

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.

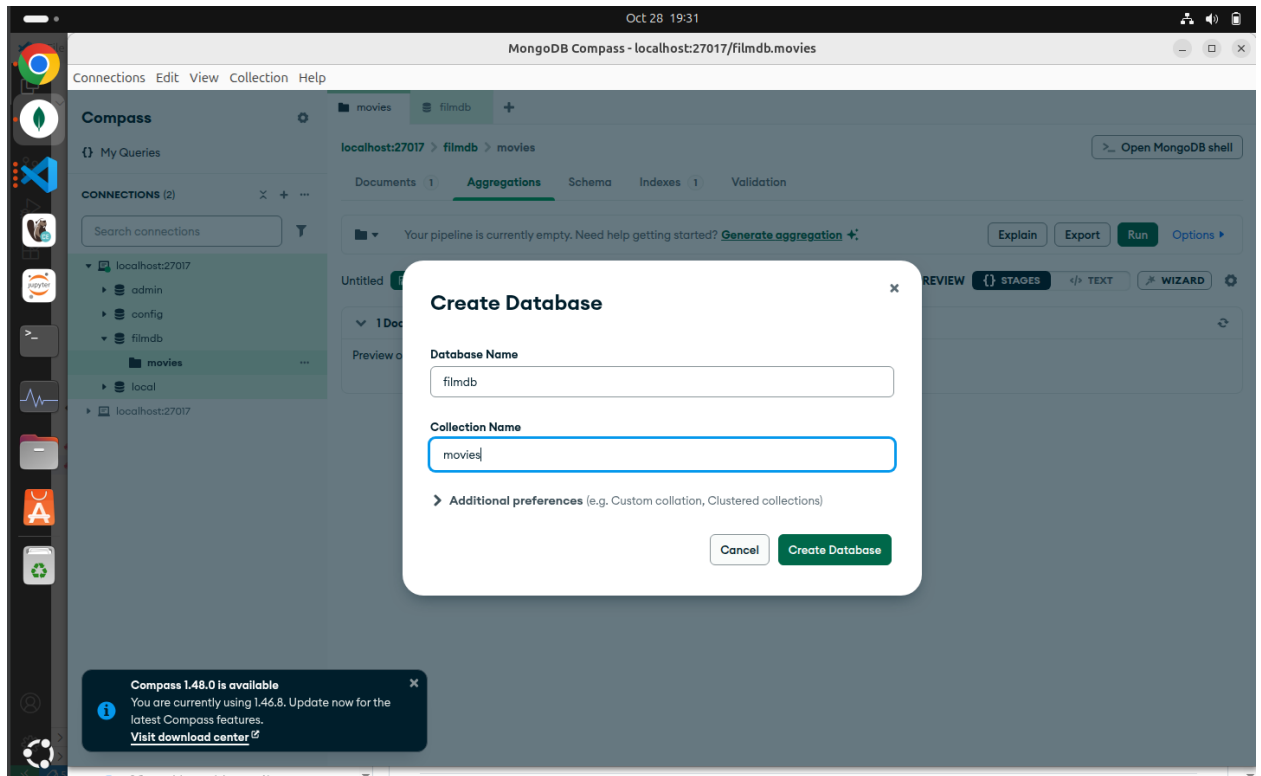
Основные структуры данных:

- 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:

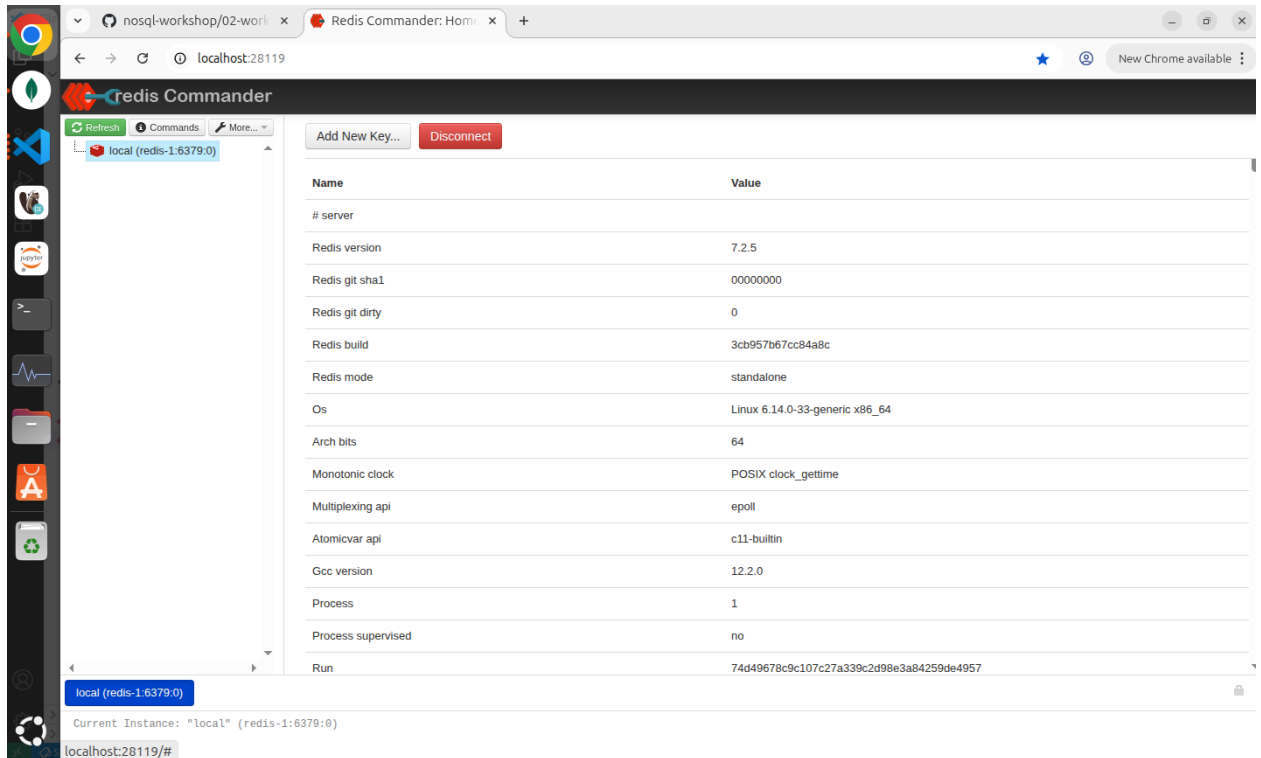
1. Вход в Mongo Compass
2. Создаем базу данных filmdb



3. Загружаем данные с помощью \oplus ADD DATA \rightarrow Insert document

Redis:

1. Открыл Reddis Commander



2. Выполнил все шаги из гит хаба:

```
mgpu@mgpu-vm:~$ docker run -it --rm --network nosql-platform bitnami/redis redis-cli -h redis-1 -p 6379
Unable to find image 'bitnami/redis:latest' locally
latest: Pulling from bitnami/redis
bc946fda1c85: Pull complete
Digest: sha256:f8fc4ba70daf1885971f82c4f655b9f4afbd701851efbea11c77faf299465210
Status: Downloaded newer image for bitnami/redis:latest
redis 17:42:20.12 INFO ==>
redis 17:42:20.13 INFO ==> Welcome to the Bitnami redis container
redis 17:42:20.15 INFO ==> Subscribe to project updates by watching https://github.com/bitnami/containers
redis 17:42:20.16 INFO ==> NOTICE: Starting August 28th, 2025, only a limited subset of images/charts will remain available
for free. Backup will be available for some time at the 'Bitnami Legacy' repository. More info at https://github.com/bitnami/
containers/issues/83267
redis 17:42:20.17 INFO ==>

redis-1:6379> PING
PONG
redis-1:6379> help @string

APPEND key value
summary: Appends a string to the value of a key. Creates the key if it doesn't exist.
since: 2.0.0

DECR key
summary: Decrements the integer value of a key by one. Uses 0 as initial value if the key doesn't exist.
since: 1.0.0

DECRBY key decrement
summary: Decrements a number from the integer value of a key. Uses 0 as initial value if the key doesn't exist.
since: 1.0.0

GET key
summary: Returns the string value of a key.
since: 1.0.0
```

Browser tabs: nosql-workshop/02-workshop, Redis Commander: Home, neo4j@bolt://localhost:7...

Address bar: localhost:28119

redis Commander

local (redis-1.6379:0)

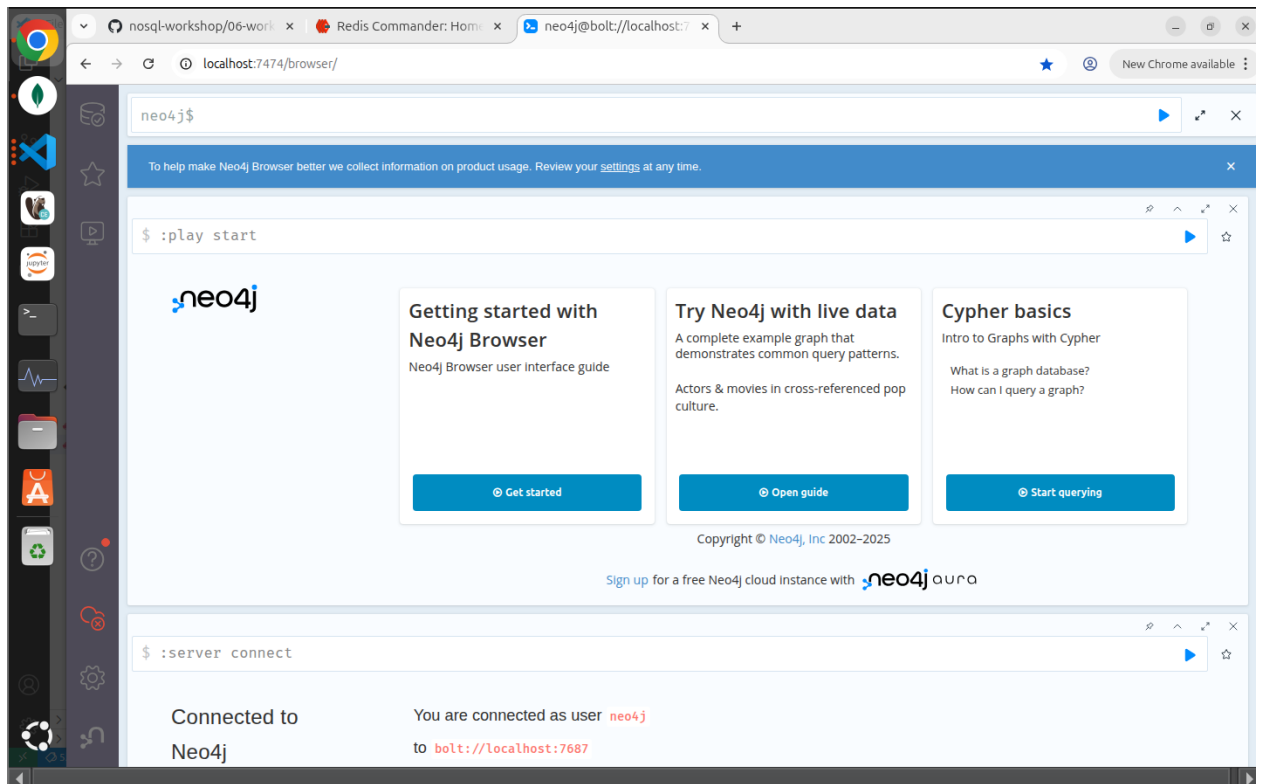
```
PING
"PONG"
SET server:name "redis-server"
"OK"
GET server:name
"redis-server"
EXISTS server:name
1
KEYS server*
1) "server:name"
KEYS *
1) "connections"
2) "server:name"
SET connections 10
"OK"
GET connections
"10"
SET connections 20
"OK"
GET connections
"20"
SETNX connections 30
0
GET connections
"20"
SETNX newkey 30
1
MSET key1 10 key2 20 key3 30
"OK"
MSET key1 key3
1) "10"
2) "30"
SET connections 10
```

Current Instance: "local" (redis-1:6379:0)

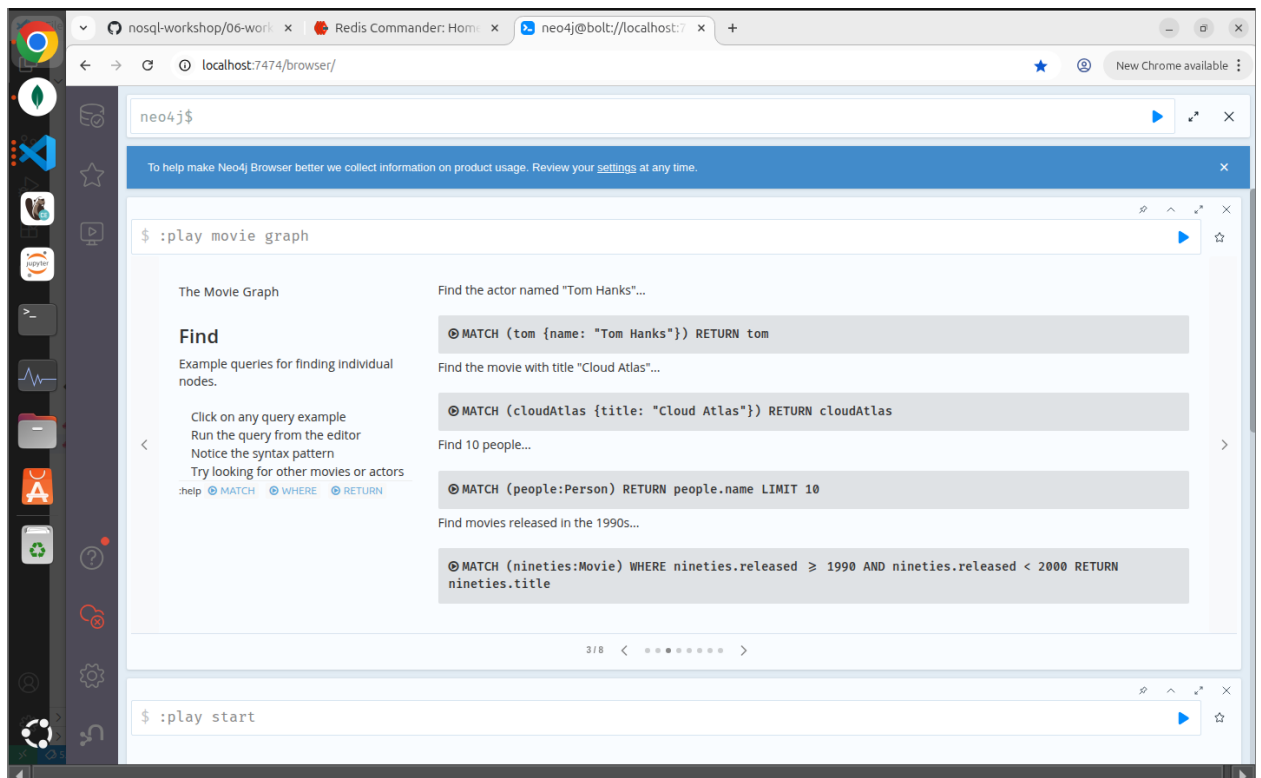
redis> |

Neo4j:

1. Подключаемся в Neo4j Browser



2. Выполняем команду `:play movie graph`



3. Создаём граф

The screenshot shows the Neo4j Browser interface. The command bar at the top contains the query: `neo4j$ CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome to the Real World'}) CREAT...`. The main area displays a graph visualization with Tom Hanks as the central node, connected to various other nodes representing movies and other actors. The right sidebar shows the 'Node properties' for a selected node, displaying details like element ID, ID, born year, and name.

4. Выполняем несколько запросов:

a. `MATCH (tom {name: "Tom Hanks"}) RETURN tom`

The screenshot shows the Neo4j Browser interface. The command bar at the top contains the query: `neo4j$ MATCH (tom {name: "Tom Hanks"}) RETURN tom`. The main area displays a single node labeled 'Tom Hanks'. The right sidebar shows the 'Overview' section, indicating that 1 node is displayed with 0 relationships.

b. `MATCH (cloudAtlas {title: "Cloud Atlas"}) RETURN cloudAtlas`

neo4j\$

To help make Neo4j Browser better we collect information on product usage. Review your [settings](#) at any time.

Node labels

- * (1) **Movie** (1) **Person** (10)

Relationship types

- * (10) **ACTED_IN** (4)
- DIRECTED** (3) **WROTE** (1)
- PRODUCED** (1) **REVIEWED** (1)

Displaying 1 nodes, 0 relationships.

Use Ctrl or Shift + scroll to zoom
Don't show again

neo4j\$ MATCH (tom {name: "Tom Hanks"}) RETURN tom

Overview

Node labels

- * (1) **Person** (1)

Displaying 1 nodes, 0 relationships.

c.

neo4j\$

To help make Neo4j Browser better we collect information on product usage. Review your [settings](#) at any time.

neo4j\$ MATCH (people:Person) RETURN people.name LIMIT 10

	people.name
1	"Keanu Reeves"
2	"Carrie-Anne Moss"
3	"Laurence Fishburne"
4	"Hugo Weaving"
5	"Lilly Wachowski"
6	"Lana Wachowski"
7	

Started streaming 10 records after 248 ms and completed after 249 ms.

neo4j\$ MATCH (cloudAtlas {title: "Cloud Atlas"}) RETURN cloudAtlas

Exports

d.

nosql-workshop/06-workshop

Redis Commander: Home

neo4j@bolt://localhost:7

localhost:7474/browser/

New Chrome available

neo4j\$

To help make Neo4j Browser better we collect information on product usage. Review your [settings](#) at any time.

neo4j\$ MATCH (nineties:Movie) WHERE nineties.released ≥ 1990 AND nineties.released < 2000 RETURN nineties.tit...

Table

	nineties.title
1	"The Matrix"
2	"The Devil's Advocate"
3	"A Few Good Men"
4	"As Good as It Gets"
5	"What Dreams May Come"
6	"Snow Falling on Cedars"
7	

Started streaming 20 records after 203 ms and completed after 208 ms.

neo4j\$ MATCH (people:Person) RETURN people.name LIMIT 10

Ctrl+click to copy to main editor

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:

```
>_MONGOSH
> use filmdb
< switched to db filmdb
filmdb> db.movies.updateMany(
  { "title": { $regex: /^The/ } },
  {
    $currentDate: {
      "last_updated": { $type: "date" }
    }
  }
);
```

...

```
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.

'''

2. Проверяю результат:

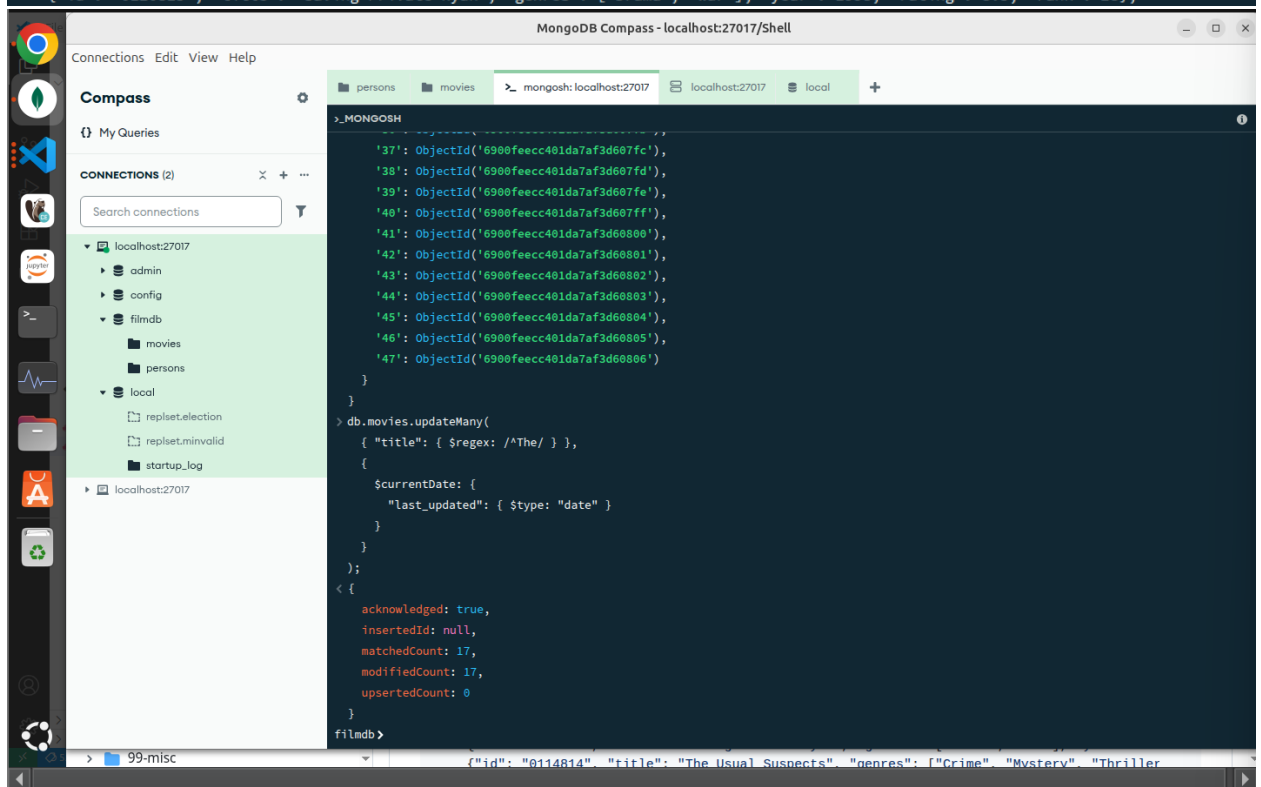
```
> use filmdb
< switched to db filmdb
> db.movies.updateMany(
  { "title": { $regex: /^The/ } },
  {
    $currentDate: {
      "last_updated": { $type: "date" }
    }
  }
);
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```

После добавления новых фильмов проверяем ещё раз код:

```

> db.movies.insertMany([
  {"id": "0111161", "title": "The Shawshank Redemption", "genres": ["Drama"], "year": 1994, "rating": 9.2, "rank": 1},
  {"id": "0068646", "title": "The Godfather", "genres": ["Crime", "Drama"], "year": 1972, "rating": 9.2, "rank": 2},
  {"id": "0071562", "title": "The Godfather: Part II", "genres": ["Crime", "Drama"], "year": 1974, "rating": 9.0, "rank": 3},
  {"id": "0468569", "title": "The Dark Knight", "genres": ["Action", "Crime", "Drama", "Thriller"], "year": 2008, "rating": 9.0, "rank": 4},
  {"id": "0050083", "title": "12 Angry Men", "genres": ["Drama"], "year": 1957, "rating": 8.9, "rank": 5},
  {"id": "0108052", "title": "Schindler's List", "genres": ["Biography", "Drama", "History"], "year": 1993, "rating": 8.9, "rank": 6},
  {"id": "0167260", "title": "The Lord of the Rings: The Return of the King", "genres": ["Adventure", "Drama", "Fantasy"], "year": 2003, "rating": 8.9, "rank": 7},
  {"id": "0060196", "title": "The Good, the Bad and the Ugly", "genres": ["Western"], "year": 1966, "rating": 8.8, "rank": 9},
  {"id": "0137523", "title": "Fight Club", "genres": ["Drama"], "year": 1999, "rating": 8.8, "rank": 10},
  {"id": "4154796", "title": "Avengers: Endgame", "genres": ["Action", "Adventure", "Fantasy", "Sci-Fi"], "year": 2019, "rating": 8.8, "rank": 11},
  {"id": "0120737", "title": "The Lord of the Rings: The Fellowship of the Ring", "genres": ["Adventure", "Drama", "Fantasy"], "year": 2001, "rating": 8.8, "rank": 12},
  {"id": "0109830", "title": "Forrest Gump", "genres": ["Drama", "Romance"], "year": 1994, "rating": 8.7, "rank": 13},
  {"id": "0080684", "title": "Star Wars: Episode V - The Empire Strikes Back", "genres": ["Action", "Adventure", "Fantasy", "Sci-Fi"], "year": 1980, "rating": 8.7, "rank": 14},
  {"id": "1375666", "title": "Inception", "genres": ["Action", "Adventure", "Sci-Fi", "Thriller"], "year": 2010, "rating": 8.7, "rank": 15},
  {"id": "0167261", "title": "The Lord of the Rings: The Two Towers", "genres": ["Adventure", "Drama", "Fantasy"], "year": 2002, "rating": 8.7, "rank": 16},
  {"id": "0073486", "title": "One Flew Over the Cuckoo's Nest", "genres": ["Drama"], "year": 1975, "rating": 8.7, "rank": 17},
  {"id": "0099685", "title": "Goodfellas", "genres": ["Biography", "Crime", "Drama"], "year": 1990, "rating": 8.7, "rank": 18},
  {"id": "0047478", "title": "Seven Samurai", "genres": ["Adventure", "Drama"], "year": 1954, "rating": 8.6, "rank": 20},
  {"id": "0114369", "title": "Se7en", "genres": ["Crime", "Drama", "Mystery", "Thriller"], "year": 1995, "rating": 8.6, "rank": 21},
  {"id": "0317248", "title": "City of God", "genres": ["Crime", "Drama"], "year": 2002, "rating": 8.6, "rank": 22},
  {"id": "0076759", "title": "Star Wars: Episode IV - A New Hope", "genres": ["Action", "Adventure", "Fantasy", "Sci-Fi"], "year": 1977, "rating": 8.6, "rank": 23},
  {"id": "0102926", "title": "The Silence of the Lambs", "genres": ["Crime", "Drama", "Thriller"], "year": 1991, "rating": 8.6, "rank": 24},
  {"id": "0038650", "title": "It's a Wonderful Life", "genres": ["Drama", "Family", "Fantasy"], "year": 1946, "rating": 8.6, "rank": 25},
  {"id": "0118799", "title": "Life Is Beautiful", "genres": ["Comedy", "Drama", "Romance", "War"], "year": 1997, "rating": 8.6, "rank": 26},
  {"id": "0245429", "title": "Spirited Away", "genres": ["Animation", "Adventure", "Family", "Fantasy", "Mystery"], "year": 2001, "rating": 8.6, "rank": 27},
  {"id": "0120815", "title": "Saving Private Ryan", "genres": ["Drama", "War"], "year": 1998, "rating": 8.5, "rank": 28},

```



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"
```

'''

2. Проверка добавленных тегов

'''

```
SMEMBERS user:99:tags
```

'''

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

'''

```
SREM user:99:tags "technology"
```

'''

4. Проверка результата после удаления

'''

```
SMEMBERS user:99:tags
```

'''

```
SADD user:99:tags "programming" "technolodgy" "databases"
```

```
3
```

```
SMEMBERS user:99:tags
```

```
1) "programming"
```

```
2) "technolodgy"
```

```
3) "databases"
```

```
SREM user:99:tags "technolodgy"
```

```
1
```

```
SMEMBERS user:99:tags
```

```
1) "programming"
```

```
2) "databases"
```

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

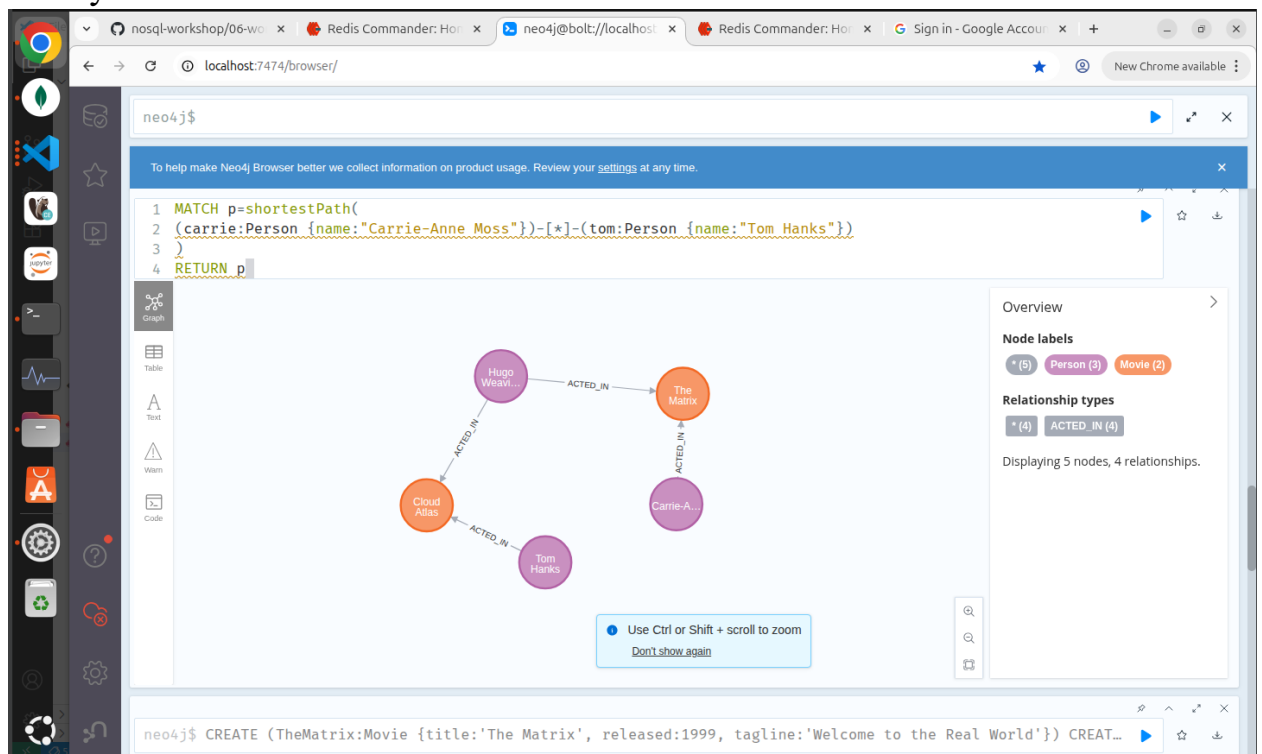
2.Вписываю код:

...

```
MATCH p=shortestPath(  
  
(carrie:Person {name:"Carrie-Anne Moss"})-[*]-(tom:Person {name:"Tom Hanks"})  
  
)  
  
RETURN p
```

...

3.Результат:



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.