Департамент образования города Москвы

Государственное автономное образовательное учреждение

высшего образования города Москвы

«Московский городской педагогический университет»

Институт цифрового образования

Департамент информатики, управления и технологий

Тяпкина Полина Андреевна

# Лабораторная работа по «Инструменты для хранения и обработки больших данных»

Выполнил:

Тяпкина Полина Андреевна

Курс обучения: 4

Форма обучения: очная

Проверил:

Босенко Тимур Муртазович

Москва

2025

# **Лабораторная работа 2.1. Изучение методов хранения данных на основе NoSQL**

**Цель работы:** изучение и практическое применение трех различных типов NoSQL баз данных, а именно: документо-ориентированной MongoDB, графовой Neo4j и ключ-значение Redis.

**Оборудование и программное обеспечение:**

* Операционная система Ubuntu
* Язык программирования Python (с библиотеками pymongo, redis, neo4j). CSV файлы с данными.

# **Краткая теоретическая справка**

**NoSQL** — это подход к проектированию баз данных, которые не являются реляционными и могут использовать различные модели данных. Они оптимизированы для конкретных сценариев использования и обеспечивают высокую масштабируемость и гибкость.

**MongoDB** ­­­­­­— документо-ориентированная СУБД. MongoDB хранит данные в гибких, JSON-подобных документах. Это означает, что поля могут варьироваться от документа к документу и структура данных может быть изменена со временем.

**Основные концепции**:

* База данных (Database): контейнер для коллекций.
* Коллекция (Collection): аналог таблицы в реляционных СУБД. Хранит группу связанных документов, но не требует от них
* Документ (Document): аналог строки (записи) в реляционных СУБД. Представляет собой структуру данных в формате BSON (бинарный JSON).
* Поле (Field): аналог столбца, пара ключ-значение в документе.
* Ключевая особенность: гибкая схема, позволяющая хранить в одной коллекции документы с разным набором полей.

**Neo4j** — это графовая база данных, разработанная для хранения и обработки связанных данных. Она идеально подходит для анализа сложных взаимосвязей, 2 таких как социальные сети, рекомендательные системы и управление зависимостями.

Основные компоненты графовой модели:

* Узлы (Nodes): представляют сущности (например, "Человек", "Фильм"). Аналогичны записям в реляционной таблице.
* Отношения (Relationships): представляют связи между узлами (например, ACTED\_IN, DIRECTED). Отношения всегда имеют направление и тип.
* Свойства (Properties): пары ключ-значение, которые хранятся внутри узлов и отношений (например, name: 'Tom Hanks').
* Язык запросов: Cypher — декларативный язык для запросов к графу.

**Redis** (Remote Dictionary Server) — это высокопроизводительное хранилище данных в памяти, работающее по принципу "ключ-значение". Оно часто используется для кэширования, управления сессиями, очередей сообщений и в качестве брокера сообщений.

Основные структуры данных:

* Строки (Strings): простейший тип, где одному ключу соответствует одно строковое значение.
* Списки (Lists): последовательности строк, упорядоченные по порядку вставки.
* Множества (Sets): неупорядоченные коллекции уникальных строк
* Хэши (Hashes): структуры для хранения объектов, состоящие из полей и их значений.
* Упорядоченные множества (Sorted Sets): множества, где каждый элемент связан с числовым значением (оценкой), которое используется для сортировки.

# **Процесс выполнения общего из GitHub**

**MongoDB:**

1. Вход в Mongo Compass
2. Создаем базу данных filmdb

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

1. Загружаем данные с помощью ⊕ADD DATA → Insert document

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.  
Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Redis:**

1. Открыл Reddis Commander

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

1. Выполнил все шаги из гит хаба:

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как программное обеспечение, текст, Мультимедийное программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Neo4j:**

1. Подключаемся в Neo4j Browser

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

1. Выполняем команду :play movie graph

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

1. Создаём граф

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

1. Выполняем несколько запросов:
   1. MATCH (tom {name: "Tom Hanks"}) RETURN tom

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

* 1. MATCH (cloudAtlas {title: "Cloud Atlas"}) RETURN cloudAtlas

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.



Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как программное обеспечение, текст, Значок на компьютере, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

# **Индивидуальные задания**

## **MongoDB**

Цель: найти все фильмы, название которых начинается с "The". Для них добавить поле last\_updated с текущей датой ($currentDate).

1. Переключаю в консоли бд с test на filmdb и вписываю код:  
   Изображение выглядит как текст, снимок экрана, Шрифт, дисплей

   Содержимое, созданное искусственным интеллектом, может быть неверным.

```  
*db.movies.updateMany(*

*{ "title": { $regex: /^The/ } }, // Фильтр: название начинается с "The"*

*{*

*$currentDate: {*

*"last\_updated": { $type: "date" } // Добавляет поле с текущей датой*

*}*

*}*

*);  
///updateMany — обновляет все документы, соответствующие условию.*

*///{ $regex: /^The/ } — регулярное выражение для поиска названий, начинающихся с "The" (регистрозависимо). Если нужен регистронезависимый поиск, используйте /^The/i.*

*///$currentDate — автоматически устанавливает текущую дату в формате ISODate.*

```

1. Проверяю результат:

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

После добавления новых фильмов проверяем ещё раз код:

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Содержимое, созданное искусственным интеллектом, может быть неверным. Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Содержимое, созданное искусственным интеллектом, может быть неверным.

Видим, что после добавления фильмов в коллекцию количество названий, начинающихся на “The” прибавилось.

## **Redis**

Цель: Смоделировать подписки на теги: для пользователя user:99 в множество user:99:tags добавить 3 тега. Удалить один из тегов (SREM).

1. Добавление 3 тегов для пользователя user:99

```

*SADD user:99:tags "programming" "technology" "databases"*

```

1. Проверка добавленных тегов

```

*SMEMBERS user:99:tags*

```

1. Удаление одного тега (например, "technology")

```

*SREM user:99:tags "technology"*

```

1. Проверка результата после удаления

```

*SMEMBERS user:99:tags*

```

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

*SADD, SREM, SISMEMBER, SMEMBERS и SUNION* – Одни из важных команд для работы с множествами.  
Команда *SADD* добавляет указанные элементы в множество. Если множество не существует, оно создается автоматически.

Команда *SREM* удаляет указанный элемент из множества и возвращает количество удаленных элементов.

## **Neo4j**

Цель: Найти кратчайший путь между "CarrieAnne Moss" и "Tom Hanks".

1.Открываю Neo4j Browser

2.Вписываю код:  
  
```

*MATCH p=shortestPath(*

*(carrie:Person {name:"Carrie-Anne Moss"})-[\*]-(tom:Person {name:"Tom Hanks"})*

*)*

*RETURN p*

```

3.Результат:  
Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.  
Похоже на правило рукопожатия (через сколько рукопожатий человек «знаком» с тем или иным человеком)

# **Вывод**

В ходе выполнения практической работы я изучила и применила на практике три основных типа NoSQL баз данных, что позволило мне понять их особенности и области применения.

С документо-ориентированной СУБД MongoDB я работала с JSON-подобными документами, выполняла сложные запросы с использованием регулярных выражений и операции обновления данных. Эта база данных показала себя как гибкое решение для работы с иерархическими данными и сложными структурами.

С графовой СУБД Neo4j я освоила язык запросов Cypher, научилась находить кратчайшие пути между узлами и анализировать связи между сущностями. Этот тип базы данных оказался чрезвычайно эффективным для работы со связанными данными и анализа отношений.

С ключ-значение СУБД Redis я работала с множествами, выполняла операции добавления и удаления элементов. Redis продемонстрировал высокую производительность и простоту использования для задач, требующих быстрого доступа к данным.

Проведенная работа показала, что выбор конкретного типа NoSQL базы данных напрямую зависит от решаемой задачи. MongoDB оптимальна для сложных структур данных, Neo4j - для анализа связей, а Redis - для высокопроизводительных операций с простыми структурами данных. Полученный опыт позволяет мне обоснованно выбирать подходящую базу данных для различных проектных задач и эффективно работать с каждой из рассмотренных СУБД.