user, password))
        self.database = database

    def create_companies(self, companies_df: pd.DataFrame):
        with self.driver.session() as session:
            for row in companies_df.iterrows():
                result = self._create_and_return_company(row[1])
                print(f'Created company: {result}')


    def _create_and_return_company(self, company_data):
        query = (
            "CREATE (cmp:Company { name: $company_name }) "
            "CREATE (sym:Symbol { name: $symbol }) "
            "CREATE (cmp)-[:SYMBOL]->(sym) "
            "CREATE (adr:Address { name: $address }) "
            "CREATE (cmp)-[:ADDRESS]->(adr) "
            "CREATE (sct:Sector { name: $sector }) "
            "CREATE (cmp)-[:SECTOR]->(sct) "
            "CREATE (ind:Industry { name: $industry }) "
            "CREATE (cmp)-[:INDUSTRY]->(ind) "
            "CREATE (emp:Number {value: $emp_amount}) "
            "CREATE (cmp)-[:EMPLOYEE_AMOUNT]->(emp) "
            "CREATE (esg_r_score:Number { value: $risk_score }) "
            "CREATE (cmp)-[:RISK_SCORE]->(esg_r_score) "
            "CREATE (ct_level:Controversy_Level { value: $controversy_level }) "
            "CREATE (cmp)-[:CONTROVERSY_LEVEL]->(ct_level) "
            "CREATE (ct_score:Number { value: $controversy_score }) "
            "CREATE (cmp)-[:CONTROVERSY_SCORE]->(ct_score) "
            "RETURN cmp.name"
        )
        try:
            record = self.driver.execute_query(
                query_=query,
                company_name=company_data['Name'],
                symbol=company_data['Symbol'],
                address=company_data['Address'],
                sector=company_data['Sector'],
                industry=company_data['Industry'],
                emp_amount=int(company_data['Full Time Employees'].replace(',', '')),
                risk_score=company_data['Total ESG Risk score'],
                controversy_level=company_data['Controversy Level'],
                controversy_score=company_data['Controversy Score'],
```

```
                database_=self.database,
                result_transformer_=lambda r: r.single(strict=True)
            )
            return record['cmp.name']
        except (DriverError, Neo4jError) as err:
            print(f'Error during query: {err}')

    def close(self):
        self.driver.close()
```
Executed at 2023.11.09 00:15:14 in 5ms

```
URI = 'bolt://localhost:7687'
USER = 'neo4j'
PASSWORD = 'ilikepkb'
DATABASE = 'companies'

db_filler = CompaniesFiller(URI, USER, PASSWORD, DATABASE)
```
Executed at 2023.11.09 00:15:15 in 4ms

```
try:
    db_filler.create_companies(df)
finally:
    db_filler.close()
```
Executed at 2023.11.09 00:15:19 in 3s 559ms

Источником данных послужила таблица с различными компаниями, а также их характеристиками. Таблица доступна по ссылке: https://www.kaggle.com/datasets/pritish509/s-and-p-500-esg-risk-ratings

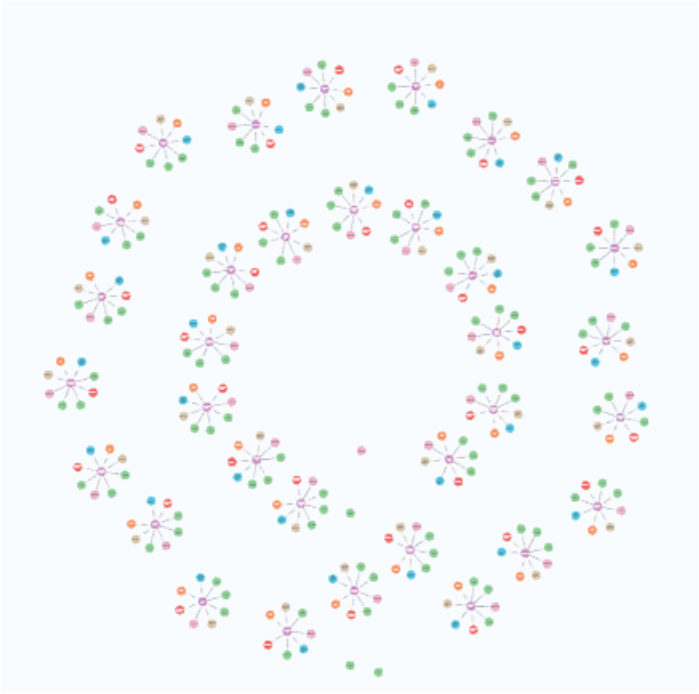
Написание запросов к базе данных

1. Выбрать абсолютно все вершины

   Запрос:

   ```
   //SELECT ABSOLOUTELY ALL
   MATCH (n) RETURN n
   ```

   Результат:



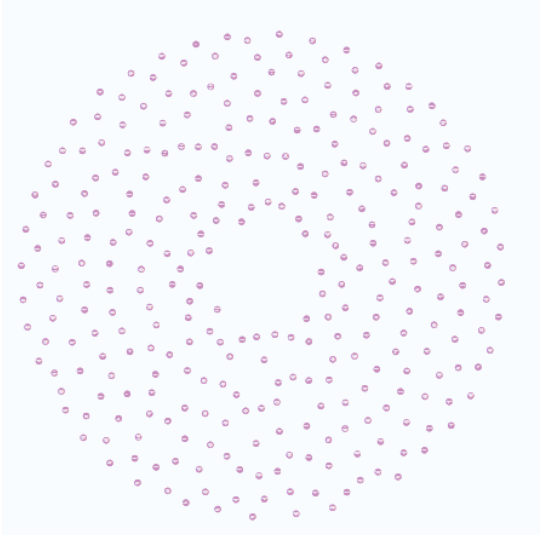
Примечание: при просмотре результатов запросов в режиме графа для обеспечения оптимизации neo4j отображает не более 300 вершин.

2. Выбрать только вершины компаний

   Запрос:

```
//SELECT ONLY COMPANIES
MATCH (cmp:Company)
RETURN cmp
```

   Результат:



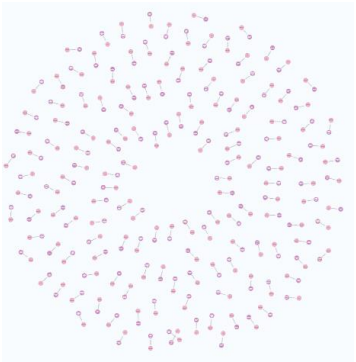
cmp
(:Company {name: "Best Buy Co Inc"})
(:Company {name: "Becton Dickinson and Co"})
(:Company {name: "Franklin Resources Inc"})
(:Company {name: "Bunge Ltd"})
(:Company {name: "Biogen Inc"})
(:Company {name: "Bio Rad Laboratories A"})
(:Company {name: "Bank of New York Mellon Corp"})
(:Company {name: "Booking Holdings Inc"})
(:Company {name: "Blackrock Inc"})
(:Company {name: "Bristol Myers Squibb Co"})

3. Выбрать те, компании, в которых уровень споров низкий (Low) или средний (Moderate)

Запрос:

```
//GET COMPANIES WHERE CONTROVERSY LEVEL IS LOW OR MODERATE
MATCH (cmp:Company)-[:CONTROVERSY_LEVEL]→(lvl:Controversy_Level)
WHERE lvl.value IN ['Low', 'Moderate']
RETURN cmp, lvl
```

Результат:



cmp	lvl
(:Company {name: "Bath & Body Works Inc"})	(:Controversy_Level {value: "Moderate"})
(:Company {name: "Best Buy Co Inc"})	(:Controversy_Level {value: "Moderate"})
(:Company {name: "Franklin Resources Inc"})	(:Controversy_Level {value: "Moderate"})
(:Company {name: "Bio Rad Laboratories A"})	(:Controversy_Level {value: "Low"})
(:Company {name: "Blackrock Inc"})	(:Controversy_Level {value: "Moderate"})
(:Company {name: "Bristol Myers Squibb Co"})	(:Controversy_Level {value: "Moderate"})
(:Company {name: "Broadridge Financial Solutio"})	(:Controversy_Level {value: "Moderate"})
(:Company {name: "Brown & Brown Inc"})	(:Controversy_Level {value: "Low"})
(:Company {name: "Boston Scientific Corp"})	(:Controversy_Level {value: "Moderate"})
(:Company {name: "Borgwarner Inc"})	(:Controversy_Level {value: "Moderate"})

4. Получить средний уровень риска в компаниях:

   Запрос:

```
//GET AVERAGE RISK SCORE
MATCH (cmp:Company)-[:RISK_SCORE]→(score:Number)
RETURN avg(score.value) as avg_score
```

   Результат:

```
╒═══════════════════╕
│avg_score          │
╞═══════════════════╡
│21.48414985590779  │
└───────────────────┘
```

5. Получить минимальное количество работников в компаниях

   Запрос:

```
//GET MINIMAL AMOUNT OF EMPLOYEES
MATCH (cmp:Company)-[:EMPLOYEE_AMOUNT]→(emp:Number)
RETURN min(emp.value) as min_emp
```

   Результат:

```
╒═══════╕
│min_emp│
╞═══════╡
│165    │
└───────┘
```

6. Выбрать компании, в именах которых содержится слово «American»

   Запрос:

```
//MATCH COMPANIES WHICH NAMES CONTAIN 'American'
MATCH (cmp:Company)
WHERE cmp.name CONTAINS 'American'
RETURN cmp
```

   Результат:

```
╒═══════════════════════════════════════════╕
│cmp                                        │
╞═══════════════════════════════════════════╡
│(:Company {name: "American Tower Corp"})   │
├───────────────────────────────────────────┤
│(:Company {name: "American Express Co"})   │
└───────────────────────────────────────────┘
```

7. Выбрать компании, в которых работает меньше 20000 сотрудников

Запрос:

```
//MATCH COMPANIES WITH LESS THAN 20000 EMPLOYEES
MATCH (cmp:Company)-[:EMPLOYEE_AMOUNT]→(emp:Number)
WHERE emp.value < 20000
RETURN cmp, emp
```

Результат:

cmp	emp
(:Company {name: "Franklin Resources Inc"})	(:Number {value: 9300})
(:Company {name: "Biogen Inc"})	(:Number {value: 8725})
(:Company {name: "Bio Rad Laboratories A"})	(:Number {value: 8200})
(:Company {name: "Blackrock Inc"})	(:Number {value: 19300})
(:Company {name: "Broadridge Financial Solutio"})	(:Number {value: 14700})
(:Company {name: "Brown & Brown Inc"})	(:Number {value: 15201})
(:Company {name: "Boston Properties Inc"})	(:Number {value: 780})
(:Company {name: "Conagra Brands Inc"})	(:Number {value: 18600})
(:Company {name: "Cboe Global Markets Inc"})	(:Number {value: 1543})
(:Company {name: "Celanese Corp"})	(:Number {value: 13263})

8. Выбрать компании, в которых уровень риска ниже среднего по компаниям

Запрос:

```
//GET COMPANIES WITH RISK SCORE LOWER THAN AVERAGE
CALL {
    MATCH (cmp:Company)-[:RISK_SCORE]→(score:Number)
    RETURN avg(score.value) AS avg_risk
}
MATCH (cmp)-[:RISK_SCORE]→(score)
WHERE score.value < avg_risk
RETURN cmp, score
```

Результат:

cmp	score
(:Company {name: "Agilent Technologies Inc"})	(:Number {value: 15.0})
(:Company {name: "Advance Auto Parts Inc"})	(:Number {value: 12.0})
(:Company {name: "Apple Inc"})	(:Number {value: 17.0})
(:Company {name: "Amerisourcebergen Corp"})	(:Number {value: 12.0})
(:Company {name: "Accenture Plc Cl A"})	(:Number {value: 10.0})
(:Company {name: "Adobe Inc"})	(:Number {value: 12.0})
(:Company {name: "Automatic Data Processing"})	(:Number {value: 14.0})
(:Company {name: "Aflac Inc"})	(:Number {value: 17.0})
(:Company {name: "Arthur J Gallagher & Co"})	(:Number {value: 21.0})
(:Company {name: "Akamai Technologies Inc"})	(:Number {value: 17.0})

9. Получить средний уровень риска в компаниях

   Запрос:

```
//GET AVERAGE RISK SCORE
MATCH (cmp:Company)-[:RISK_SCORE]→(score:Number)
RETURN avg(score.value) as avg_score
```

   Результат:

avg_score
21.48414985590779

10. Выбрать те, компании, которые занимаются планами в здравоохранении (Healthcare Plans) или средствами связи (Communication Equipment), в которых степень риска меньше, чем поделённая пополам максимальная степень рискам по компаниям, а также в которых работают больше человек, чем стократное минимальное количество

    Запрос:

```
//GET COMPANIES WHICH INDUSTRY IS HEALTHCARE PLANS OR COMMUNICATION EQUIPMENT, EMPLOYEE AMOUNT MO
DIVIDED BY TWO
CALL {
    MATCH (cmp:Company)-[:EMPLOYEE_AMOUNT]→(emp:Number)
    RETURN min(emp.value) * 100 AS min_emp_hun
}
CALL {
    MATCH (cmp)-[:RISK_SCORE]→(score:Number)
    RETURN max(score.value) / 2 AS score_twice
}
MATCH (cmp:Company)-[:INDUSTRY]→(ind:Industry), (cmp:Company)-[:EMPLOYEE_AMOUNT]→(emp:Number),
(cmp:Company)-[:RISK_SCORE]→(score:Number)
WHERE
ind.name in ['Communication Equipment', 'Healthcare Plans']
AND
emp.value > min_emp_hun
AND
score.value < score_twice
RETURN cmp, ind, score, emp
```

Результат:

cmp	ind	score	emp
(:Company {name: "The Cigna Group"})	(:Industry {name: "Healthcare Plans"})	(:Number {value: 12.0})	(:Number {value: 71300})
(:Company {name: "Centene Corp"})	(:Industry {name: "Healthcare Plans"})	(:Number {value: 21.0})	(:Number {value: 74300})
(:Company {name: "Elevance Health Inc"})	(:Industry {name: "Healthcare Plans"})	(:Number {value: 11.0})	(:Number {value: 102300})
(:Company {name: "Hewlett Packard Enterprise"})	(:Industry {name: "Communication Equipment"})	(:Number {value: 12.0})	(:Number {value: 60200})
(:Company {name: "Humana Inc"})	(:Industry {name: "Healthcare Plans"})	(:Number {value: 22.0})	(:Number {value: 67100})
(:Company {name: "Motorola Solutions Inc"})	(:Industry {name: "Communication Equipment"})	(:Number {value: 14.0})	(:Number {value: 20000})
(:Company {name: "Unitedhealth Group Inc"})	(:Industry {name: "Healthcare Plans"})	(:Number {value: 18.0})	(:Number {value: 400000})

Вывод

В лабораторной работе были изучены основы работы с графовыми СУБД на примере Neo4j. Были предприняты решения по автоматизации создания базы данных на примере языка программирования Python и библиотеки neo4j. Помимо этого был рассмотрен язык запросов Cypher, были написаны различные запросы при помощи данного языка запросов.