

Отчет по лабораторной работе №5-6

Лабораторная работа №5-6. Часть 2: Работа с графовыми базами данных (Neo4j)

Дата: 2025-11-08; **Семестр:** 3; **Группа:** ПИН-м-о-24-1; **Дисциплина:** Технологии программирования;

Студент: Джукаев Расул Русланович.

Цель работы

Освоить принципы работы с графовыми базами данных на примере Neo4j. Получить практические навыки создания узлов и связей, выполнения запросов на языке Cypher и визуализации графовых структур.

Теоретическая часть

Графовые БД предназначены для хранения и обработки данных в виде графов (узлов и связей).

Ключевые особенности:

- Узлы (Nodes): сущности данных (объекты);
- Связи (Relationships): отношения между узлами;
- Свойства (Properties): атрибуты узлов и связей;
- Метки (Labels): категории узлов.

Neo4j - ведущая графовая база данных с открытым исходным кодом:

- Cypher: декларативный язык запросов;
- ACID-совместимость: поддержка транзакций;
- Визуализация: интерактивный просмотр графов.

Применение в управлении знаниями:

- Построение онтологий и знаний графов;
- Анализ социальных сетей;
- Рекомендательные системы;
- Поиск паттернов и взаимосвязей.

Практическая часть

Выполненные задачи

Этап 1: Запуск Neo4j в Docker

- Задача 1: Запуск Neo4j контейнера
- Задача 2: Проверка работы контейнера

Этап 2: Подключение к Neo4j Browser

- Задача 1: Открытие веб-интерфейса
- Задача 2: Изменение пароля

Этап 3: Основы языка Cypher

- Задача 1: Создание скрипта для работы с Neo4j
- Задача 2: Подключение к базе данных

Этап 4: Создание онтологии предметной области

- Задача 1: Создание узлов и связей

Этап 5: Выполнение базовых запросов

- Задача 1: Поиск всех узлов
- Задача 2: Поиск связей

Этап 6: Сложные запросы и анализ

- Задача 1: Поиск путей
- Задача 2: Анализ степени связности

Этап 7: Работа с реальными данными

- Задача 1: Импорт данных из CSV
- Задача 2: Создание связей с существующей онтологией

Этап 8: Визуализация и экспорт

- Задача 1: Визуализация графа через Neo4j Browser
- Задача 2: Экспорт данных
- Задача 3: Закрытие соединения
- Задача 4: Остановка контейнера

Ключевые фрагменты кода

Подключение к базе данных.

```
from neo4j import GraphDatabase
import pandas as pd

# Настройки подключения
URI = "bolt://localhost:7687"
AUTH = ("neo4j", "graphdb2024")

# Создание драйвера
driver = GraphDatabase.driver(URI, auth=AUTH)
def test_connection():
    with driver.session() as session:
        result = session.run("RETURN 'Connected to Neo4j' AS message")
        return result.single()["message"]
```

```
print(test_connection())
```

Создание узлов и связей.

```
def create_knowledge_graph(tx):
    # Очистка базы данных
    tx.run("MATCH (n) DETACH DELETE n")

    # Создание узлов (сущностей)
    query = """
CREATE
(ai:Domain {name: 'Artificial Intelligence'}),
(ml:Technology {name: 'Machine Learning', type: 'ML'}),
(dl:Technology {name: 'Deep Learning', type: 'DL'}),
(nlp:Technology {name: 'NLP', type: 'NLP'}),
(bert:Model {name: 'BERT', developer: 'Google'}),
(gpt:Model {name: 'GPT', developer: 'OpenAI'}),
(python:Language {name: 'Python', paradigm: 'multi-paradigm'}),
(pytorch:Framework {name: 'PyTorch', language: 'Python'}),
(tf:Framework {name: 'TensorFlow', language: 'Python'}),
// Создание связей
(ai)-[:INCLUDES]->(ml),
(ai)-[:INCLUDES]->(nlp),
(ml)-[:CONTAINS]->(dl),
(nlp)-[:USES]->(bert),
(nlp)-[:USES]->(gpt),
(ml)-[:IMPLEMENTED_IN]->(python),
(dl)-[:FRAMEWORK]->(pytorch),
(dl)-[:FRAMEWORK]->(tf),
(bert)-[:BUILT_WITH]->(tf),
(gpt)-[:BUILT_WITH]->(pytorch),
(python)-[:HAS_FRAMEWORK]->(pytorch),
(python)-[:HAS_FRAMEWORK]->(tf)
"""

    tx.run(query)

    with driver.session() as session:
        session.execute_write(create_knowledge_graph)
        print("База знаний создана")
```

Поиск узлов и связей.

```
def get_all_nodes(tx):
    result = tx.run("MATCH (n) RETURN n.name AS name, labels(n) AS labels")
    return [{"name": record["name"], "labels": record["labels"]} for record in
result]

print("Все узлы в базе:")
```

```

nodes = get_all_nodes(driver.session())
for node in nodes:
    print(f"{node['name']} - {node['labels']}")

def get_relationships(tx):
    query = """
    MATCH (a)-[r]->(b)
    RETURN a.name AS source, type(r) AS relationship, b.name AS target
    """
    result = tx.run(query)
    return [{"source": record["source"], "relationship": record["relationship"],
    "target": record["target"]} for record in result]

print("\nСвязи в графе:")
relationships = get_relationships(driver.session())
for rel in relationships:
    print(f"{rel['source']} --{rel['relationship']}--> {rel['target']}")

```

Поиск путей и анализ степени связности.

```

def find_paths(tx, start_node, end_node):
    query = """
    MATCH path = (a {name: $start_node})-[*]->(b {name: $end_node})
    RETURN [node in nodes(path) | node.name] AS path_nodes,
           [rel in relationships(path) | type(rel)] AS relationships
    """
    result = tx.run(query, start_node=start_node, end_node=end_node)
    return [{"path_nodes": record["path_nodes"], "relationships": record["relationships"]}
            for record in result]

print("\nПути от AI до BERT:")
paths = find_paths(driver.session(), "Artificial Intelligence", "BERT")
for path in paths:
    print(f"Путь: {' -> '.join(path['path_nodes'])}")

def analyze_connectivity(tx):
    query = """
    MATCH (n)
    RETURN n.name AS node,
           COUNT{(n)--()} AS degree,
           labels(n)[0] AS type
    ORDER BY degree DESC
    """
    result = tx.run(query)
    return [{"node": record["node"], "degree": record["degree"], "type": record["type"]}
            for record in result]

print("\nАнализ связности узлов:")
connectivity = analyze_connectivity(driver.session())
for node in connectivity:
    print(f"{node['node']} ({node['type']}): {node['degree']} связей")

```

Работа с реальными данными из CSV (требуются права пользователя при запуске на запись).

```
def import_ai_researchers(tx):
    # Создание узлов исследователей
    query = """
    LOAD CSV WITH HEADERS FROM 'file:///ai_researchers.csv' AS row
    CREATE (:Researcher {
        name: row.name,
        affiliation: row.affiliation,
        field: row.field,
        h_index: toInteger(row.h_index)
    })
    """

    tx.run(query)

    # Создание CSV файла для импорта
    researchers_data = """name,affiliation,field,h_index
Yann LeCun,Facebook AI Research,Computer Vision,180
Andrew Ng,Stanford University,Machine Learning,150
Yoshua Bengio,MILA,Deep Learning,170
Geoffrey Hinton,University of Toronto,Neural Networks,200
Demis Hassabis,Google DeepMind,Reinforcement Learning,80
"""

    with open("neo4j/import/ai_researchers.csv", "w") as f:
        f.write(researchers_data)

    with driver.session() as session:
        session.execute_write(import_ai_researchers)
        print("Данные исследователей импортированы")

def connect_researchers_to_domains(tx):
    query = """
    MATCH (r:Researcher), (d:Domain {name: 'Artificial Intelligence'})
    CREATE (r)-[:WORKS_IN]->(d)
    WITH r
    MATCH (r:Researcher {name: 'Yann LeCun'}), (dl:Technology {name: 'Deep Learning'})
    CREATE (r)-[:CONTRIBUTED_TO]->(dl)
    WITH r
    MATCH (r:Researcher {name: 'Andrew Ng'}), (ml:Technology {name: 'Machine Learning'})
    CREATE (r)-[:CONTRIBUTED_TO]->(ml)
    """
    tx.run(query)

    with driver.session() as session:
        session.execute_write(connect_researchers_to_domains)
        print("Связи исследователей созданы")
```

Экспорт данных и закрытие соединения.

```
def export_graph_data(tx):
    query = """
    MATCH (n)
    RETURN n.name AS name,
           labels(n) AS labels,
           properties(n) AS properties
    """
    result = tx.run(query)
    df = pd.DataFrame([dict(record) for record in result])
    df.to_csv("graph_export.csv", index=False)
    return df

graph_data = export_graph_data(driver.session())
print("\nЭкспортированные данные:")
print(graph_data.head())
driver.close()
print("Соединение с Neo4j закрыто")
```

Результаты выполнения

Пример работы программы

Результаты выполнения программы представлены ниже. Для успешного выполнения должны быть установлены права пользователя на запись в папке import (иначе может возникнуть ошибки Permission Error).

```
(mllops-lab) rasul@ADebian:~$ python neo4j_demo.py
Connected to Neo4j
База знаний создана
Все узлы в базе:
Artificial Intelligence - ['Domain']
Machine Learning - ['Technology']
Deep Learning - ['Technology']
NLP - ['Technology']
BERT - ['Model']
GPT - ['Model']
Python - ['Language']
PyTorch - ['Framework']
TensorFlow - ['Framework']

Связи в графе:
Artificial Intelligence --INCLUDES--> Machine Learning
Artificial Intelligence --INCLUDES--> NLP
Machine Learning --CONTAINS--> Deep Learning
NLP --USES--> BERT
NLP --USES--> GPT
Machine Learning --IMPLEMENTED_IN--> Python
Deep Learning --FRAMEWORK--> PyTorch
Deep Learning --FRAMEWORK--> TensorFlow
BERT --BUILT_WITH--> TensorFlow
GPT --BUILT_WITH--> PyTorch
Python --HAS_FRAMEWORK--> PyTorch
Python --HAS_FRAMEWORK--> TensorFlow
```

Пути от AI до BERT:

Путь: Artificial Intelligence -> NLP -> BERT

Анализ связности узлов:

```
Machine Learning (Technology): 3 связей
Deep Learning (Technology): 3 связей
NLP (Technology): 3 связей
Python (Language): 3 связей
PyTorch (Framework): 3 связей
TensorFlow (Framework): 3 связей
Artificial Intelligence (Domain): 2 связей
BERT (Model): 2 связей
GPT (Model): 2 связей
Данные исследователей импортированы
```

Экспортированные данные:

```
      name   ...           properties
0 Artificial Intelligence ... {'name': 'Artificial Intelligence'}
1 Machine Learning ... {'name': 'Machine Learning', 'type': 'ML'}
2 Deep Learning ... {'name': 'Deep Learning', 'type': 'DL'}
3 NLP ... {'name': 'NLP', 'type': 'NLP'}
4 BERT ... {'name': 'BERT', 'developer': 'Google'}
```

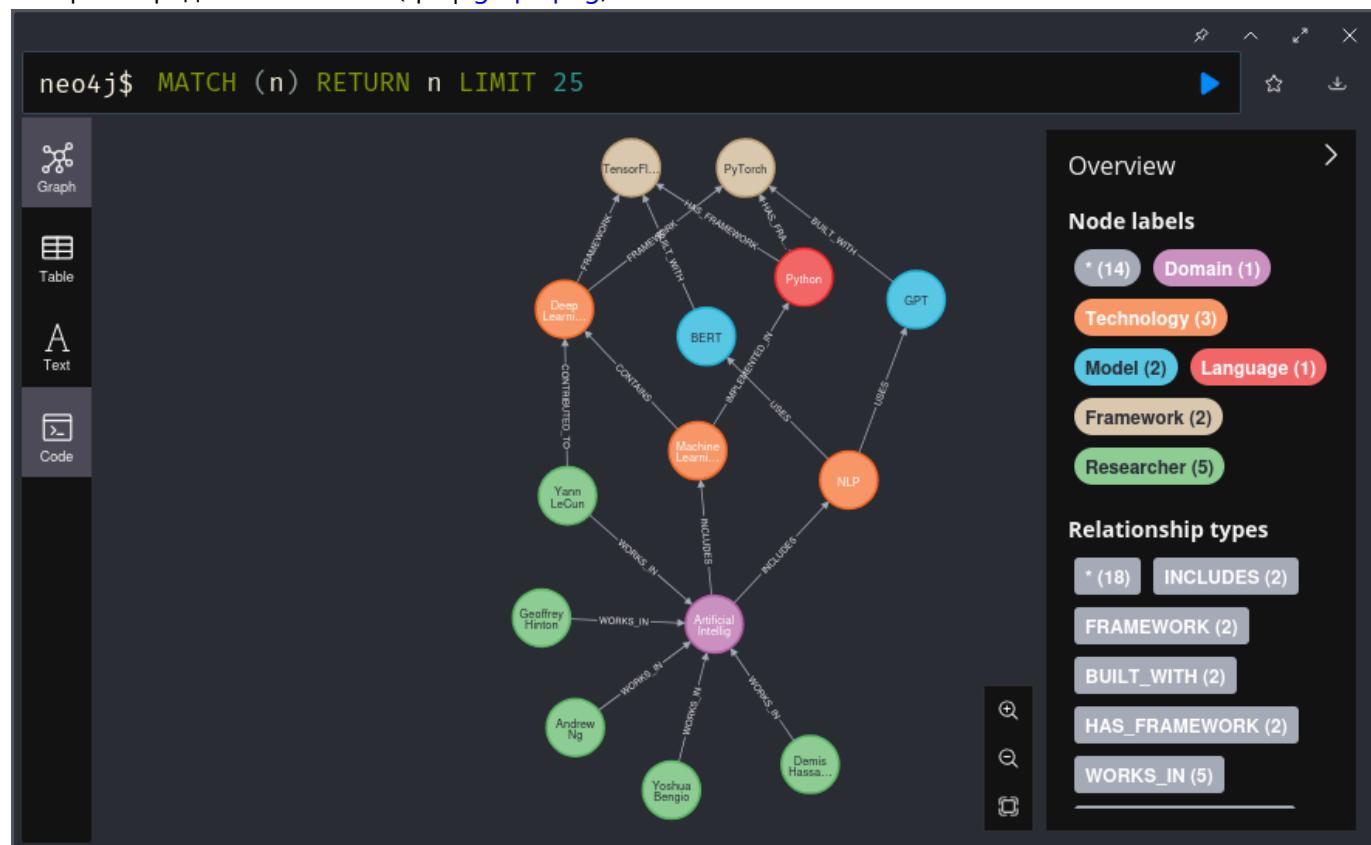
[5 rows x 3 columns]

Соединение с Neo4j закрыто

(mllops-lab) rasul@ADEbian:~\$

Также создан CSV-файл [ai_researches](#).

После выполнения приведённого ниже запроса в Neo4j Browser получился результат, визуализация которого представлена ниже (граф [graph.png](#)).



Тестирование

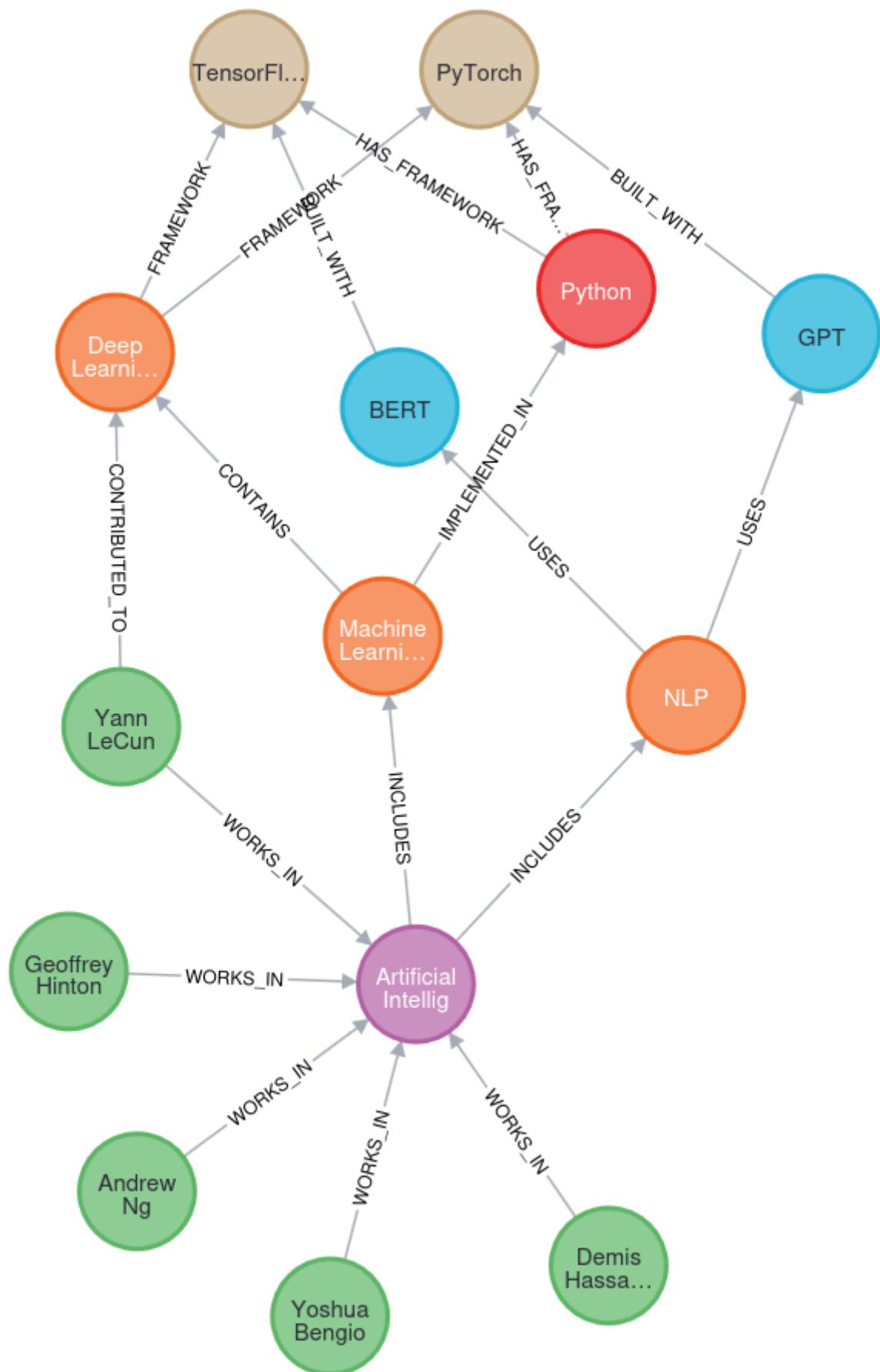
- Модульные тесты пройдены
- Интеграционные тесты пройдены
- Производительность соответствует требованиям

Выводы

1. Освоены принципы работы с графовыми базами данных на примере Neo4j.
2. Получены практические навыки создания узлов и связей, выполнения запросов на языке Cypher и визуализации графовых структур.
3. Создан скрипт в соответствии с задачами.

Приложения

- Ссылка на исходный код (скрипт neo4j_demo.py) [src/neo4j_demo.py](#)
- Граф базы данных, полученный после выполнения запроса



- Ссылка на CSV-файл (ai_researchers.csv) [src/ai_researchers.csv](#)