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.

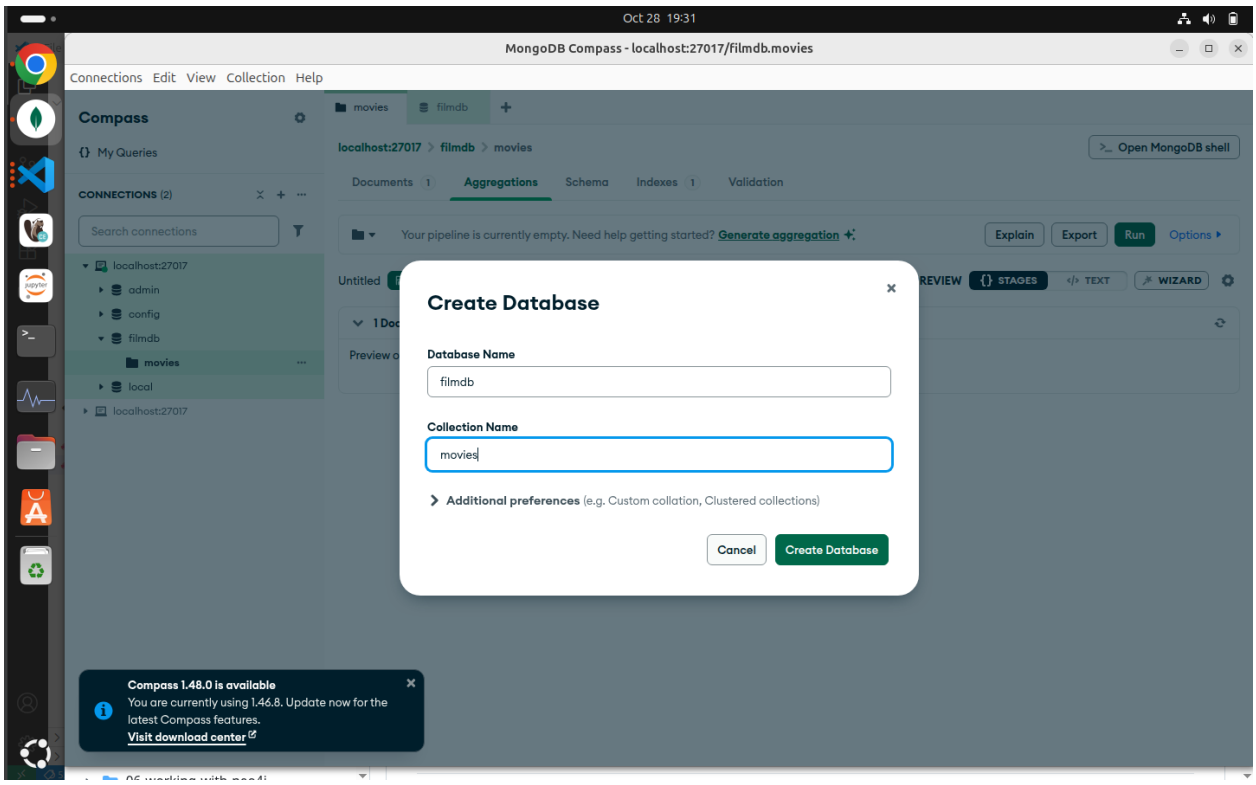
Основные структуры данных:

- 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.

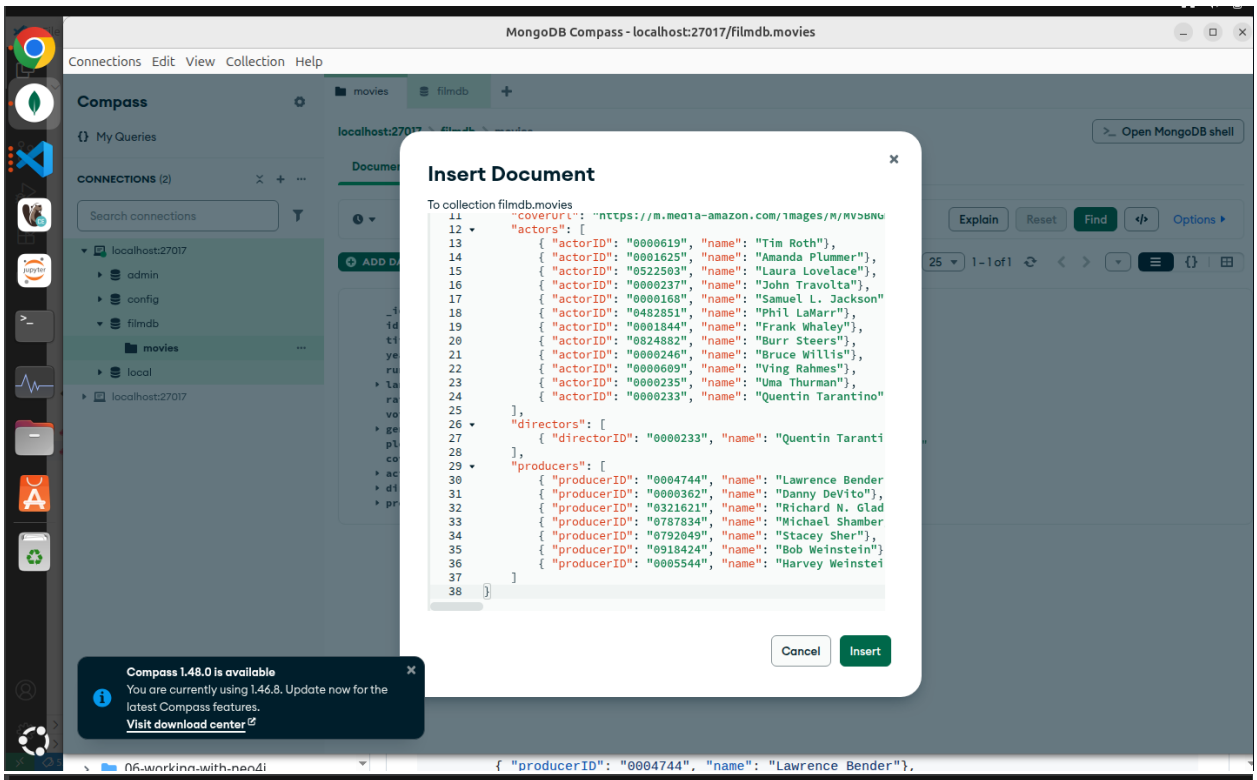
Процесс выполнения общего из GitHub

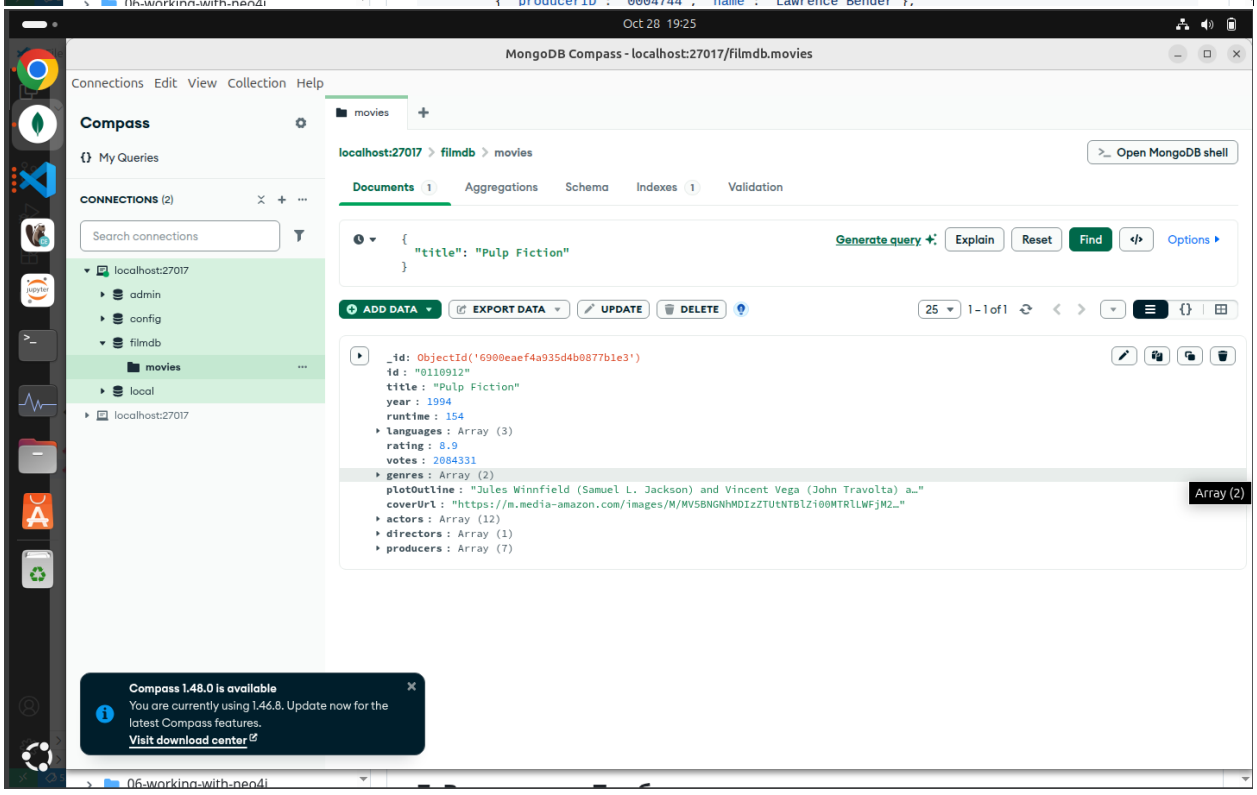
MongoDB:

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



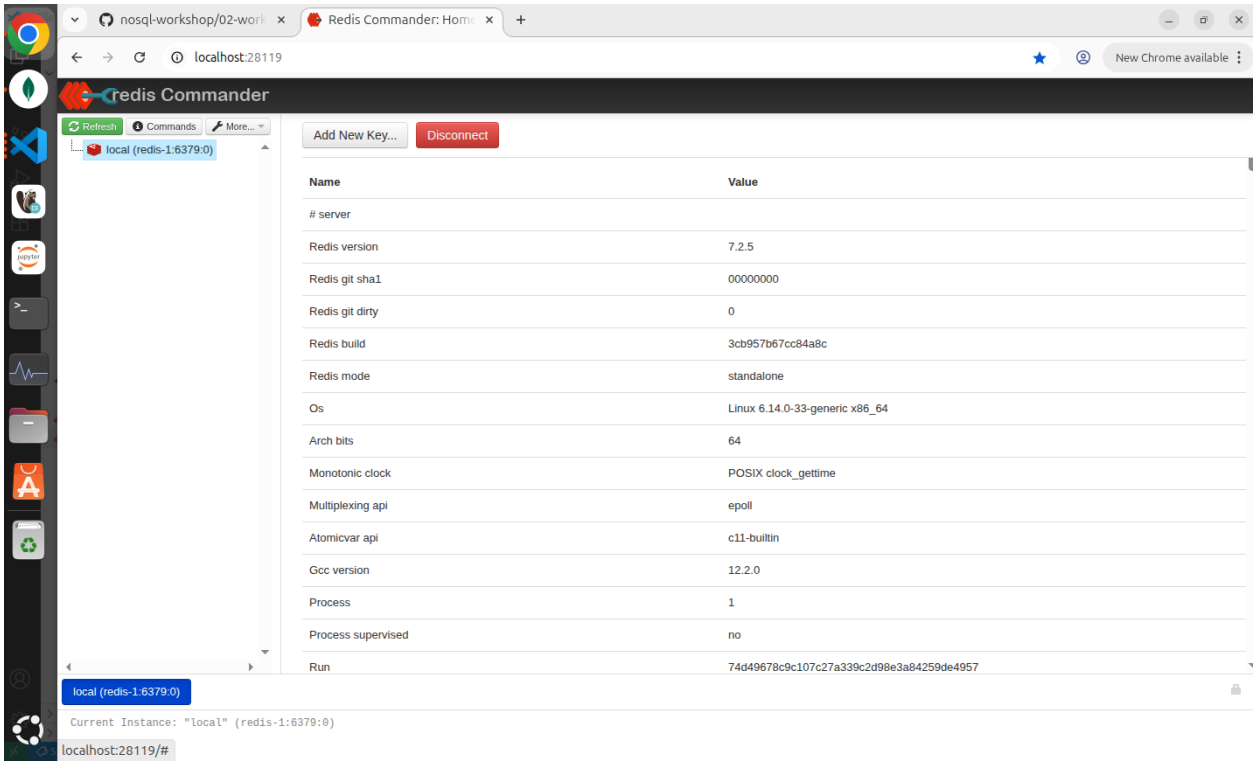
3. Загружаем данные с помощью ⊕ADD DATA → Insert document


MongoDB Compass - localhost:27017/filmdb.movies
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (2)
Search connections
localhost:27017
admin
config
filmdb
movies
local
Open MongoDB shell
Explain
Reset
Find
Options
Insert Document
To collection filmdb.movies
Cancel
Insert
Compass 1.48.0 is available
You are currently using 1.46.8. Update now for the latest Compass features.
Visit download center


Oct 28 19:25
MongoDB Compass - localhost:27017/filmdb.movies
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (2)
Search connections
localhost:27017
admin
config
filmdb
movies
local
localhost:27017
localhost:27017 > filmdb > movies
Open MongoDB shell
Documents 1
Aggregations
Schema
Indexes 1
Validation
{ "title": "Pulp Fiction" }
Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
1 - 1 of 1
_id: ObjectId('6900eaef4a935d4b0877b1e3')
id : "0110912"
title : "Pulp Fiction"
year : 1994
runtime : 154
languages : Array (3)
rating : 8.9
votes : 2084331
genres : Array (2)
plotOutline : "Jules Winnfield (Samuel L. Jackson) and Vincent Vega (John Travolta) a…"
coverUrl : "https://m.media-amazon.com/images/M/MV5BNGNhMDIzZTUtNTBlZi00MTRlLWFjM2…"
actors : Array (12)
directors : Array (1)
producers : Array (7)
Array (2)
Compass 1.48.0 is available
You are currently using 1.46.8. Update now for the latest Compass features.
Visit download center

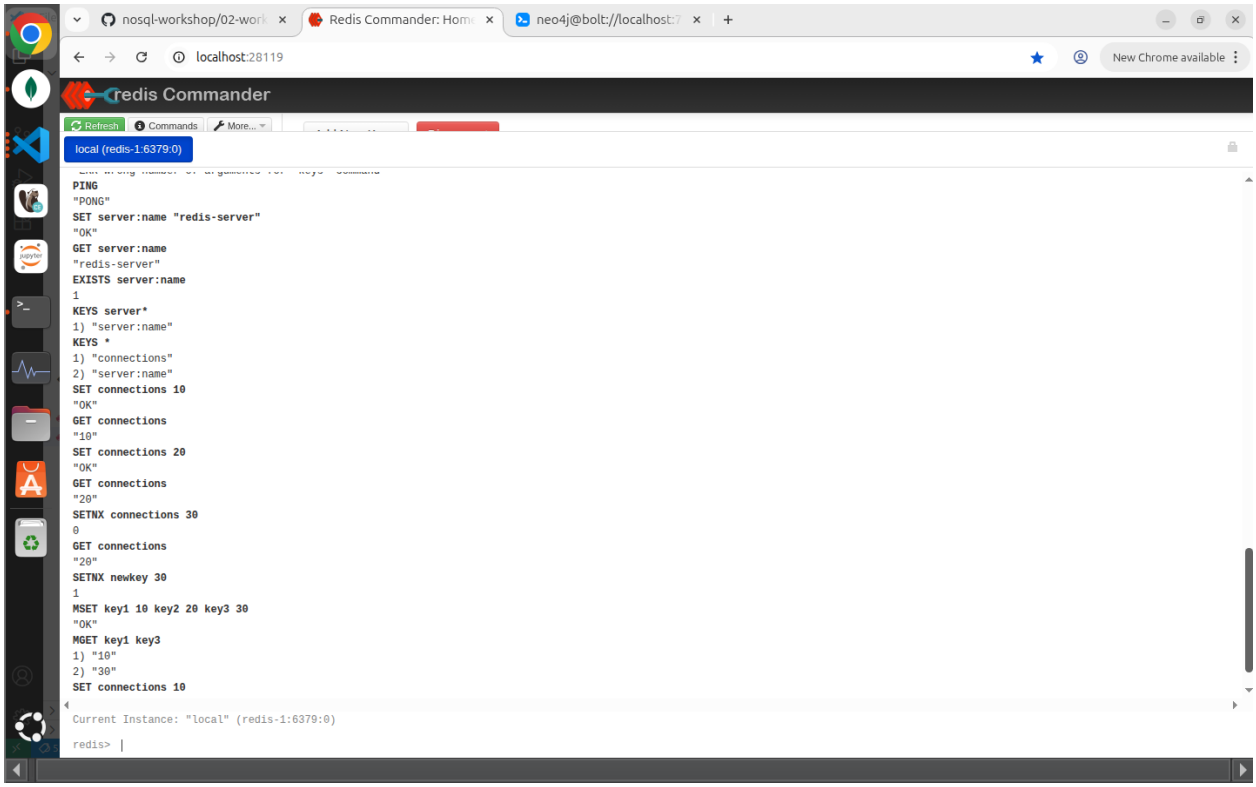
Redis:

1. Открыл Reddis Commander



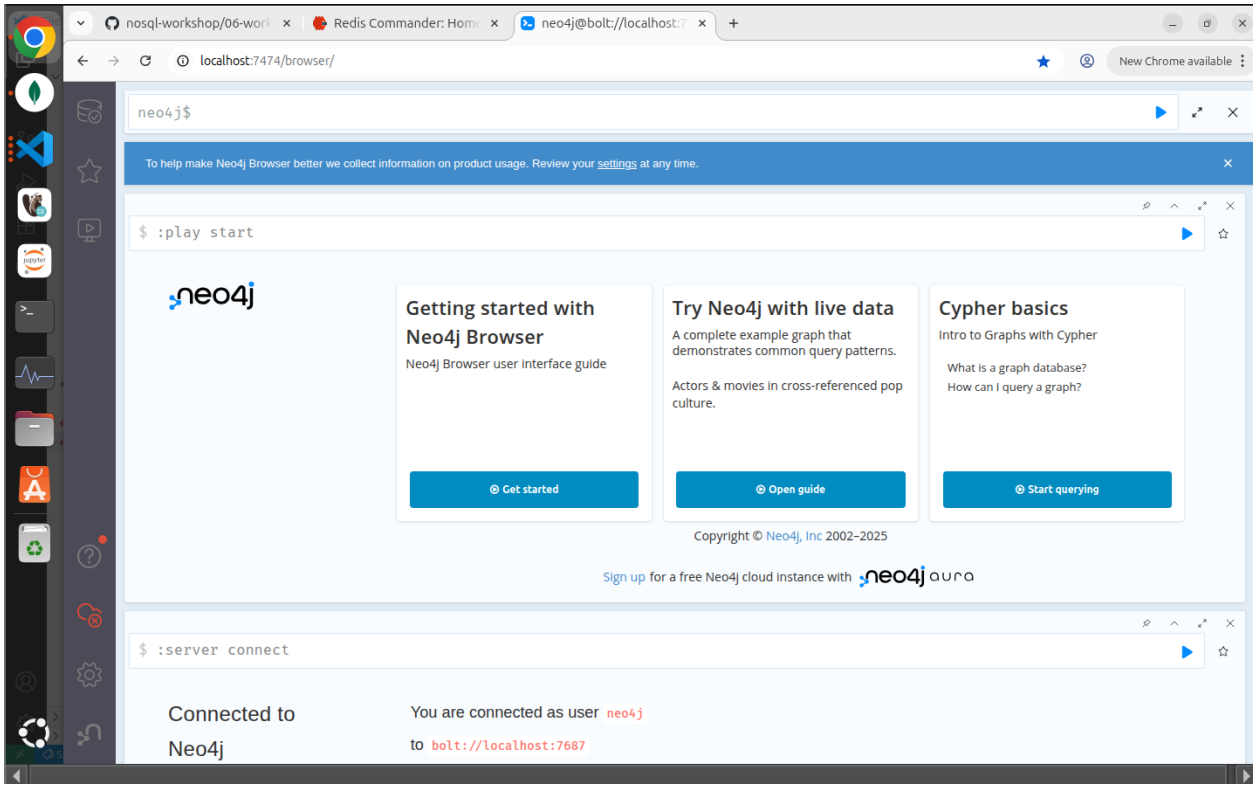
2. Выполнил все шаги из гит хаба:

```
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
```

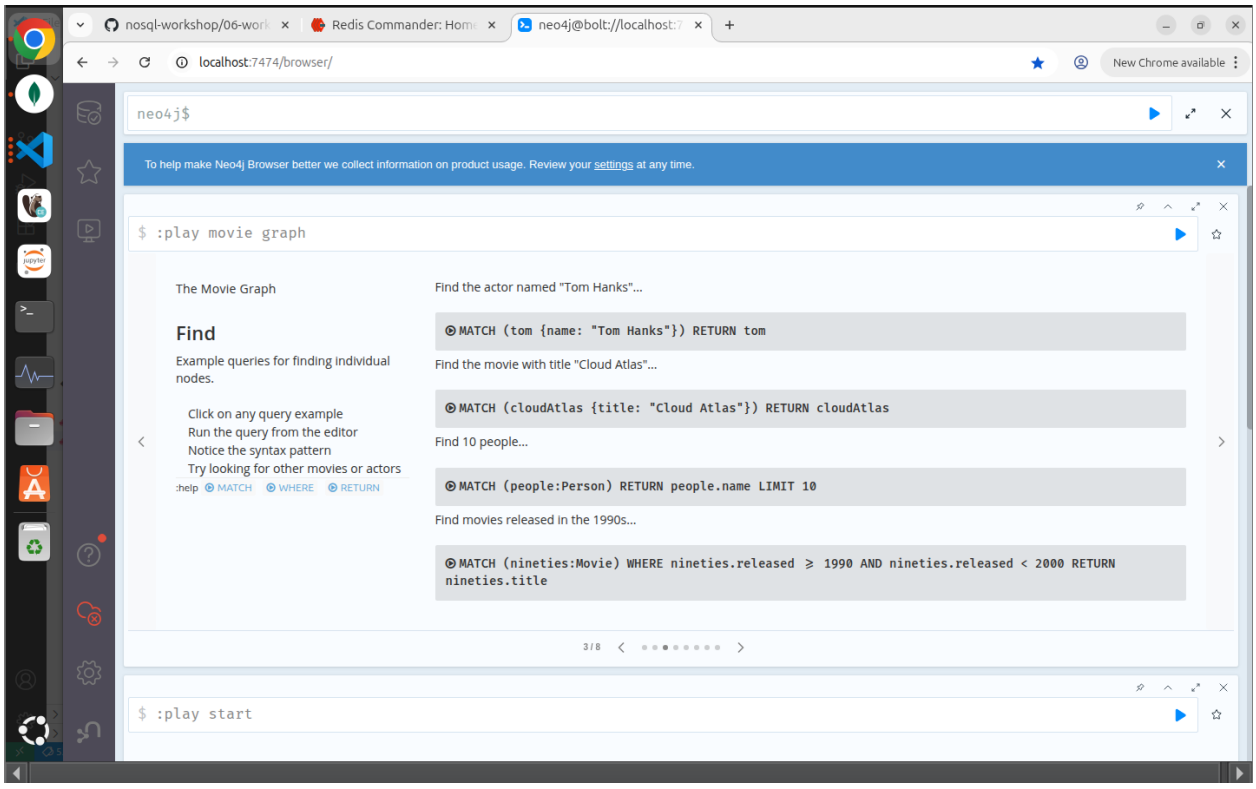

nosql-workshop/02-work
Redis Commander: Hom
neo4j@bolt://localhost:7
localhost:28119
New Chrome available
Redis Commander
Refresh
Commands
More...
local (redis-1:6379:0)
PING
"PONG"
SET server:name "redis-server"
"OK"
GET server:name
"redis-server"
EXISTS server:name
1
KEYS server*
1) "server:name"
KEYS *
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"
MGET key1 key3
1) "10"
2) "30"
SET connections 10
Current Instance: "local" (redis-1:6379:0)
redis>

Neo4j:

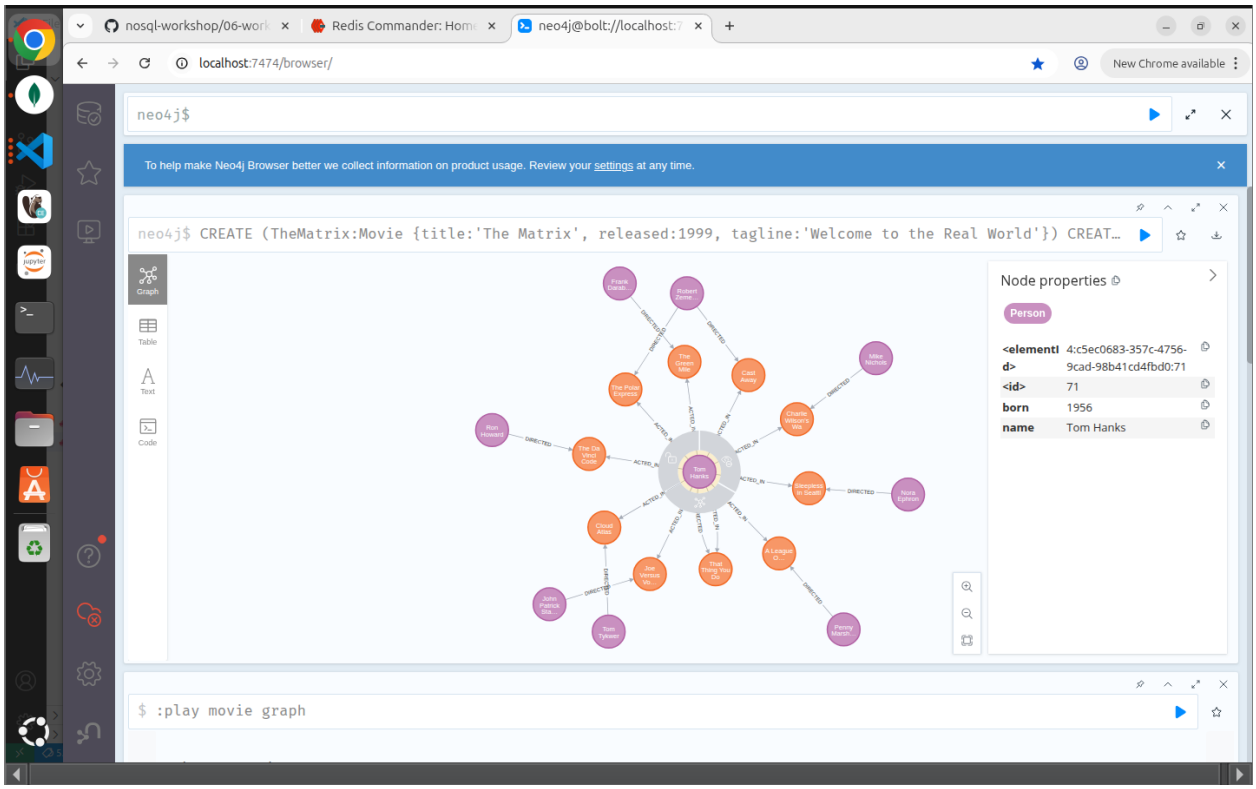
1. Подключаемся в Neo4j Browser



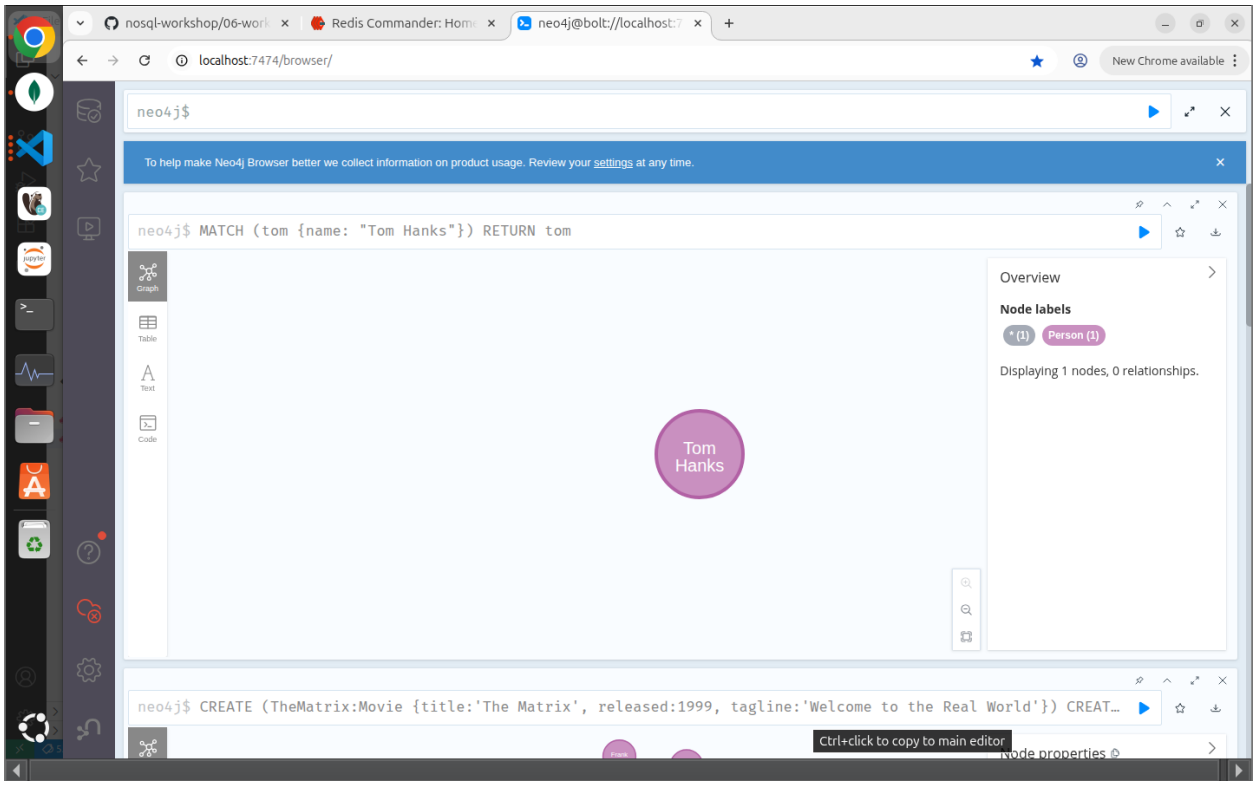
2. Выполняем команду :play movie graph



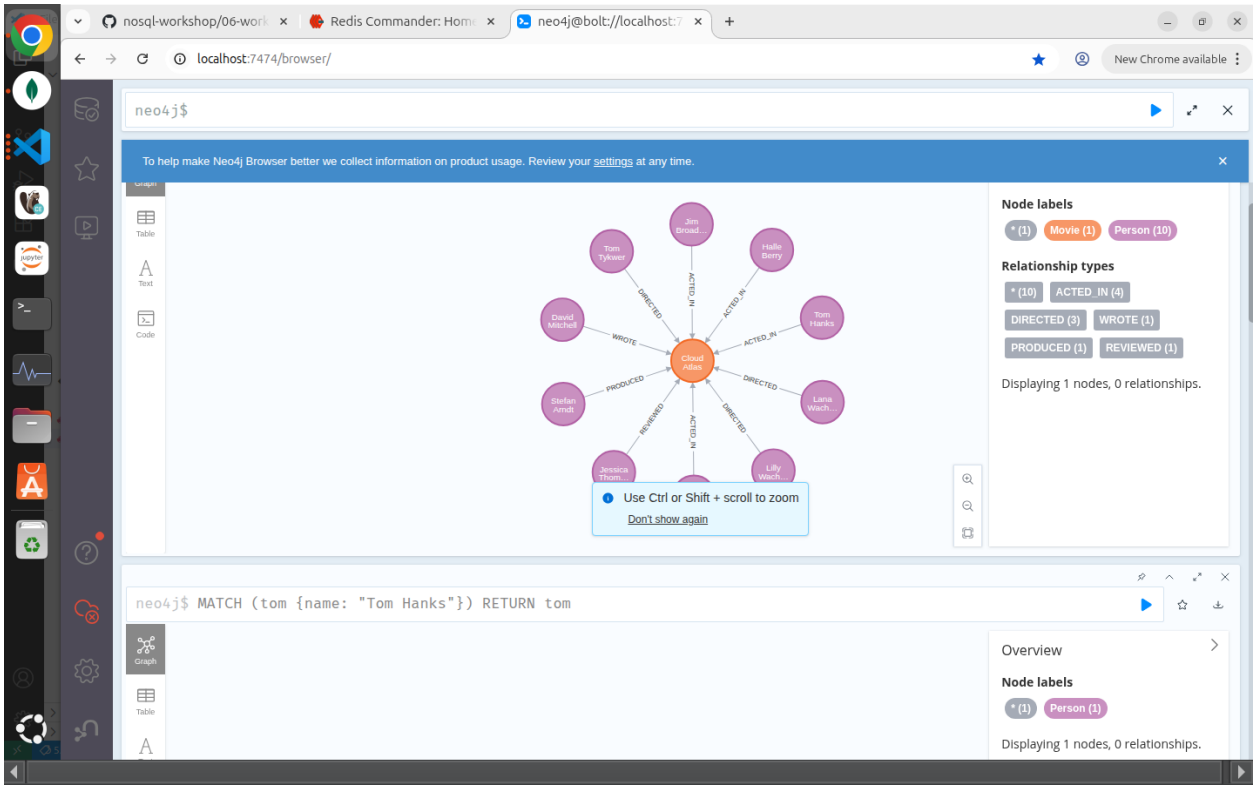
3. Создаём граф



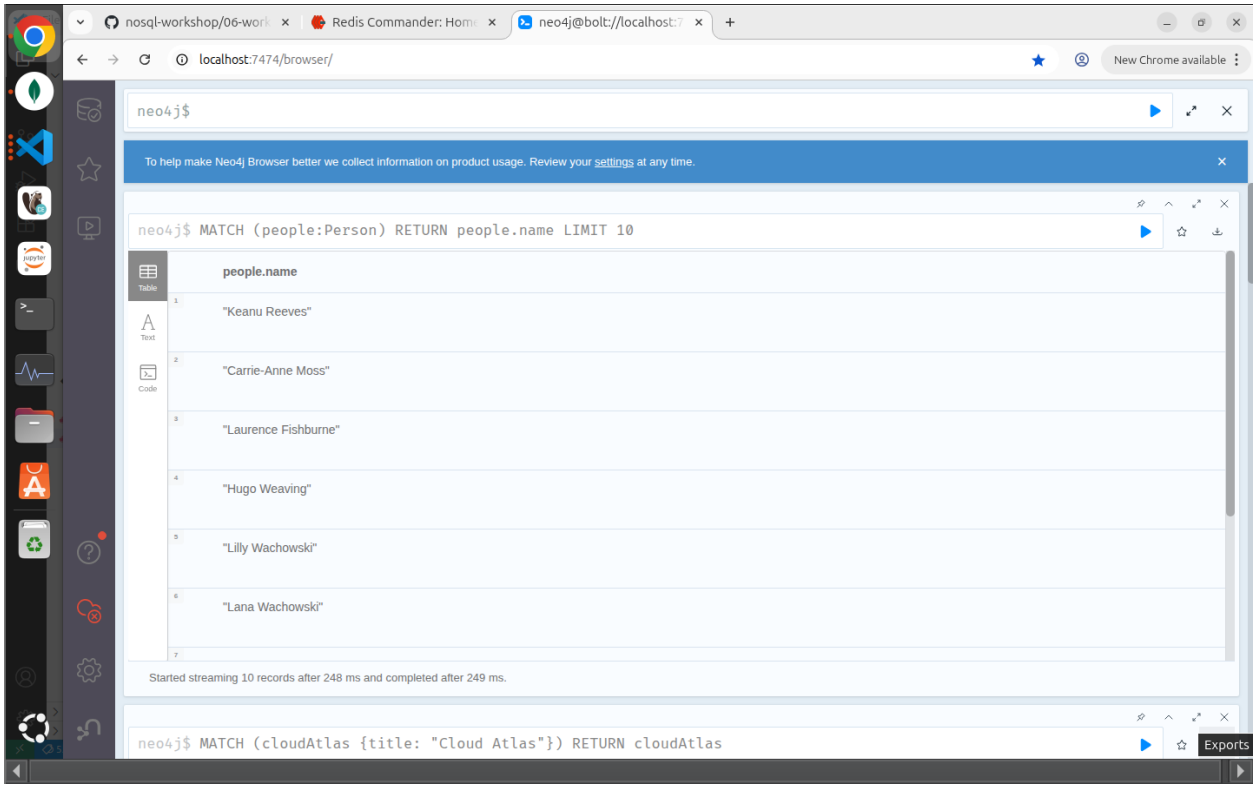
4. Выполняем несколько запросов:
    a. MATCH (tom {name: "Tom Hanks"}) RETURN tom



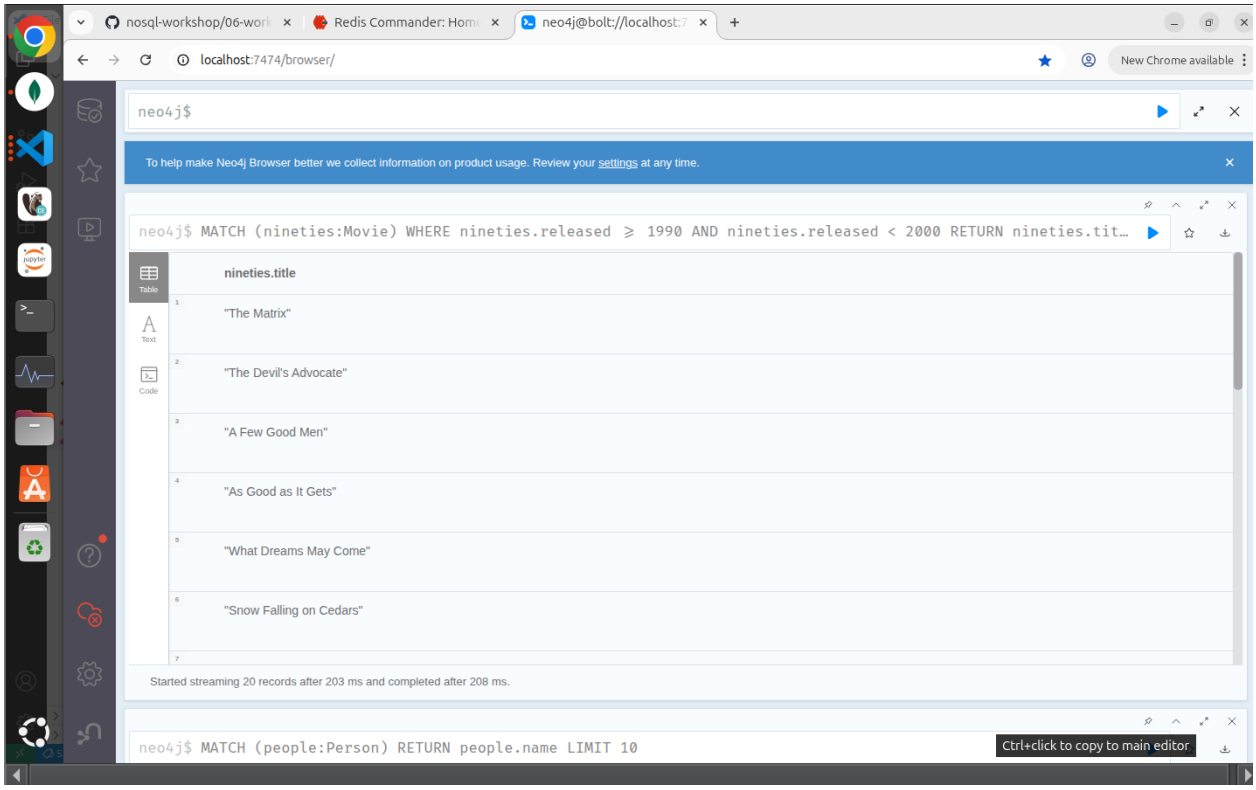
    b. MATCH (cloudAtlas {title: "Cloud Atlas"}) RETURN cloudAtlas


nosql-workshop/06-work
Redis Commander: Hom
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.
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.


nosql-workshop/06-work
Redis Commander: Hom
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 (people:Person) RETURN people.name LIMIT 10
people.name
"Keanu Reeves"
"Carrie-Anne Moss"
"Laurence Fishburne"
"Hugo Weaving"
"Lilly Wachowski"
"Lana Wachowski"
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-work
Redis Commander: Hom
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
Text
Code
nineties.title
"The Matrix"
"The Devil's Advocate"
"A Few Good Men"
"As Good as It Gets"
"What Dreams May Come"
"Snow Falling on Cedars"
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:

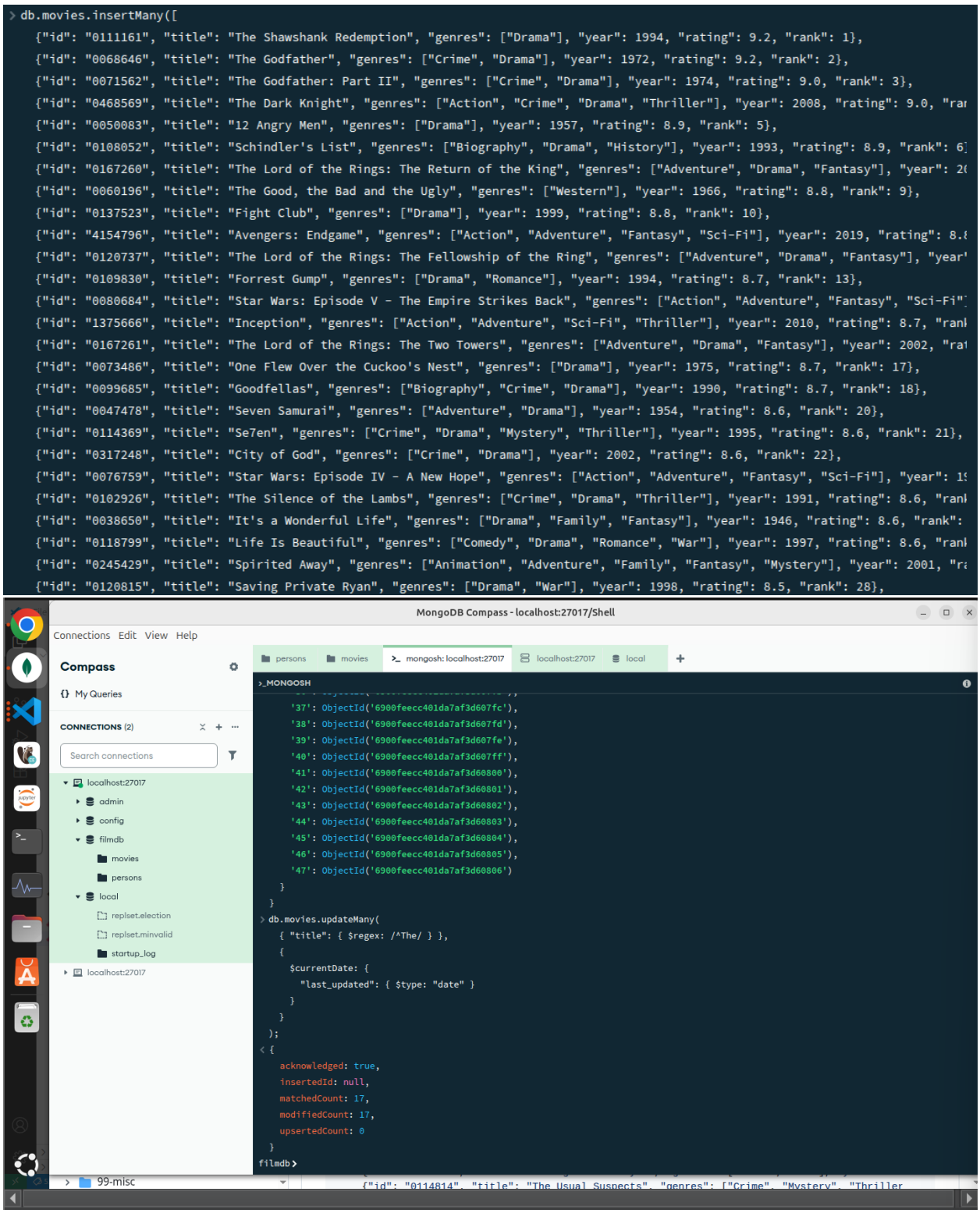

```
db.movies.updateMany(
  { "title": { $regex: /^The/ } },  // Фильтр: название начинается с "The"
  {
    $currentDate: {
      "last_updated": { $type: "date" }  // Добавляет поле с текущей датой
    }
  }
);
```

///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.

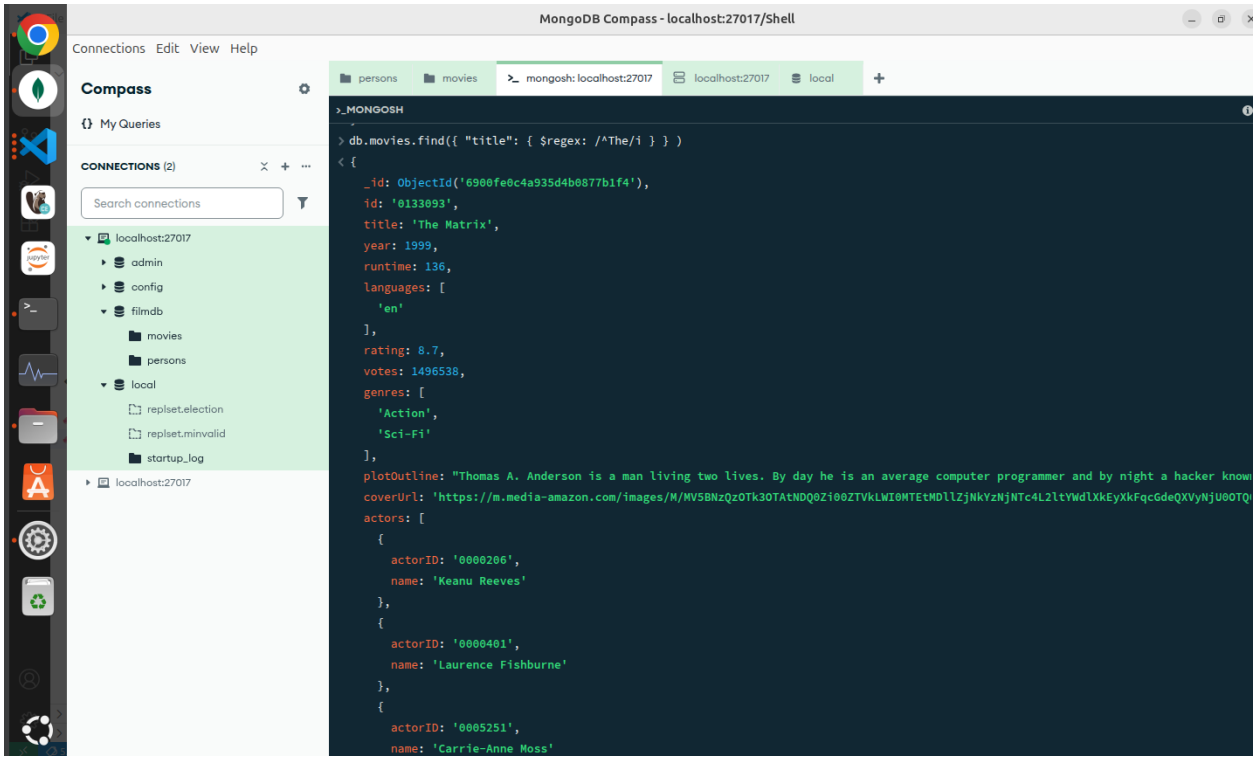
```

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:
```



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

В целях личного интереса выведем все фильмы, начинающиеся с “The” с помощью команды:

```

*db.movies.find({ "title": { $regex: /^The/i } })*
```

```

Итог:


MongoDB Compass - localhost:27017/Shell
Connections Edit View Help
Compass
My Queries
CONNECTIONS (2)
Search connections
localhost:27017
admin
config
filmdb
movies
persons
local
replset.election
replset.minvalid
startup_log
localhost:27017
persons
movies
mongosh: localhost:27017
localhost:27017
local
>_MONGOSH
> db.movies.find({ "title": { $regex: /^The/i } } )
< {
_id: ObjectId('6900fe0c4a935d4b0877b1f4'),
id: '0133093',
title: 'The Matrix',
year: 1999,
runtime: 136,
languages: [
'en'
],
rating: 8.7,
votes: 1496538,
genres: [
'Action',
'Sci-Fi'
],
plotOutline: "Thomas A. Anderson is a man living two lives. By day he is an average computer programmer and by night a hacker known
coverUrl: 'https://m.media-amazon.com/images/M/MV5BNzQzOTk3OTAtNDQ0Zi00ZTVkLWI0MTEtMDllZjNkYzNjNTc4L2ltYWdlXkEyXkFqcGdeQXVyNjU0OTQ
actors: [
{
actorID: '0000206',
name: 'Keanu Reeves'
},
{
actorID: '0000401',
name: 'Laurence Fishburne'
},
{
actorID: '0005251',
name: 'Carrie-Anne Moss'
```

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

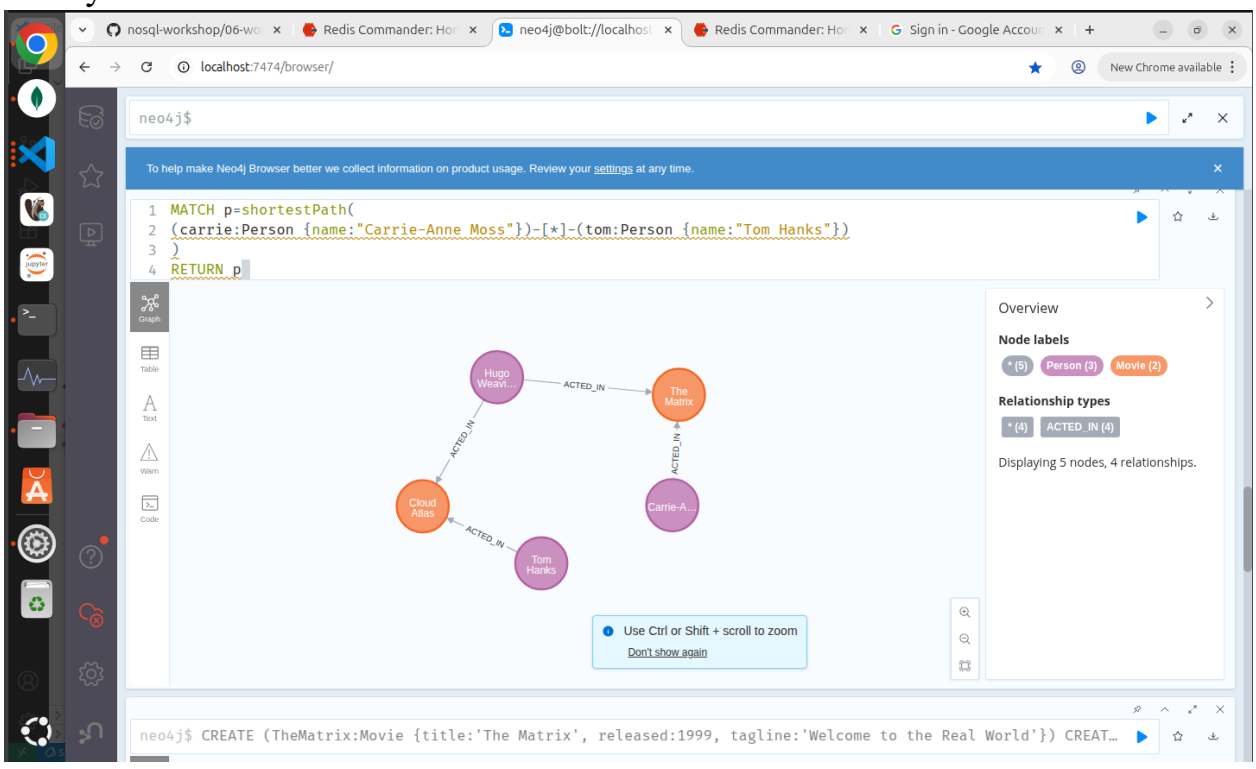
Цель: Найти кратчайший путь между "CarrieAnne Moss" и "Tom Hanks".

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.