Лабораторная работа: "Изучение методов хранения данных на основе NoSQL в MongoDB, Redis, Neo4j"

Цель работы:
Изучить и освоить методы хранения и работы с данными в NoSQL базах данных MongoDB, Redis и Neo4j. Научиться загружать данные из CSV файлов в указанные СУБД и выполнять базовые операции по работе с данными.

Оборудование и ПО:
- Операционная система Ubuntu
- Установленные пакеты для работы с NoSQL базами данных: MongoDB, Redis, Neo4j
- Язык программирования Python (с библиотеками pymongo, redis, neo4j)
- CSV файлы с данными

Теоретическая часть:
1. **MongoDB:** документо-ориентированная NoSQL база данных, где данные хранятся в формате JSON-подобных документов.
2. **Redis:** высокопроизводительная база данных типа "ключ-значение", часто используемая для кеширования и временного хранения данных.
3. **Neo4j:** графовая база данных, которая позволяет хранить данные в виде вершин и рёбер графа, что удобно для моделирования сложных взаимосвязей.

Ход работы:

Шаг 1: Установка и настройка MongoDB

1.1. Установите MongoDB:
```bash
sudo apt-get update
sudo apt-get install -y mongodb
```

1.2. Запустите MongoDB:
```bash
sudo systemctl start mongodb
```

Подключение к MongoDB с аутентификацией

Когда MongoDB настроена на требование аутентификации, вам нужно указать имя пользователя и пароль для доступа к базе данных. Библиотека `pymongo` позволяет сделать это довольно просто.

Установка библиотеки `pymongo`

Если библиотека `pymongo` еще не установлена, установите ее с помощью `pip`:

```bash
pip install pymongo
```

Чтобы проверить соединение с MongoDB с использованием Python и библиотеки `pymongo`, можно выполнить следующие шаги:

1. Установка библиотеки `pymongo`
Если библиотека `pymongo` еще не установлена, установите ее:

```bash
pip install pymongo
```

Для подключения к MongoDB из JupyterLab, создания тестовых данных и их загрузки в коллекцию `labs`, при условии, что требуется аутентификация, следуйте следующим шагам. Также добавим проверку успешного подключения.

1. Установка необходимых библиотек

Вначале убедитесь, что у вас установлена библиотека `pymongo`. Если нет, выполните следующую команду:

```python
!pip install pymongo
```

2. Подключение к MongoDB с аутентификацией

Предположим, что у вас есть следующие параметры для подключения:

- **URI MongoDB**: `mongodb://localhost:27017/`
- **Имя базы данных**: `test_db`
- **Имя пользователя**: `your_username`
- **Пароль**: `your_password`
- **Имя коллекции (таблицы)**: `labs`

Создадим подключение и проверим его успешность.

```python
```

```
from pymongo import MongoClient

# Задайте параметры подключения
mongo_uri =
"mongodb://your_username:your_password@localhost:27017/test_db"

try:
    # Подключение к MongoDB
    client = MongoClient(mongo_uri)

    # Проверка подключения
    client.admin.command('ping')
    print("Подключение к MongoDB установлено успешно!")

    # Выбор базы данных
    db = client['test_db']

    # Выбор коллекции
    labs_collection = db['labs']

except Exception as e:
    print(f"Ошибка подключения: {e}")
```

3. Создание тестовых данных

Создадим пример тестовых данных:

```python
# Пример тестовых данных
test_data = [
    {"lab_name": "Lab 1", "subject": "Physics", "date": "2024-08-28", "score":
85},
    {"lab_name": "Lab 2", "subject": "Chemistry", "date": "2024-08-29", "score":
90},
    {"lab_name": "Lab 3", "subject": "Biology", "date": "2024-08-30", "score":
88},
]
```

4. Загрузка данных в коллекцию `labs`

Теперь загрузим созданные тестовые данные в коллекцию `labs`:

```python
try:
```

```
    # Вставка данных в коллекцию
    result = labs_collection.insert_many(test_data)

    # Вывод идентификаторов вставленных документов
    print("Данные успешно загружены в коллекцию 'labs'.")
    print("Идентификаторы вставленных документов:", result.inserted_ids)
except Exception as e:
    print(f"Ошибка при загрузке данных: {e}")
```

5. Проверка подключения и данных

Чтобы убедиться, что подключение прошло успешно и данные загружены правильно, выполните запрос на выборку всех документов из коллекции:

```python
try:
    # Получение всех документов из коллекции
    all_docs = labs_collection.find()

    # Вывод всех документов
    print("Документы в коллекции 'labs':")
    for doc in all_docs:
        print(doc)
except Exception as e:
    print(f"Ошибка при чтении данных: {e}")
```

Примечания:
- Убедитесь, что ваш пользователь имеет права на запись и чтение в базе данных.
- Если у вас используются дополнительные настройки безопасности, например, SSL или другие параметры, это также следует учесть в URI подключения.
- Проверка подключения через команду `ping` является стандартной практикой для проверки доступности MongoDB.

Теперь у вас есть готовый код для подключения к MongoDB, проверки подключения, создания тестовых данных и их загрузки в коллекцию `labs`.
# Формирование строки подключения
connection_string = f"mongodb://{username}:{password}@{host}:{port}/{database_name}?authSource={auth_source}"

Подключение к MongoDB

```
try:
    client = MongoClient(connection_string)
    # Проверка подключения
    client.admin.command('ping')
    print("Соединение с MongoDB успешно установлено.")
except ConnectionError:
    print("Не удалось подключиться к MongoDB.")

# Закрытие соединения
client.close()
```

Объяснение кода:

- **Импорт библиотек:** Импортируются необходимые библиотеки для работы с MongoDB и обработки ошибок соединения.
- **Параметры подключения:** Определяются параметры подключения, включая имя пользователя, пароль, адрес сервера, порт и имя базы данных.
- **Формирование строки подключения:** Строка подключения формируется на основе указанных параметров.
- **Подключение и проверка:** Осуществляется подключение к MongoDB через объект `MongoClient`. Затем выполняется команда `ping`, которая отправляет запрос к серверу MongoDB для проверки его доступности.
- **Обработка ошибок:** Если соединение не удается установить, выводится сообщение об ошибке.
- **Закрытие соединения:** После проверки соединение закрывается.

Результат:
Если соединение установлено успешно, вы увидите сообщение `Соединение с MongoDB успешно установлено.`. В противном случае будет выведено сообщение об ошибке подключения.

Подключение к MongoDB с аутентификацией

1. **Импортируйте библиотеку `pymongo`:**

   ```python
   from pymongo import MongoClient
   ```

2. **Создайте объект клиента `MongoClient` с указанием аутентификационных данных:**

   Для подключения к MongoDB с использованием аутентификации необходимо указать строку подключения, содержащую имя

пользователя, пароль, адрес сервера, порт и имя базы данных. Формат строки подключения следующий:

```python
client = MongoClient('mongodb://username:password@host:port/database_name')
```

Если база данных защищена с использованием авторизации, необходимо подключиться с использованием правильных учетных данных. Например, если MongoDB работает на `localhost` (127.0.0.1) на порту `27017`, и у вас есть пользователь с именем `myUser` и паролем `myPassword` в базе данных `admin`, строка подключения будет выглядеть так:

```python
client = MongoClient('mongodb://myUser:myPassword@localhost:27017/admin')
```

Если вам нужно подключиться к конкретной базе данных, можно указать ее в конце строки подключения:

```python
client = MongoClient('mongodb://myUser:myPassword@localhost:27017/my_database')
```

Вы также можете указать дополнительные параметры подключения, такие как `authSource` (если пользователь хранится в другой базе данных):

```python
client = MongoClient('mongodb://myUser:myPassword@localhost:27017/my_database?authSource=admin')
```

3. **Выбор базы данных:**

После успешного подключения к серверу MongoDB выберите базу данных:

```python
db = client['my_database']
```

```

4. **Выбор коллекции:**

   Выберите коллекцию для работы:

   ```python
   collection = db['my_collection']
   ```

5. **Выполнение операций:**

   Теперь можно выполнять различные операции, например, вставку или выборку данных:

   ```python
   document = {"name": "Jane Doe", "age": 25, "city": "San Francisco"}
   collection.insert_one(document)

   documents = collection.find()
   for doc in documents:
       print(doc)
   ```

6. **Закрытие подключения:**

   После завершения работы с базой данных закройте соединение:

   ```python
   client.close()
   ```

Полный пример подключения и работы с MongoDB с аутентификацией

```python
from pymongo import MongoClient

# Подключение к MongoDB с аутентификацией
client = MongoClient('mongodb://myUser:myPassword@localhost:27017/my_database?authSource=admin')

# Выбор базы данных
db = client['my_database']
```
```

```
# Выбор коллекции
collection = db['my_collection']

# Вставка документа в коллекцию
document = {"name": "Jane Doe", "age": 25, "city": "San Francisco"}
collection.insert_one(document)

# Получение всех документов из коллекции
documents = collection.find()
for doc in documents:
    print(doc)

# Закрытие соединения
client.close()
```

Этот пример демонстрирует, как подключиться к MongoDB с использованием аутентификации, выбрать базу данных и коллекцию, выполнить операции с данными и закрыть соединение.

1.3. Загрузка данных из CSV в MongoDB:
```python
import csv
from pymongo import MongoClient

client = MongoClient('localhost', 27017)
db = client['lab_db']
collection = db['lab_collection']

with open('data.csv', mode='r') as file:
    reader = csv.DictReader(file)
    for row in reader:
        collection.insert_one(row)
```

Шаг 2: Установка и настройка Redis

2.1. Установите Redis:
```bash
sudo apt-get install redis-server
```

2.2. Запустите Redis:
```bash
sudo systemctl start redis
```

```

2.3. Загрузка данных из CSV в Redis:
```python
import csv
import redis

r = redis.Redis(host='localhost', port=6379, db=0)

with open('data.csv', mode='r') as file:
    reader = csv.reader(file)
    header = next(reader)  # Считываем заголовок
    for row in reader:
        r.hmset(f"row:{row[0]}", {header[i]: row[i] for i in range(1, len(row))})
```

Шаг 3: Установка и настройка Neo4j

3.1. Установите Neo4j:
```bash
wget -O - https://debian.neo4j.com/neotechnology.gpg.key | sudo apt-key add
-
echo 'deb https://debian.neo4j.com stable 4.x' | sudo tee
/etc/apt/sources.list.d/neo4j.list
sudo apt-get update
sudo apt-get install neo4j
```

3.2. Запустите Neo4j:
```bash
sudo systemctl start neo4j
```

3.3. Загрузка данных из CSV в Neo4j:
```python
from neo4j import GraphDatabase

driver = GraphDatabase.driver("bolt://localhost:7687", auth=("neo4j",
"password"))

def create_node(tx, label, properties):
    query = f"CREATE (n:{label} {{"
    query += ", ".join([f"{key}: ${key}" for key in properties.keys()])
    query += "})"
    tx.run(query, **properties)
```
```

```
with driver.session() as session:
    with open('data.csv', mode='r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            session.write_transaction(create_node, "LabNode", row)
```

Шаг 4: Выполнение операций с данными

4.1. **MongoDB:**
   - Найдите документ по определённому критерию:
   ```python
   result = collection.find_one({"field_name": "value"})
   print(result)
   ```

   - Обновите документ:
   ```python
   collection.update_one({"field_name": "value"}, {"$set": {"field_name": "new_value"}})
   ```

   - Удалите документ:
   ```python
   collection.delete_one({"field_name": "value"})
   ```

4.2. **Redis:**
   - Получите данные по ключу:
   ```python
   data = r.hgetall('row:1')
   print(data)
   ```

   - Обновите данные:
   ```python
   r.hset('row:1', 'field_name', 'new_value')
   ```

   - Удалите ключ:
   ```python
   r.delete('row:1')
   ```

4.3. **Neo4j:**
   - Найдите узел по определённому критерию:

```python
with driver.session() as session:
    result = session.run("MATCH (n:LabNode {field_name: $value}) RETURN n", value="value")
    for record in result:
        print(record['n'])
```

- Обновите узел:
```python
with driver.session() as session:
    session.run("MATCH (n:LabNode {field_name: $value}) SET n.field_name = $new_value", value="value", new_value="new_value")
```

- Удалите узел:
```python
with driver.session() as session:
    session.run("MATCH (n:LabNode {field_name: $value}) DELETE n", value="value")
```

Варианты для студентов:

Каждому студенту предоставляется CSV файл с уникальными данными, который он должен загрузить в MongoDB, Redis и Neo4j. Студент должен продемонстрировать выполнение базовых операций (вставка, выборка, обновление, удаление) для каждой из СУБД. Ниже приводятся примеры тем, с которыми могут быть связаны данные в CSV:

1. Сотрудники компании и их контактные данные.
2. Товары интернет-магазина и их описание.
3. Книги и авторы в библиотеке.
4. Курсы и преподаватели в университете.
5. Пациенты и их медицинские карты.
6. Заказы и клиенты интернет-магазина.
7. Фильмы и режиссеры в кинобазе.
8. Услуги и клиенты сервисного центра.
9. Транспортные средства и их владельцы.
10. Продукты и их поставщики.
11. Сотрудники и их департаменты в организации.
12. Студенты и их курсы в образовательном учреждении.
13. Проекты и участники в IT компании.
14. Клиенты и их заказы в ресторане.
15. Маршруты и станции в транспортной сети.
16. Оборудование и его спецификации в производстве.

17. Музыкальные альбомы и артисты.
18. Туристические маршруты и гиды.
19. Программные продукты и их версии.
20. Клиенты и их счета в банке.
21. Бренды и их товары на рынке.
22. Квартиры и арендаторы в жилом комплексе.
23. Ивенты и участники в системе мероприятий.
24. Лекции и студенты в образовательной платформе.
25. Контракты и их условия для компании.

Отчет по лабораторной работе:
Отчет должен включать:
1. Описание загрузки данных в каждую из NoSQL СУБД.
2. Пример кода для работы с MongoDB, Redis и Neo4j.
3. Результаты выполнения операций с данными.
4. Выводы по работе с различными типами NoSQL баз данных.

Заключение:
Эта лабораторная работа позволит студентам ознакомиться с различными методами хранения данных в NoSQL системах на примере MongoDB, Redis и Neo4j. Студенты получат навыки работы с данными, загружаемыми из CSV файлов, а также научатся выполнять базовые операции в каждой из СУБД.

User prefers lab work that involves databases and graph databases such as Neo4j, using Cypher queries.### 25 Вариантов заданий для студентов в контексте базы данных "Учебные курсы" на Neo4j

1. **Вариант 1:** Напишите Cypher-запрос, который вернет список всех студентов, записанных на курс "Discrete Mathematics".

2. **Вариант 2:** Составьте запрос, который вернет список курсов, на которые записаны студенты с именем "Nina" и "Olga".

3. **Вариант 3:** Найдите всех авторов курса "Data Processing" и верните их имена.

4. **Вариант 4:** Напишите запрос, который вернет список курсов, созданных сотрудником с именем "Andrey".

5. **Вариант 5:** Составьте запрос для получения списка всех курсов, где "Stas" является редактором.

6. **Вариант 6:** Напишите запрос, который вернет всех студентов, записанных на курс "Databases", и укажите, какие сотрудники связаны с этим курсом как авторы, дикторы и редакторы.

7. **Вариант 7:** Найдите всех сотрудников, которые имеют отношение к созданию курса "Discrete Mathematics", и определите их роль.

8. **Вариант 8:** Напишите запрос, чтобы найти всех студентов, которые учатся на всех трех курсах ("Discrete Mathematics", "Databases", "Data Processing").

9. **Вариант 9:** Составьте запрос, который вернет список студентов, обучающихся на курсе "Data Processing", и перечислите авторов этого курса.

10. **Вариант 10:** Напишите запрос, который вернет количество студентов, записанных на каждый из курсов.

11. **Вариант 11:** Найдите всех студентов, которые учатся у "Elena", и верните список курсов, на которые они записаны.

12. **Вариант 12:** Напишите запрос, чтобы получить список всех сотрудников, которые участвуют в создании хотя бы одного курса в роли автора, диктора или редактора.

13. **Вариант 13:** Составьте запрос, который вернет список студентов, записанных на курсы, где "Katarina" является дизайнером.

14. **Вариант 14:** Напишите запрос, чтобы найти все курсы, которые созданы командой сотрудников (автор, диктор, редактор), включающей "Dmitry".

15. **Вариант 15:** Найдите всех студентов, которые не записаны ни на один курс.

16. **Вариант 16:** Напишите запрос, который вернет список студентов и количество курсов, на которые они записаны.

17. **Вариант 17:** Составьте запрос, чтобы получить список всех курсов, где один и тот же человек выполняет несколько ролей (например, автор и редактор).

18. **Вариант 18:** Напишите запрос, который вернет список всех студентов, записанных на курс "Discrete Mathematics", и их наставников.

19. **Вариант 19:** Найдите всех сотрудников, которые являются одновременно авторами и дикторами какого-либо курса.

20. **Вариант 20:** Напишите запрос, чтобы найти все курсы, на которые записаны студенты "Elena" и "Stas".

21. **Вариант 21:** Составьте запрос, чтобы получить список студентов, которые записаны на курс "Databases" и имеют того же наставника, что и "Nina".

22. **Вариант 22:** Напишите запрос, который вернет список курсов, на которых "Roman" является студентом, а "Olga" - сотрудником (в любой роли).

23. **Вариант 23:** Найдите всех сотрудников, которые принимали участие в создании курсов, на которых учится студент "Konstantin".

24. **Вариант 24:** Напишите запрос, чтобы найти студентов, которые записаны на курсы, на которых "Victoria" является дизайнером.

25. **Вариант 25:** Составьте запрос, чтобы получить список студентов, которые учатся на более чем двух курсах, и перечислите эти курсы.

Эти задания помогут студентам освоить язык Cypher, научиться работать с графовыми базами данных и выполнять сложные запросы для анализа данных в контексте учебных курсов.