

## Лабораторная работа 2-1. 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. Установка необходимых библиотек

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

```
``python
!pip install pymongo
``
```

##### 1.2. Подключение к MongoDB с аутентификацией

Предположим, что у вас есть следующие параметры для подключения:

- URI MongoDB: mongodb://localhost:27017/
- Имя базы данных: labs
- Имя пользователя: mongouser
- Пароль: mongopasswd
- Имя коллекции (таблицы): lab21

Создадим подключение и проверим его успешность.

```
``python
from pymongo import MongoClient
#Задайте параметры подключения
mongo_uri = "mongodb://mongouser: mongopasswd @localhost:27017"
```

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[' labs']
# Выбор коллекции
```

```
labs_collection = db['lab21']
```

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

### 1.3. 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},  
]  
...
```

### 1.4. 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}")  
...
```

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

### 1.6. Закрытие подключения.

После завершения работы с базой данных закройте соединение:

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

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

## 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. Установка необходимых библиотек

Вначале убедитесь, что у вас установлена библиотека **redis**. Если нет, выполните следующую команду:

```
``python
!pip install redis
``
```

2.4. После установки библиотеки можно подключиться к Redis и выполнить необходимые операции.

```
``python
import redis
import csv
import json
# Подключение к Redis с аутентификацией
r = redis.Redis(
    host='localhost',
    port=6379,
    db=0 # Подключение к базе данных 0
)
# Проверка соединения
try:
    r.ping()
    print("Соединение с Redis успешно установлено.")
except redis.ConnectionError:
    print("Не удалось подключиться к Redis.")
``
```

2.5. Генерация и загрузка данных в Redis. Сгенерируем десять записей и загрузим их в базу данных Redis.

```
``python
from datetime import datetime
# Создание 10 записей
for i in range(1, 11):
    key = f"key_{i}"
    value = f"value_{i}"
    r.set(key, value)
    print(f"Создана запись: {key} = {value}")

# Проверка созданных записей
for i in range(1, 11):
    key = f"key_{i}"
    value = r.get(key)
    print(f"Проверка: {key} = {value.decode('utf-8')}")
```

```
# Функция для печати разделителя
def print_separator(message):
    print(f"\n{'='*20} {message} {'='*20}")
```

2.6. Создание по 5 записей различных типов данных.

```
``python
print_separator("Создание данных")
```

```

# Строки
for i in range(1, 6):
    r.set(f"string_{i}", f"value_{i}")
    print(f"Создана строка: string_{i} = value_{i}")

# Списки
for i in range(1, 6):
    r.rpush(f"list_{i}", *[f"item_{j}" for j in range(1, 4)])
    print(f"Создан список: list_{i} = {r.lrange(f'list_{i}', 0, -1)}")

# Множества
for i in range(1, 6):
    r.sadd(f"set_{i}", *[f"element_{j}" for j in range(1, 4)])
    print(f"Создано множество: set_{i} = {r.smembers(f'set_{i}')}")

# Хэши
for i in range(1, 6):
    r.hset(f"hash_{i}", mapping={f"field_{j}": f"value_{j}" for j in range(1, 4)})
    print(f"Создан хэш: hash_{i} = {r.hgetall(f'hash_{i}')}")

# Упорядоченные множества
for i in range(1, 6):
    r.zadd(f"zset_{i}", {f"member_{j}": j for j in range(1, 4)})
    print(f"Создано упорядоченное множество: zset_{i} = {r.zrange(f'zset_{i}', 0, -1, withscores=True)}")

```

**2.7. Получение данных по ключу.**

```

```python
print_separator("Получение данных")

print(f"Строка: {r.get('string_1')}")
print(f"Список: {r.lrange('list_1', 0, -1)}")
print(f"Множество: {r.smembers('set_1')}")
print(f"Хэш: {r.hgetall('hash_1')}")
print(f"Упорядоченное множество: {r.zrange('zset_1', 0, -1, withscores=True)}")

```

**2.8. Обновление данных по ключу.**

```

```python
print_separator("Обновление данных")
r.set("string_1", "new_value")
print(f"Обновлена строка: string_1 = {r.get('string_1')}")
r.lset("list_1", 0, "new_item")
print(f"Обновлен список: list_1 = {r.lrange('list_1', 0, -1)}")
r.sadd("set_1", "new_element")
print(f"Обновлено множество: set_1 = {r.smembers('set_1')}")
r.hset("hash_1", "new_field", "new_value")
print(f"Обновлен хэш: hash_1 = {r.hgetall('hash_1')}")
r.zadd("zset_1", {"new_member": 5})
print(f"Обновлено упорядоченное множество: zset_1 = {r.zrange('zset_1', 0, -1, withscores=True)}")

```

## 2.9. Удаление данных по ключу.

```
```python
```

```
print_separator("Удаление данных")
r.delete("string_5", "list_5", "set_5", "hash_5", "zset_5")
print("Удалены ключи: string_5, list_5, set_5, hash_5, zset_5")
```

## 2.10. Проверка удаленных данных

```
```python
```

```
print_separator("Проверка удаленных данных")
for key in ["string_5", "list_5", "set_5", "hash_5", "zset_5"]:
    print(f"Существует ли ключ {key}? {r.exists(key)}")
```

## 2.10. Выгрузить все данные из Redis в csv.

```
```python
```

```
def flatten_data(data):
    if isinstance(data, (str, int, float)):
        return str(data)
    elif isinstance(data, list):
        return json.dumps(data, ensure_ascii=False)
    elif isinstance(data, dict):
        return json.dumps(data, ensure_ascii=False)
    else:
        return str(data)

def dump_redis_to_csv(filename='redis_dump.csv'):
    # Подключение к Redis
    r = redis.Redis(host='localhost', port=6379, db=0)

    # Получение всех ключей
    keys = r.keys('*')

    with open(filename, 'w', newline="", encoding='utf-8') as csvfile:
        csvwriter = csv.writer(csvfile)
        csvwriter.writerow(['Key', 'Type', 'Value']) # Заголовки

    for key in keys:
        # Декодирование ключа из байтов в строку
        key_str = key.decode('utf-8')

        # Определение типа данных ключа
        key_type = r.type(key).decode('utf-8')

        if key_type == 'string':
            value = r.get(key).decode('utf-8')
        elif key_type == 'list':
            value = r.lrange(key, 0, -1)
            value = [item.decode('utf-8') for item in value]
        elif key_type == 'set':
            value = list(r.smembers(key))
            value = [item.decode('utf-8') for item in value]
        elif key_type == 'hash':
```

```

        value = r.hgetall(key)
        value = {k.decode('utf-8'): v.decode('utf-8') for k, v in value.items()}
    elif key_type == 'zset':
        value = r.zrange(key, 0, -1, withscores=True)
        value = [(item[0].decode('utf-8'), item[1]) for item in value]
    else:
        value = f"Неподдерживаемый тип данных: {key_type}"

    # Записываем данные в CSV
    csvwriter.writerow([key_str, key_type, flatten_data(value)])

# Заккрытие соединения
r.close()

print(f"Данные сохранены в файл '{filename}'")

# Выполнение выгрузки
dump_redis_to_csv()

...

```

CSV формат ограничен в представлении сложных структур данных. Для списков, множеств, хешей и упорядоченных множеств мы преобразуем их в JSON-строки. Это позволяет сохранить всю информацию, но может затруднить дальнейшую обработку данных в CSV.

Каждая строка в CSV будет содержать три столбца:

- Key: ключ из Redis.
- Type: тип данных (string, list, set, hash, zset).
- Value: значение, преобразованное в строку или JSON.

### **Варианты заданий**

Каждому студенту предоставляется CSV файл с уникальными данными, который он должен загрузить в MongoDB, Redis. Студент должен продемонстрировать выполнение базовых операций (вставка, выборка, обновление, удаление) для каждой из СУБД. Ниже приводятся примеры тем, с которыми должны быть связаны данные в CSV (не менее 100 записей).

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.

Следующие задания могут быть выполнены с помощью “песочницы”:  
<http://console.neo4j.org> в контексте использования базы «Учебные курсы», которая на языке Cypher описана следующим образом:

```
CREATE (C:course{name:'Discrete Mathematics'}),
(S:course{name:'Databases'}),
(I:course{name:'Data Processing'}),
(E:person{name:'Elena'}),
(N:person{name:'Natalia'}),
(V:person{name:'Victoria'}),
(O:person{name:'Olga'}),
(St:person{name:'Stas'}),
(A:person{name:'Andrey'}),
(D:person{name:'Dan'}),
(Al:person{name:'Alex'}),
(Dm:person{name:'Dmitry'}),
(K:person{name:'Katarina'}),
(Elena:student{name:'Teresa'}),
(Nina:student{name:'Nina'}),
(Victor:student{name:'Victor'}),
(Olga:student{name:'Olga'}),
(Stas:student{name:'Stas'}),
(Anna:student{name:'Elen'}),
(Denis:student{name:'Denis'}),
(Alexandr:student{name:'Alexandr'}),
(Dmitry:student{name:'Dmitry'}),
(Konstantin:student{name:'Konstantin'}),
(Roman:student{name:'Roman'}),
(Ilia:student{name:'Ilia'}),
(E)-[:author]->(C),
(N)-[:author]->(C),
(A)-[:author]->(S),
(A)-[:author]->(I),
(Al)-[:author]->(I),
(Dm)-[:author]->(I),
(O)-[:speaker]->(C),
(St)-[:editor]->(C),
(V)-[:designer]->(C),
(D)-[:speaker]->(S),
(St)-[:editor]->(S),
(V)-[:designer]->(S),
(D)-[:speaker]->(I),
(St)-[:editor]->(I),
(K)-[:designer]->(I),
(Elena)-[:learn]->(I),
(Elena)-[:learn]->(C),
```

```
(Nina)-[:learn]->(S),
(Nina)-[:learn]->(C),
(Victor)-[:learn]->(S),
(Victor)-[:learn]->(C),
(Olga)-[:learn]->(S),
(Olga)-[:learn]->(C),
(Stas)-[:learn]->(I),
(Stas)-[:learn]->(C),
(Stas)-[:learn]->(S),
(Anna)-[:learn]->(I),
(Anna)-[:learn]->(C),
(Anna)-[:learn]->(S),
(Denis)-[:learn]->(I),
(Alexandr)-[:learn]->(C),
(Alexandr)-[:learn]->(S),
(Dmitry)-[:learn]->(I),
(Dmitry)-[:learn]->(C),
(Konstantin)-[:learn]->(S),
(Roman)-[:learn]->(C),
(Ilia)-[:learn]->(C)
```

Узлы базы: **person** — сотрудники, **student** — студенты. Отношение **learn** связывает студентов с курсами, на которые они записались, а отношения **author**, **speaker** и **editor** связывают авторов, дикторов и монтажеров с курсами, которые они создавали. Именам сотрудников, студентов и курсов соответствуют атрибуты **name**.

### Вариантов заданий в контексте базы данных «Учебные курсы» на Neo4j

1. Напишите Cypher-запрос, который вернет список всех студентов, записанных на курс "Discrete Mathematics".
2. Составьте запрос, который вернет список курсов, на которые записаны студенты с именем "Nina" и "Olga".
3. Найдите всех авторов курса "Data Processing" и верните их имена.
4. Напишите запрос, который вернет список курсов, созданных сотрудником с именем "Andrey".
5. Составьте запрос для получения списка всех курсов, где "Stas" является редактором.
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. Найдите всех сотрудников, которые имеют отношение к созданию курса "Discrete Mathematics", и определите их роль.
8. Напишите запрос, чтобы найти всех студентов, которые учатся на всех трех курсах ("Discrete Mathematics", "Databases", "Data Processing").
9. Составьте запрос, который вернет список студентов, обучающихся на курсе "Data Processing", и перечислите авторов этого курса.
10. Напишите запрос, который вернет количество студентов, записанных на каждый из курсов.
11. Найдите всех студентов, которые учатся у "Elena", и верните список курсов, на которые они записаны.
12. Напишите запрос, чтобы получить список всех сотрудников, которые участвуют в создании хотя бы одного курса в роли автора, диктора или редактора.
13. Составьте запрос, который вернет список студентов, записанных на курсы, где "Katarina" является дизайнером.
14. Напишите запрос, чтобы найти все курсы, которые созданы командой сотрудников (автор, диктор, редактор), включающей "Dmitry".



15. Найдите всех студентов, которые не записаны ни на один курс.
16. Напишите запрос, который вернет список студентов и количество курсов, на которые они записаны.
17. Составьте запрос, чтобы получить список всех курсов, где один и тот же человек выполняет несколько ролей (например, автор и редактор).
18. Напишите запрос, который вернет список всех студентов, записанных на курс "Discrete Mathematics", и их наставников.
19. Найдите всех сотрудников, которые являются одновременно авторами и дикторами какого-либо курса.
20. Напишите запрос, чтобы найти все курсы, на которые записаны студенты "Elena" и "Stas".
21. Составьте запрос, чтобы получить список студентов, которые записаны на курс "Databases" и имеют того же наставника, что и "Nina".
22. Напишите запрос, который вернет список курсов, на которых "Roman" является студентом, а "Olga" - сотрудником (в любой роли).
23. Найдите всех сотрудников, которые принимали участие в создании курсов, на которых учится студент "Konstantin".
24. Напишите запрос, чтобы найти студентов, которые записаны на курсы, на которых "Victoria" является дизайнером.
25. Составьте запрос, чтобы получить список студентов, которые учатся на более чем двух курсах, и перечислите эти курсы.

### **Отчет по работе**

Отчет должен включать:

- Описание загрузки данных в каждую из NoSQL СУБД.
- Пример кода для работы с MongoDB, Redis и Neo4j.
- Результаты выполнения операций с данными.
- Выводы по работе с различными типами NoSQL баз данных.