

Государственное автономное образовательное учреждение  
высшего образования города Москвы  
«Московский городской педагогический университет»

Институт цифрового образования  
Департамент информатики, управления и технологий

**Лабораторная работа №2.1 (вариант №12)**

**Тема: «Изучение методов хранения данных на основе NoSQL»**

Дисциплина «Инструменты для хранения и обработки больших данных»

Направление подготовки 38.03.05 – бизнес-информатика  
Профиль подготовки «Аналитика данных и эффективное управление»  
(очная форма обучения)

Выполнила:

Студентка группы АДЭУ-221

Сабитова Алина Булатовна

Преподаватель:

Босенко Тимур Муртазович

Москва

2025

**Цель работы:** изучение трех различных типов NoSQL баз данных: MongoDB, Neo4j, Redis, освоение основных команд для работы с ними и последующее применение полученных знаний для решения практических задач.

Запустим все контейнеры докера:

```
● mgpu@mgpu-vm:~/Downloads/idb/nosql-workshop/01-environment/docker$ sudo docker compose up -d
[sudo] password for mgpu:
[+] Running 9/9
 ✓ Container redis-1           Running      0.0s
 ✓ Container neo4j-1           Running      0.0s
 ✓ Container admin-mongo       Running      0.0s
 ✓ Container cassandra-web     Running      0.0s
 ✓ Container redis-commander   Running      0.0s
 ✓ Container mongo-express     Running      0.0s
 ✓ Container cassandra-1       Running      0.0s
 ✓ Container jupyter           Running      0.0s
 ✓ Container mongo-1           Running      0.0s
```

## MongoDB

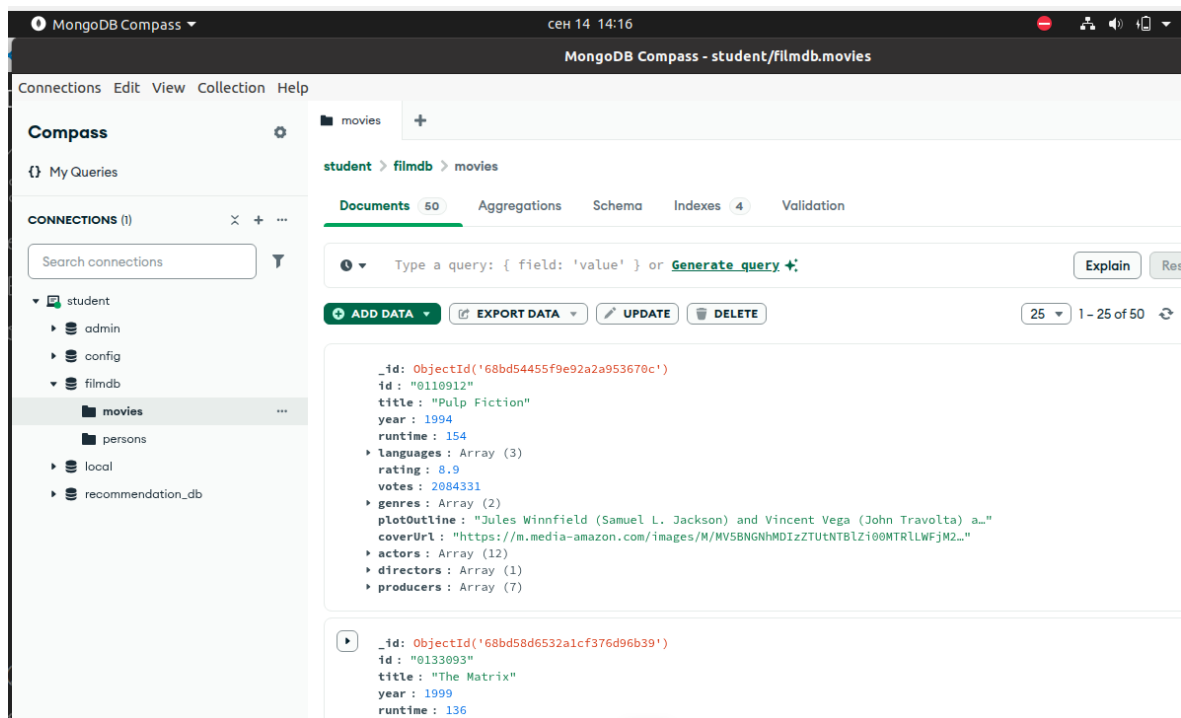
Задание: найти 5 самых старых фильмов в коллекции и обновить у них поле `is_vintage` на `true`.

Шаги выполнения:

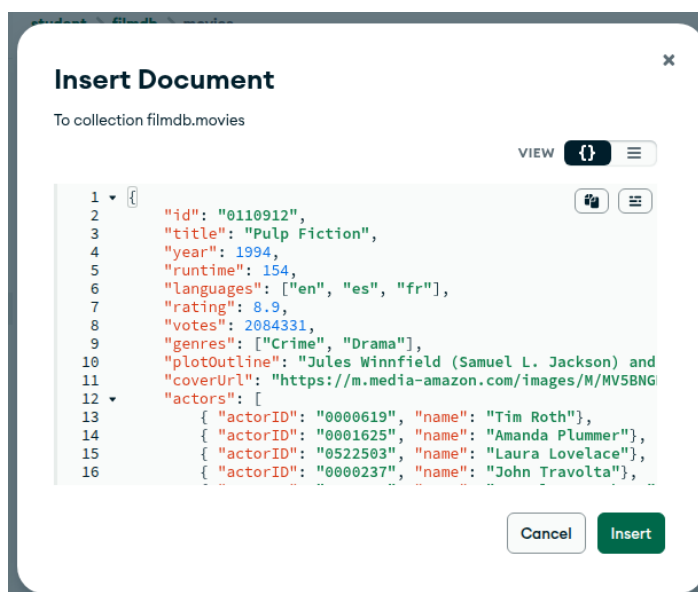
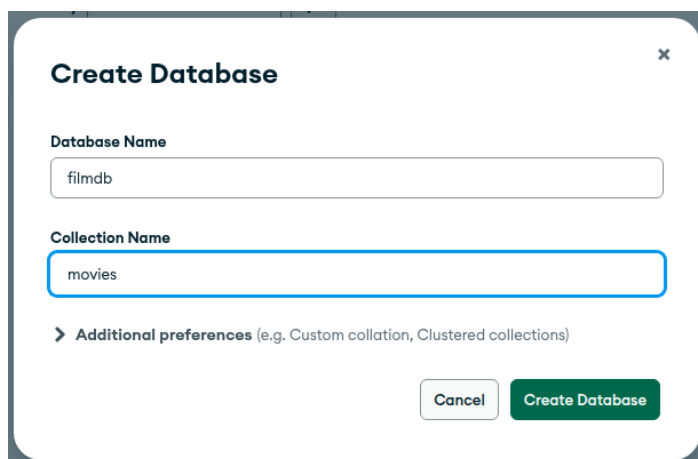
1. Подключимся к контейнеру Mongo и запустим оболочки внутри него:

```
○ mgpu@mgpu-vm:~/Downloads/idb/nosql-workshop/01-environment/docker$ sudo docker exec -ti mongo-1 mongo -u "root" -p "abc123!" --authenticationDatabase admin
MongoDB shell version v4.4.29
connecting to: mongodb://127.0.0.1:27017/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("3ed4b8d4-1c4a-4a31-bc12-098fb7f093cf") }
MongoDB server version: 4.4.29
---
The server generated these startup warnings when booting:
  2025-09-13T16:58:56.477+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
---
```

2. Создадим подключение в MongoDB Compass:



3. Создадим базу данных filmdb и коллекцию movies:



4. Добавим еще 49 фильм через терминал для того, чтобы работать со списком 50 лучших фильмов из IMDB:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
> db.movies.insertOne(
... {
...   "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 as Neo. Neo has always questioned his reality, but the truth is far beyond his imagi-
nation. Neo finds himself targeted by the police when he is contacted by Morpheus, a legendary computer hacker br-
anded a terrorist by the government. Morpheus awakens Neo to the real world, a ravaged wasteland where most of hu-
manity have been captured by a race of machines that live off of the humans' body heat and electrochemical energy
and who imprison their minds within an artificial reality known as the Matrix. As a rebel against the machines,
Neo must return to the Matrix and confront the agents: super-powerful computer programs devoted to snuffing out N
eo and the entire human rebellion.",
...   "coverUrl": "https://m.media-amazon.com/images/M/MV5BNzQzOTk3OTAtNDQ0Zi00ZTVkLWI0MTEtMDllZjNkYzNjNTc4L2lt
YWdlXkEyXkFqcGdeQXVyNjU0OTQ0OTY@._V1_SX101_CR0,0,101,150_.jpg",
...   "actors": [
...     { "actorID": "0000206", "name": "Keanu Reeves"},

```

Select Indentation

```
Ln 222, Col 28 Spaces: 2 UTF-8 LF {} Compose
```

```
> db.movies.insertMany([
... {"id": "0111161", "title": "The Shawshank Redemption", "genres": ["Drama"], "year": 1994, "rating": 9.2, "ran-
k": 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": 200
8, "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, "ra-
ting": 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", "Dram
a", "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"],

```

Select Language Mode

```
Ln 222, Col 28 Spaces: 2 UTF-8 LF {} Compose
```

Убедимся, что их действительно 50:

```
> db.movies.find().count()
50
>
```

5. Приступим к выполнению индивидуального задания. Найдем 5 самых старых фильмов:

```
db.movies.find().sort({ "year": 1 }).limit(5).pretty()
```

```
> db.movies.find().sort({"year": 1}).limit(5).pretty()
{
  "_id" : ObjectId("68bd5d02532a1cf376d96b5e"),
  "id" : "0021749",
  "title" : "City Lights",
  "genres" : [
    "Comedy",
    "Drama",
    "Romance"
  ],
  "year" : 1931,
  "rating" : 8.5,
  "rank" : 35
}
{
  "_id" : ObjectId("68bd5d02532a1cf376d96b62"),
  "id" : "0027977",
  "title" : "Modern Times",
  "genres" : [
    "Comedy",
    "Drama",
    "Family",
    "Romance"
  ],
  "year" : 1936,
  "rating" : 8.5,
```

```

    "rank" : 39
  }
  {
    "_id" : ObjectId("68bd5d02532a1cf376d96b5f"),
    "id" : "0034583",
    "title" : "Casablanca",
    "genres" : [
      "Drama",
      "Romance",
      "War"
    ],
    "year" : 1942,
    "rating" : 8.5,
    "rank" : 36
  }
  {
    "_id" : ObjectId("68bd5d02532a1cf376d96b54"),
    "id" : "0038650",
    "title" : "It's a Wonderful Life",
    "genres" : [
      "Drama",
      "Family",
      "Fantasy"
    ],
    "year" : 1946,
    "rating" : 8.6,
    "rank" : 25
  }
}
```

```

{
  "_id" : ObjectId("68bd5d02532a1cf376d96b4f"),
  "id" : "0047478",
  "title" : "Seven Samurai",
  "genres" : [
    "Adventure",
    "Drama"
  ],
  "year" : 1954,
  "rating" : 8.6,
  "rank" : 20
}
>
```

Это фильмы «City Lights», «Modern Times», «Casablanca», «It's a wonderful life» и «Seven Samurai».

6. Для каждого из них зададим значение поля `is_vintage: true` с помощью команды `updateOne()`:

```
db.movies.updateOne({title: "film_title"}, {$set: {is_vintage: true}})
```

```

> db.movies.updateOne({title: "City Lights"}, {$set: {is_vintage: true}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.movies.updateOne({title: "Modern Times"}, {$set: {is_vintage: true}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.movies.updateOne({title: "Casablanca"}, {$set: {is_vintage: true}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.movies.updateOne({title: "It's a Wonderful Life"}, {$set: {is_vintage: true}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.movies.updateOne({title: "Seven Samurai"}, {$set: {is_vintage: true}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

```

7. С помощью команды `find()` проверим, что поле `is_vintage` действительно приобрело нужное нам значение:

`db.movies.find().sort({"year": 1}).limit(5).pretty()`

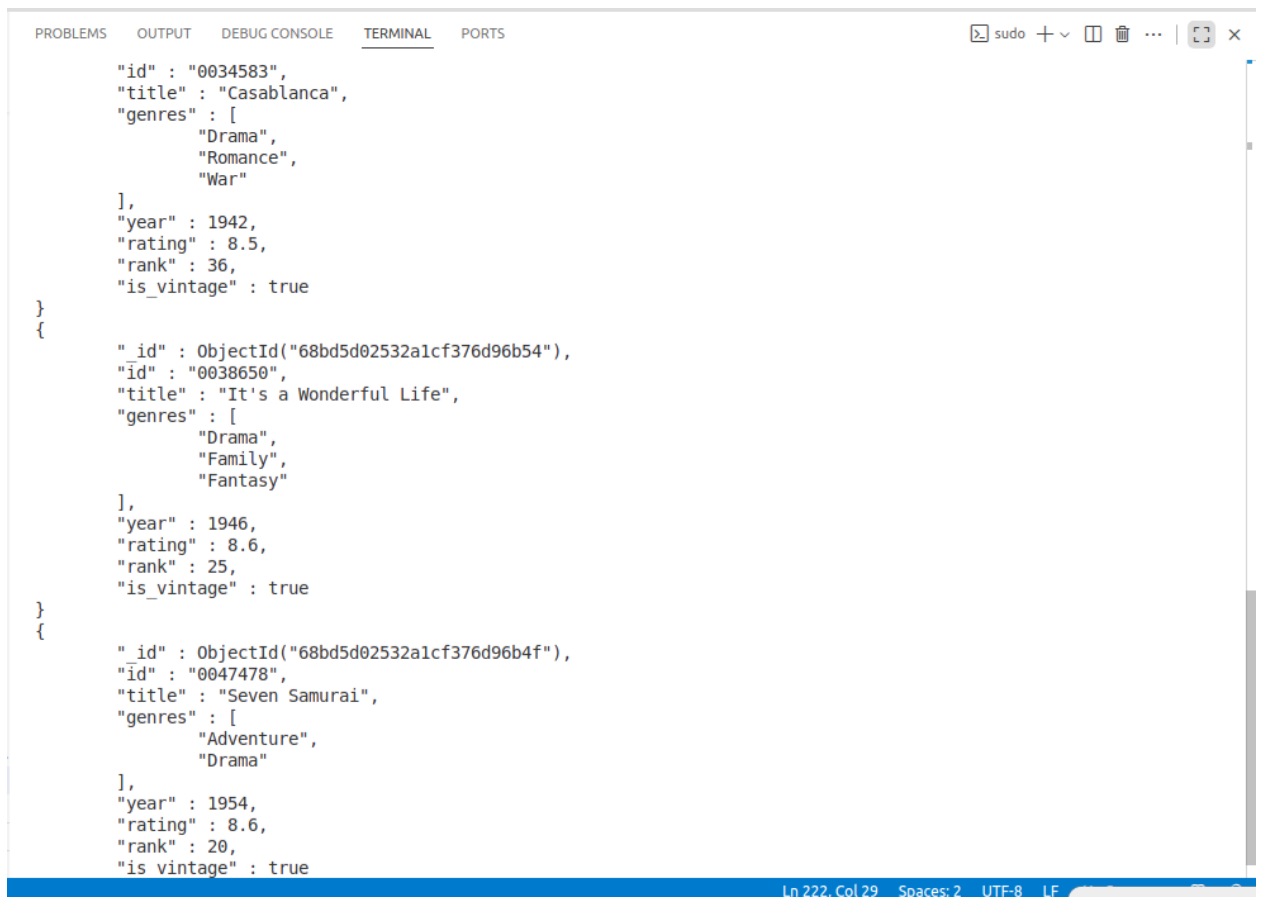


```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
> db.movies.find().sort({"year": 1}).limit(5).pretty()
{
  "_id" : ObjectId("68bd5d02532a1cf376d96b5e"),
  "id" : "0021749",
  "title" : "City Lights",
  "genres" : [
    "Comedy",
    "Drama",
    "Romance"
  ],
  "year" : 1931,
  "rating" : 8.5,
  "rank" : 35,
  "is_vintage" : true
}
{
  "_id" : ObjectId("68bd5d02532a1cf376d96b62"),
  "id" : "0027977",
  "title" : "Modern Times",
  "genres" : [
    "Comedy",
    "Drama",
    "Family",
    "Romance"
  ],
  "year" : 1936,
  "rating" : 8.5,
  "rank" : 39,
  "is_vintage" : true
}
{
  "_id" : ObjectId("68bd5d02532a1cf376d96b5f"),
  "id" : "0034583",
  "title" : "Casablanca",
  "genres" : [
    "Drama",
    "Romance",
    "War"
  ],
  "year" : 1942,
  "rating" : 8.5,
  "rank" : 36,
  "is_vintage" : true
}

```

Ln 222, Col 29 Spaces: 2 UTF-8 LF {} Compose



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
{id : "0034583",
title : "Casablanca",
genres : [
  "Drama",
  "Romance",
  "War"
],
year : 1942,
rating : 8.5,
rank : 36,
is_vintage : true
}
{
  "_id" : ObjectId("68bd5d02532a1cf376d96b54"),
  id : "0038650",
  title : "It's a Wonderful Life",
  genres : [
    "Drama",
    "Family",
    "Fantasy"
  ],
  year : 1946,
  rating : 8.6,
  rank : 25,
  is_vintage : true
}
{
  "_id" : ObjectId("68bd5d02532a1cf376d96b4f"),
  id : "0047478",
  title : "Seven Samurai",
  genres : [
    "Adventure",
    "Drama"
  ],
  year : 1954,
  rating : 8.6,
  rank : 20,
  is_vintage : true
}
```

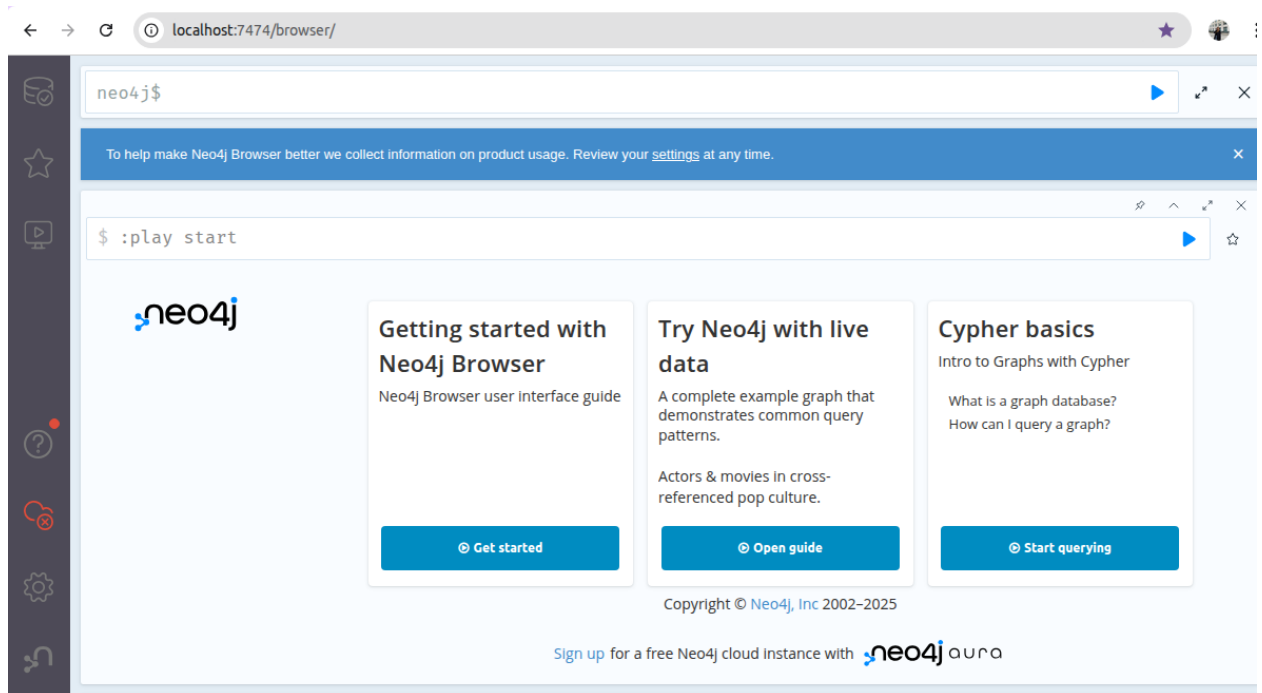
Видим, что у всех фильмов появилось поле `is_vintage` со значением `true`.

## Neo4j

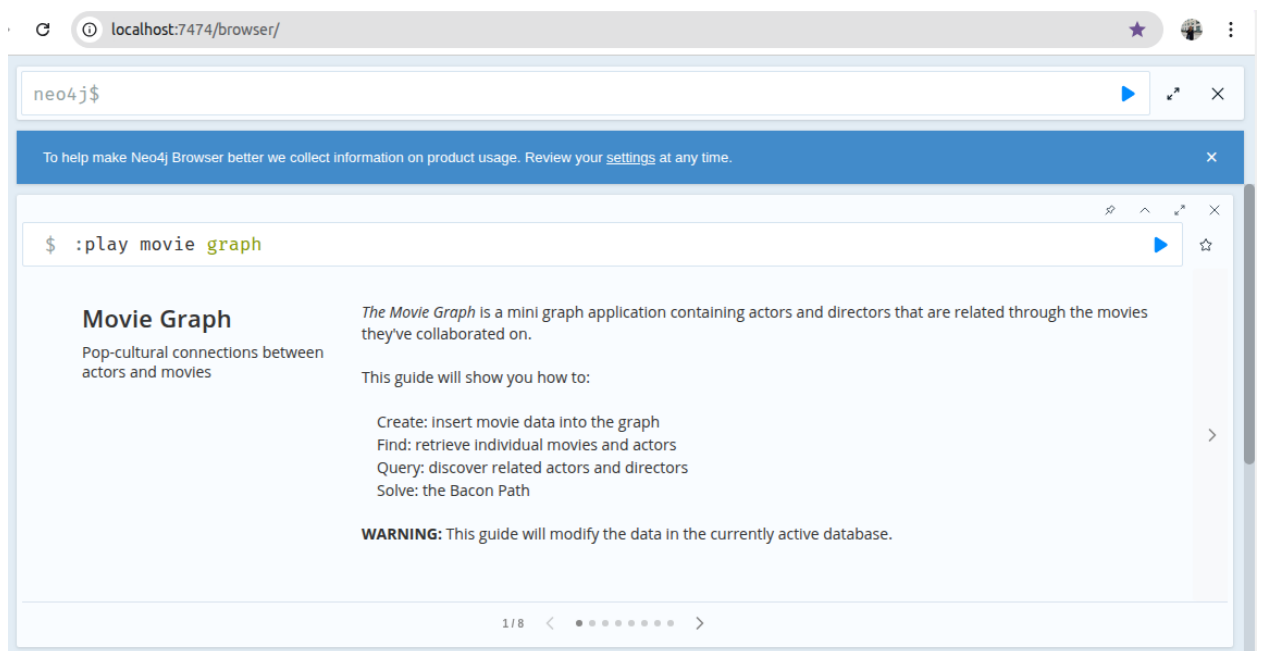
Задание: найти всех актеров, которые не снимались в фильмах с "Tom Hanks".

Шаги выполнения:

1. Подключимся к Neo4j Browser:

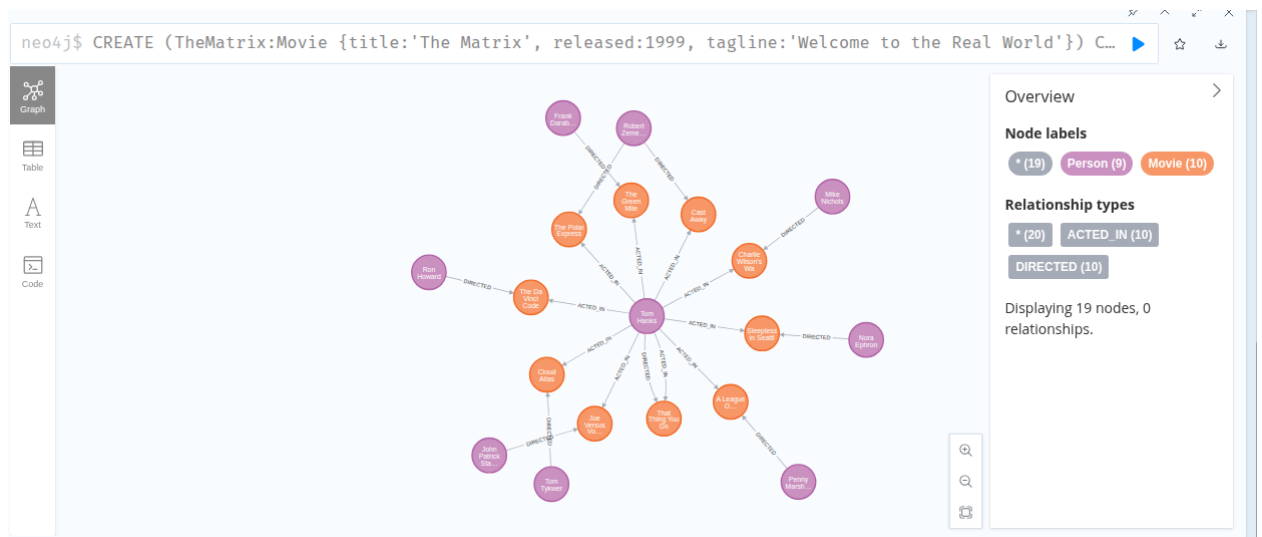


## 2. Начнем работу с учебным пособием Movie Graph:



## 3. Загрузим граф фильмов:





4. Найдем актеров, которые никогда не снимались с Томом Хэнксом.

Запрос:

MATCH (p:Person)-[:ACTED\_IN]-&gt;(m:Movie)

WHERE NOT (p)-[:ACTED\_IN]->(:Movie)<-[:ACTED\_IN]-(:Person  
{name:"Tom Hanks"}) AND p.name <> "Tom Hanks"

RETURN DISTINCT p.name

Результат выполнения:



Видим, что 67 актеров никогда не снимались с Томом Хэнксом.

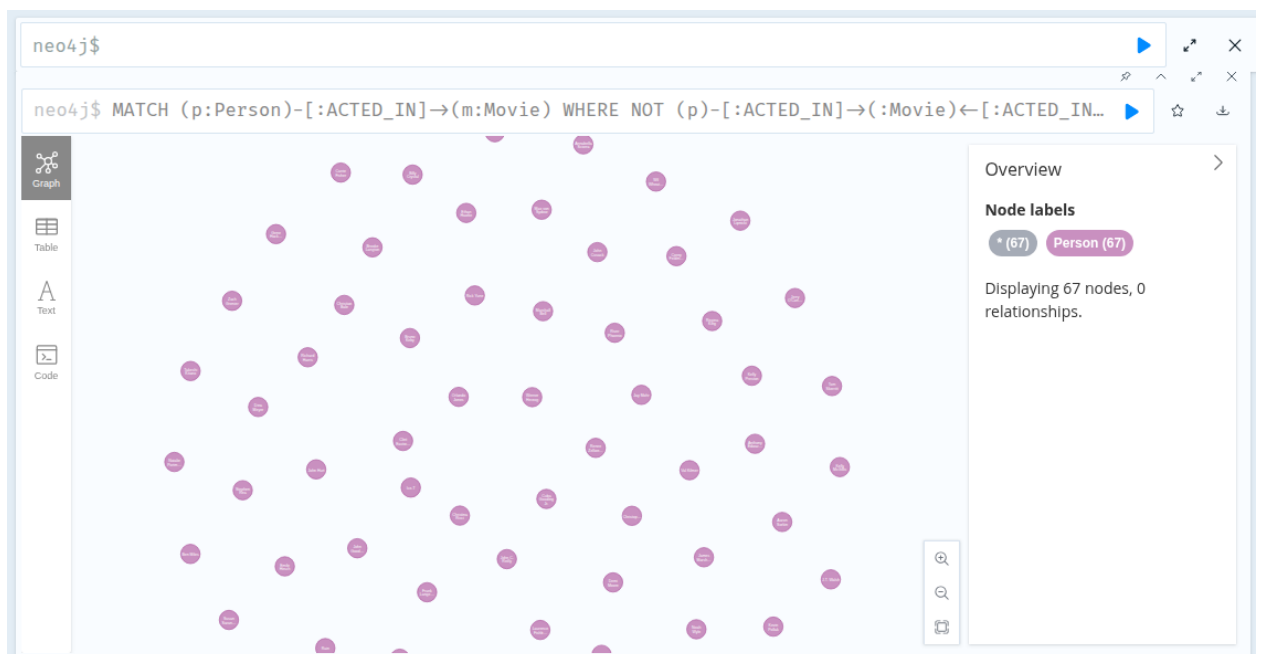
Можно также отобразить в виде графа. Запрос:

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
```

```
WHERE NOT (p)-[:ACTED_IN]->(:Movie)<-[:ACTED_IN]-(:Person  
{name:"Tom Hanks"}) AND p.name <> "Tom Hanks"
```

```
RETURN p
```

Результат выполнения:



## Redis

Задание: смоделировать права доступа: создать множество user:33:permissions и добавить в него права "read", "write". Проверить, есть ли у пользователя право "delete" (SISMEMBER).

Шаги выполнения:

1. Запустим Redis CLI в docker-контейнере:

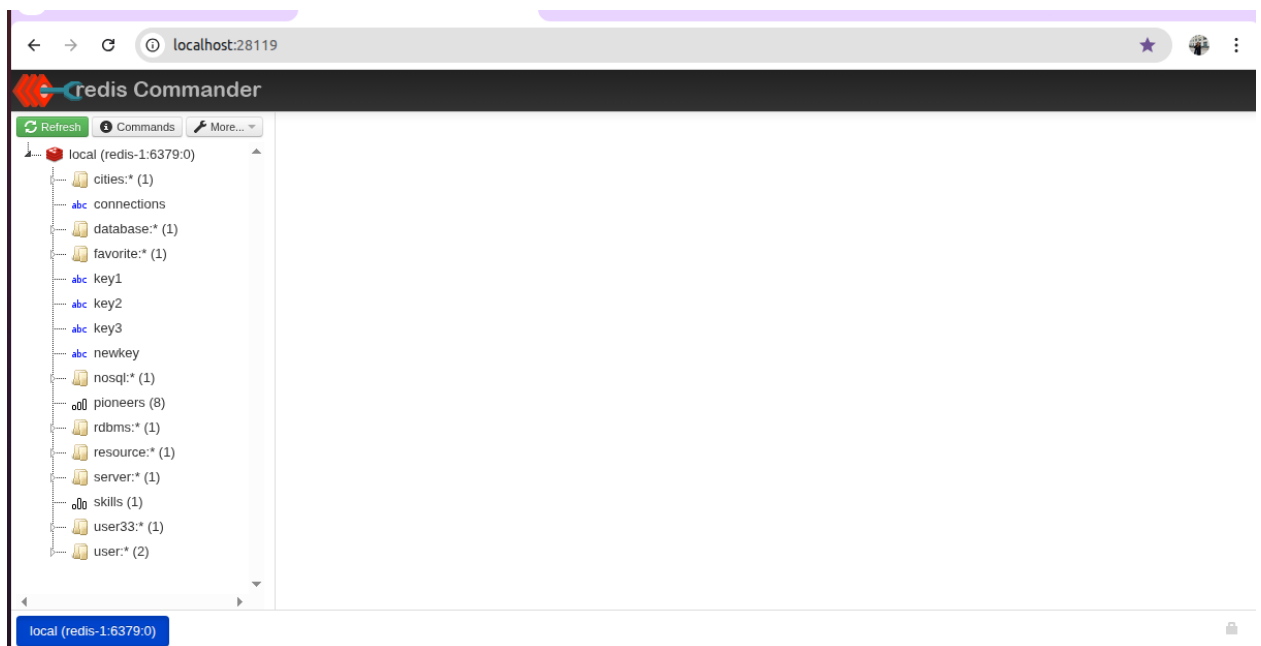
```

mgpu@mgpu-vm:~/Downloads/idb/nosql-workshop/01-environment/docker$ docker run -it --rm --network nosql-platform bitnami/redis redis-cli -h redis-1 -p 6379
redis 17:02:10.96 INFO ==>
redis 17:02:10.97 INFO ==> Welcome to the Bitnami redis container
redis 17:02:10.98 INFO ==> Subscribe to project updates by watching https://github.com/bitnami/containers
redis 17:02:10.99 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 17:02:11.00 INFO ==>

redis-1:6379> docker run -it --rm --network nosql-platform bitnami/redis redis-benchmark -h redis-1 -a "abc123!" -q -n 100000
(error) ERR unknown command 'docker', with args beginning with: 'run' '-it' '--rm' '--network' 'nosql-platform' 'bitnami/redis' 'redis-benchmark' '-h' 'redis-1' '-a' 'abc123!' '-q' '-n' '100000'
redis-1:6379>

```

## 2. Создадим подключение в Redis Commander:



## 3. Создадим множество user:33:permissions и добавим в него значение «read»:

```

SADD user:33:permissions "read"
1

```

## 4. Теперь добавим значение «write»:

```

SADD user:33:permissions "write"
1

```

## 5. Проверим, добавились ли в множество указанные нами права:

```

SMEMBERS user:33:permissions
1) "read"
2) "write"

```

## 6. Проверим, есть ли у пользователя право «delete»:

```
SISMEMBER user:33:permissions "delete"
```

```
0
```

Видим результат — 0, значит такое значение в множестве отсутствует.

Последовательность выполненных команд:

```
SADD user:33:permissions "read"
```

```
SADD user:33:permissions "write"
```

```
SMEMBERS user:33:permissions
```

```
SISMEMBER user:33:permissions "delete"
```

## Работа в JupyterLab

Рассмотрим взаимодействие с MongoDB через язык Python.

### 1. Установка pymongo:

```
[1]: !pip install pymongo
Requirement already satisfied: pymongo in /opt/conda/lib/python3.11/site-packages (4.14.1)
Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in /opt/conda/lib/python3.11/site-packages (from pymongo) (2.8.0)

[2]: from pymongo import MongoClient
```

### 2. Подключение к базе данных:

```
[13]: try:
      client = MongoClient("mongodb://root:abc123!@mongo-1:27017/")
      db = client.student
      collection = db.test_labs

      client.server_info()
      print("✅ Успешное подключение к MongoDB!")
    except Exception as e:
      print(f"❌ Ошибка подключения: {e}")

✅ Успешное подключение к MongoDB!
```

### 3. Очистка коллекции перед началом работы:

```
[14]: # Очистка коллекции перед началом работы
      collection.delete_many({})

[14]: DeleteResult({'n': 0, 'ok': 1.0}, acknowledged=True)
```

### 4. Вставка данных в коллекцию:

```
[16]: # 4. Вставка данных (Create)
test_data = [
    {"lab_name": "Lab 1", "subject": "Physics", "score": 85},
    {"lab_name": "Lab 2", "subject": "Chemistry", "score":
90},
    {"lab_name": "Lab 3", "subject": "Biology", "score": 88},
]
result = collection.insert_many(test_data)
print(f"\nВставлено документов: {len(result.inserted_ids)}")
```

Вставлено документов: 3

## 5. Просмотр добавленных документов:

```
18]: # 5. Чтение данных (Read)
print("\nСодержимое коллекции:")
for doc in collection.find():
    print(doc)
```

Содержимое коллекции:  
{'\_id': ObjectId('68c5735098fb8b3ad97594fc'), 'lab\_name': 'Lab 1', 'subject': 'Physics', 'score': 85}  
{'\_id': ObjectId('68c5735098fb8b3ad97594fd'), 'lab\_name': 'Lab 2', 'subject': 'Chemistry', 'score': 90}  
{'\_id': ObjectId('68c5735098fb8b3ad97594fe'), 'lab\_name': 'Lab 3', 'subject': 'Biology', 'score': 88}

## 6. Обновление данных:

```
[8]: # 6. Обновление данных (Update)
collection.update_one({"subject": "Physics"}, {"$set": {"score": 95}})
print("\nДокумент после обновления:")
print(collection.find_one({"subject": "Physics"}))
```

Документ после обновления:  
{'\_id': ObjectId('68c6b5d52d2221ff95451ef8'), 'lab\_name': 'Lab 1', 'subject': 'Physics', 'score': 95}

## 7. Удаление данных:

```
22]: # 7. Удаление данных (Delete)
collection.delete_one({"subject": "Chemistry"})
print(f"\nКоличество документов после удаления: {collection.count_documents({}})")
```

Количество документов после удаления: 2

## 8. Закрытие подключения:

```
: # 9. Закрытие подключения
if 'client' in locals() and client:
    client.close()
print("\nПодключение к MongoDB закрыто.")
```

Подключение к MongoDB закрыто.

Выводы: в ходе выполнения лабораторной работы были изучены основные команды для работы с тремя различными типами NoSQL баз данных. В MongoDB были освоены: создание баз данных и коллекций, поиск, подсчет и сортировка документов, селекторы, обновление документов и пр. В Neo4j удалось ознакомиться с созданием узлов и отношений, научиться выполнять запросы и строить графы. В Redis были рассмотрены команды для работы со

списками, множествами, упорядоченными множествами и многие другие. Кроме того, была изучена работа с MongoDB через Python в Jupyter notebook. В результате были получены следующие практические навыки: подключение к различным NoSQL базам данных, опыт написания запросов и понимание принципов работы с данными в каждой из них.