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

Институт цифрового образования
Департамент информатики, управления и технологий

Вариант 13

Лабораторная работа 2.1
Тема: «Изучение методов хранения данных на основе NoSQL»
Дисциплина «Инструменты для хранения и обработки больших данных»

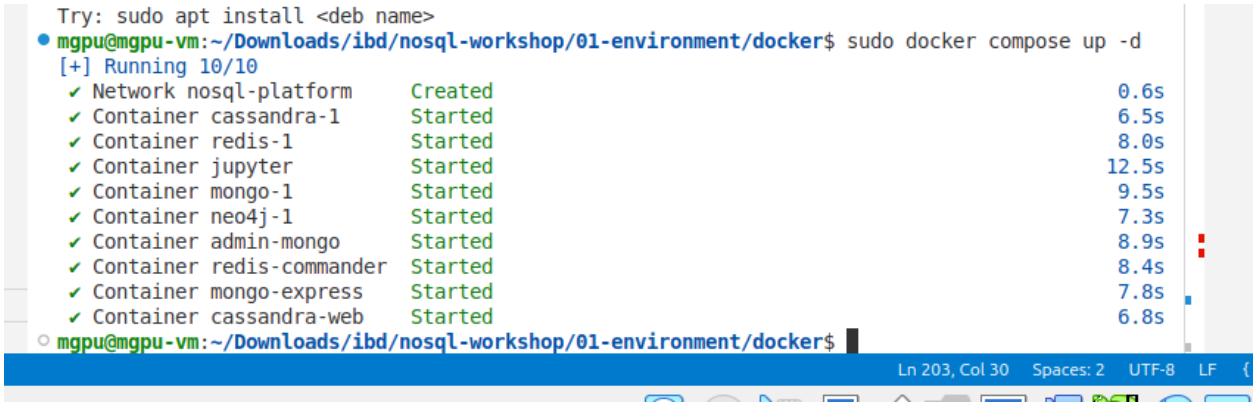
Выполнила:
Студентка группы АДЭУ-221
Селиверстова Светлана Николаевна

Преподаватель:
Тимур Муртазович Босенко
доцент, к.т.н.

Москва
2025

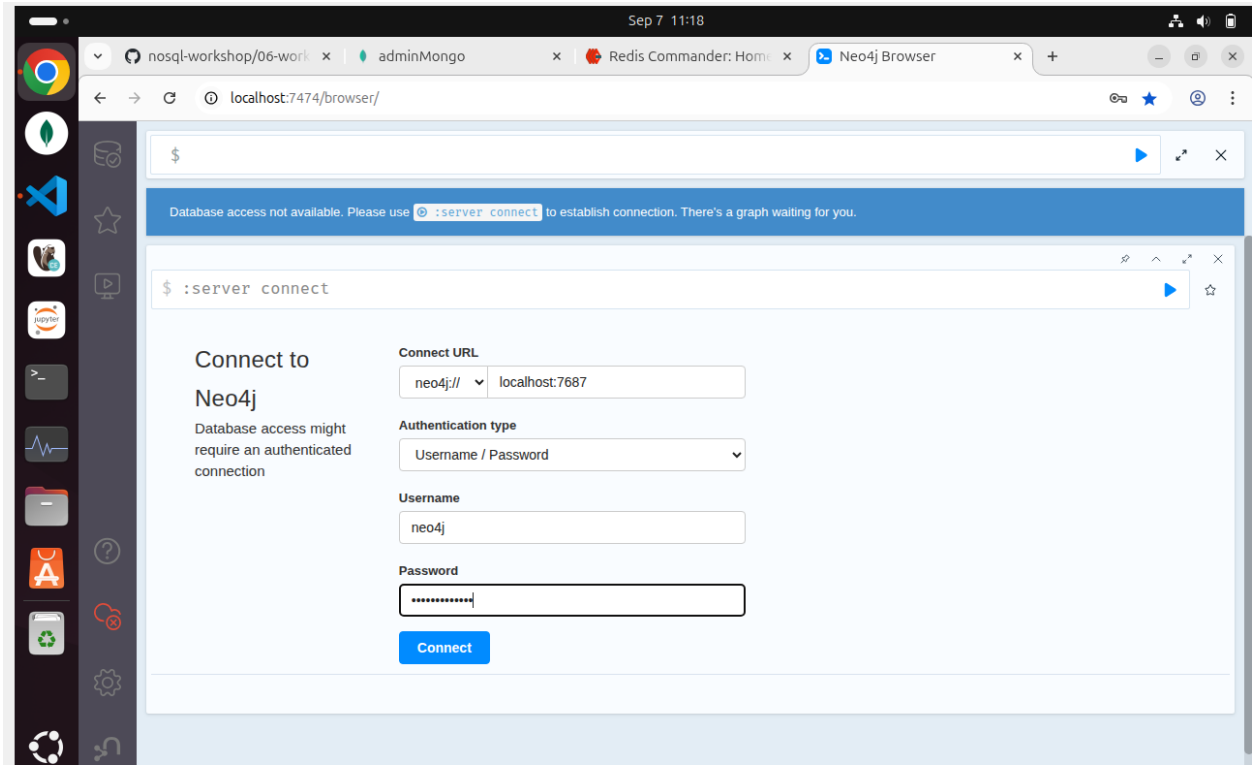
Цель работы: освоить и применить на практике три вида NoSQL-баз данных: документо-ориентированную (MongoDB), графовую (Neo4j) и хранящую данные по принципу «ключ-значение» (Redis). В рамках работы необходимо освоить создание, наполнение и анализ структур данных в каждой из этих СУБД, а также формировать запросы для извлечения нужных сведений, развивая навыки работы с нереляционными моделями данных.

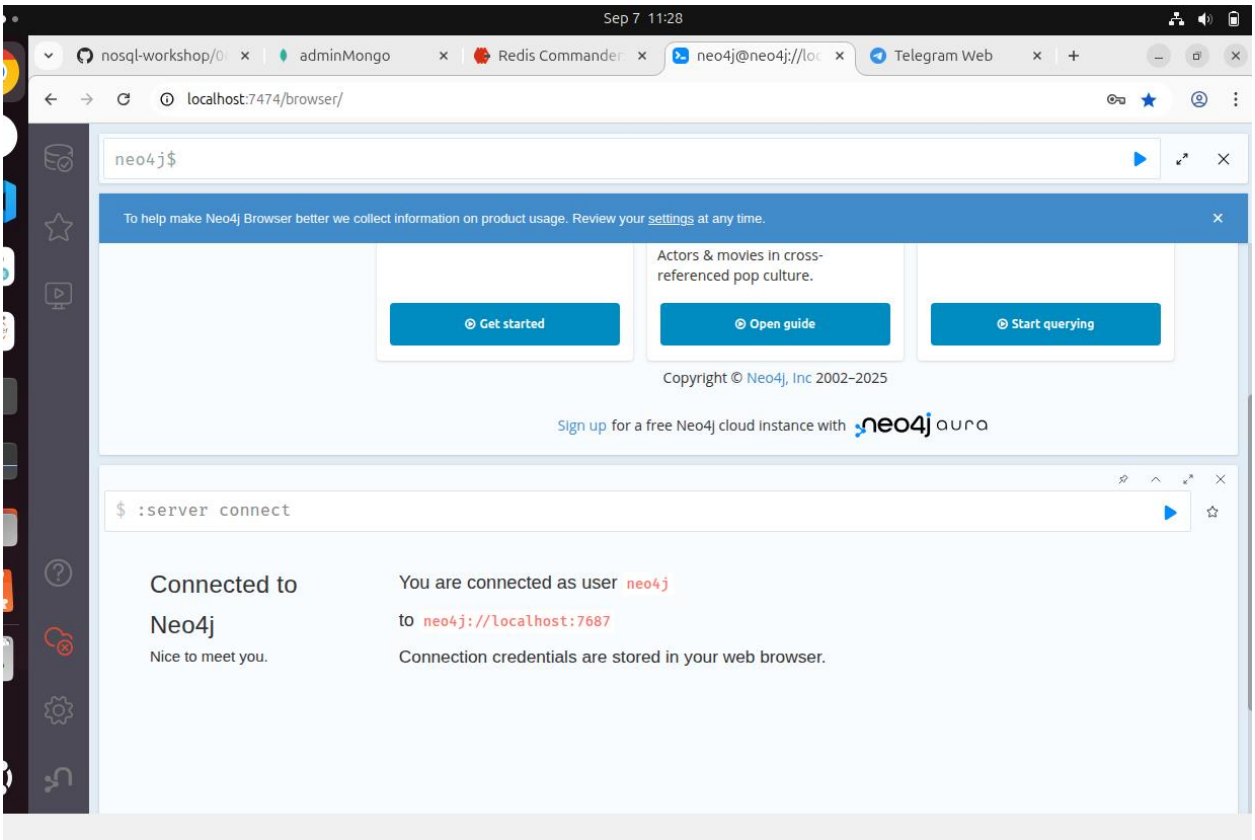
Для того, чтобы начать работу со всеми 3-мя СУБД, сначала активирую все контейнеры докер:



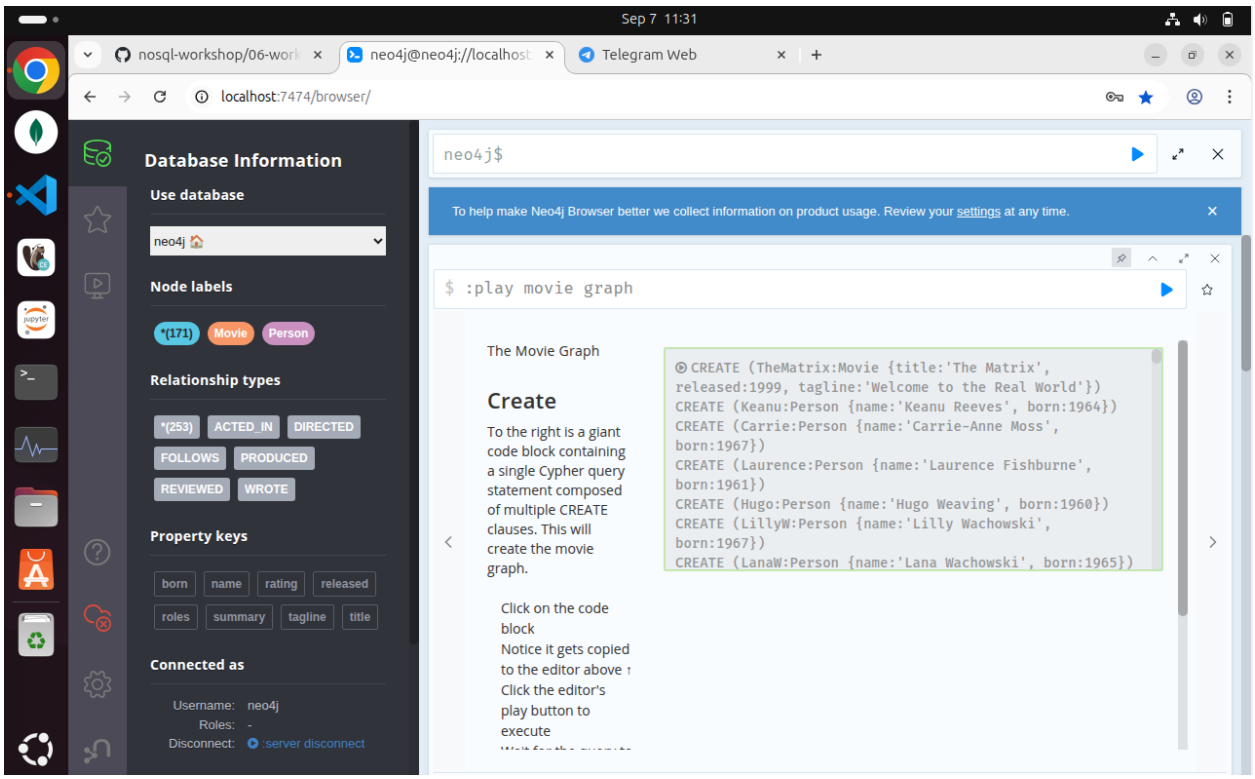
РАБОТА С NEO4J

Для начала работы с Neo4J в браузере перехожу к http://localhost:7474/ и авторизуюсь:





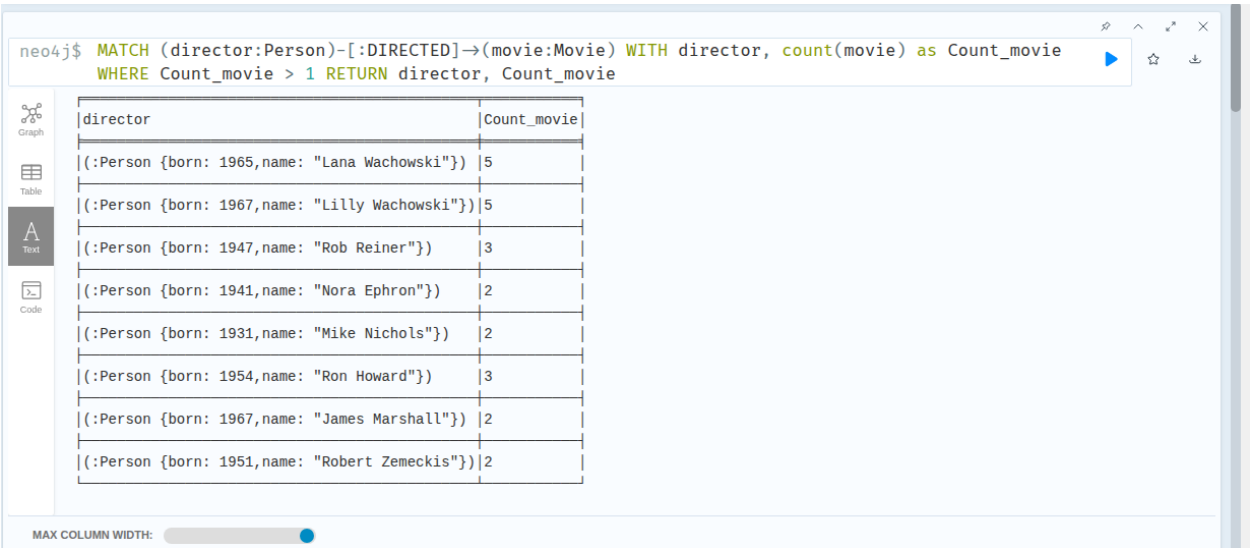
Далее перехожу к графу фильмов, создаю его:

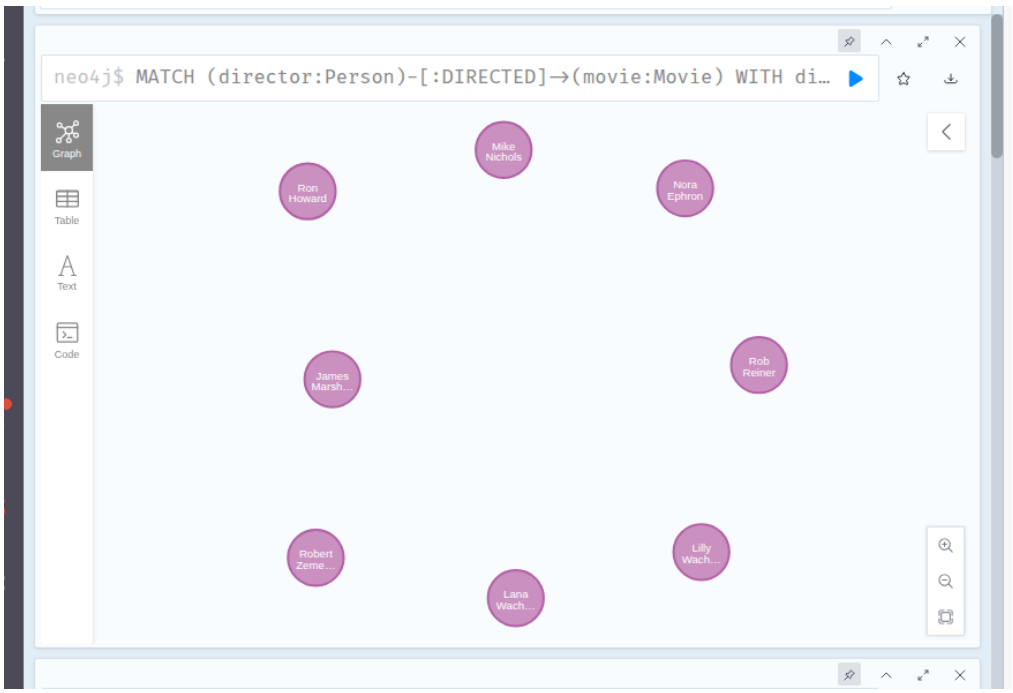


Индивидуальное задание:

Задача: найти всех режиссеров, снявших более одного фильма в базе данных.

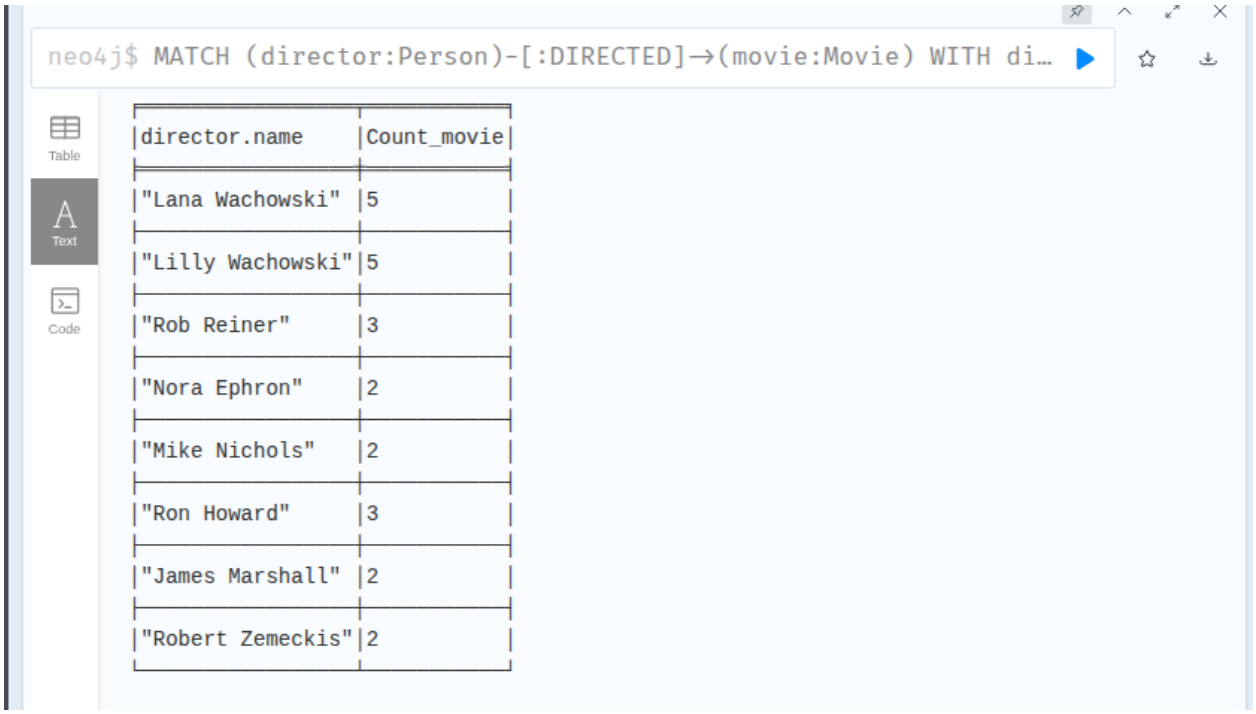
Решение: MATCH (director:Person)-[:DIRECTED]->(movie:Movie) WITH director, count(movie) as Count_movie WHERE Count_movie > 1 RETURN director, Count_movie



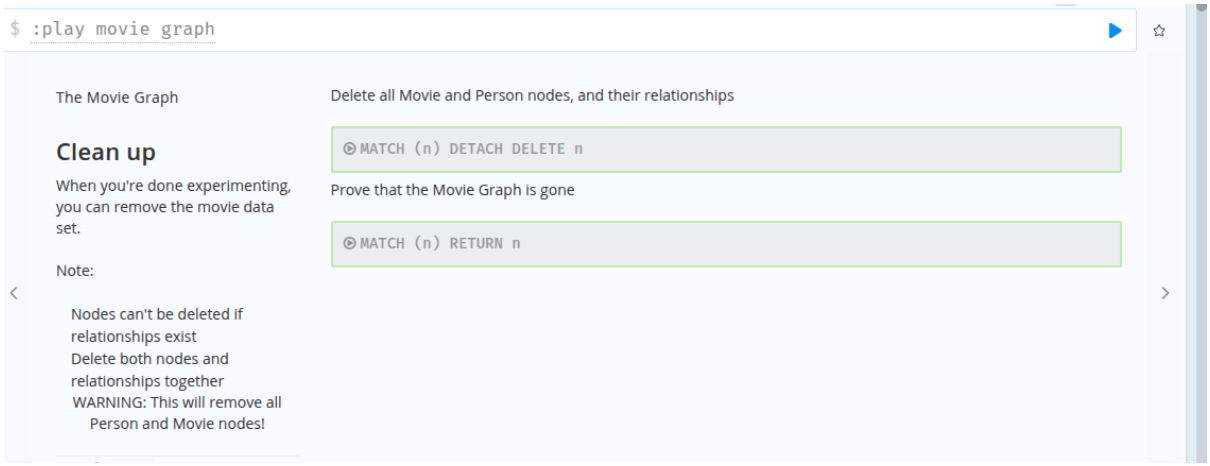


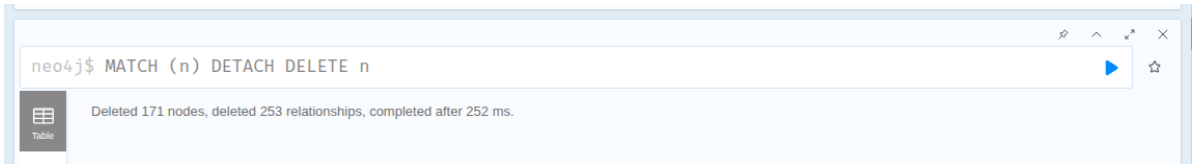
Теперь выведу только имена режиссеров:

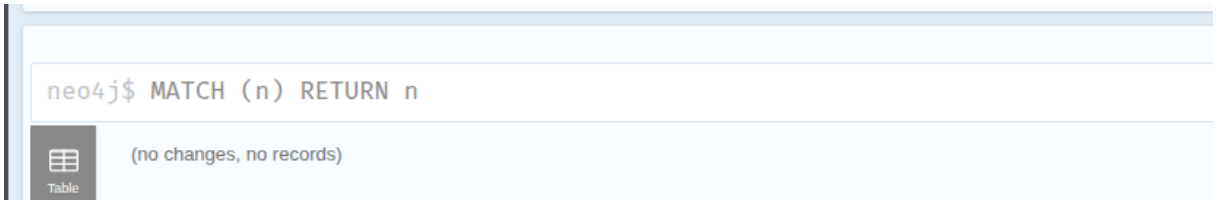
MATCH (director:Person)-[:DIRECTED]->(movie:Movie) WITH director, count(movie) as Count_movie WHERE Count_movie > 1 RETURN director.name, Count_movie



После завершения работы очищаю базу данных:


$:play movie graph
The Movie Graph
Clean up
When you're done experimenting, you can remove the movie data set.
Note:
Nodes can't be deleted if relationships exist
Delete both nodes and relationships together
WARNING: This will remove all Person and Movie nodes!
Delete all Movie and Person nodes, and their relationships
MATCH (n) DETACH DELETE n
Prove that the Movie Graph is gone
MATCH (n) RETURN n

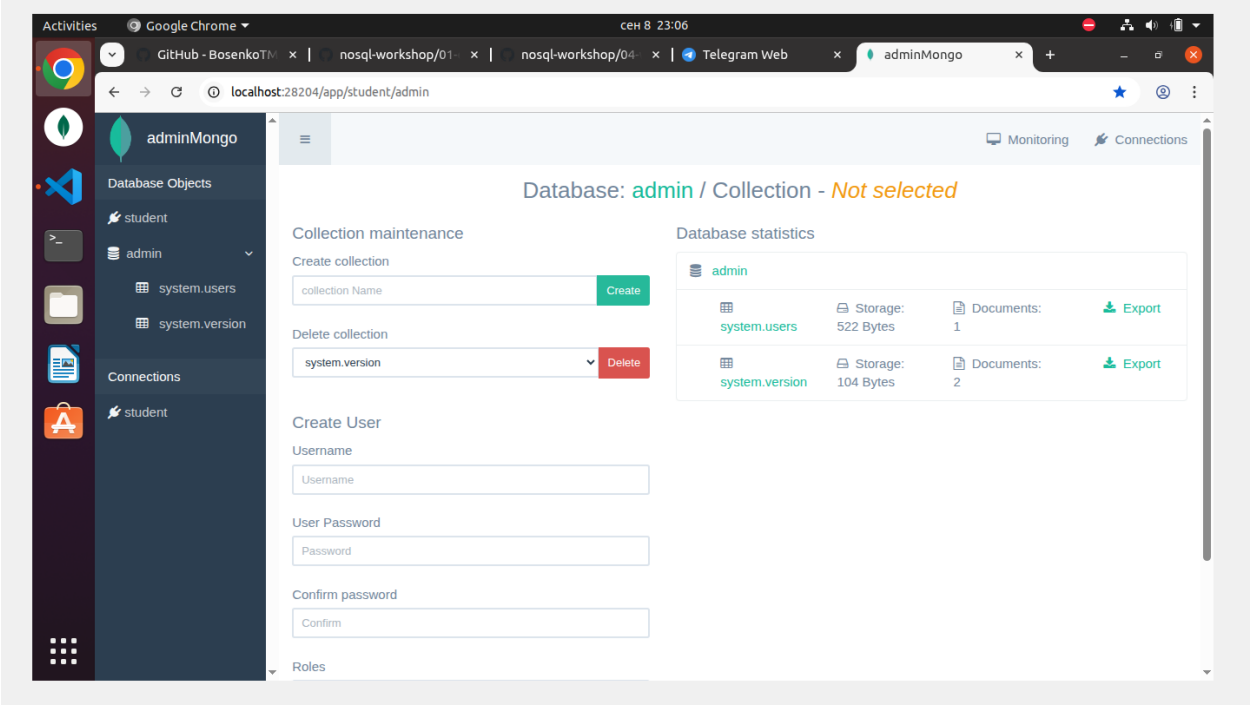

neo4j$ MATCH (n) DETACH DELETE n
Table
Deleted 171 nodes, deleted 253 relationships, completed after 252 ms.


neo4j$ MATCH (n) RETURN n
Table
(no changes, no records)

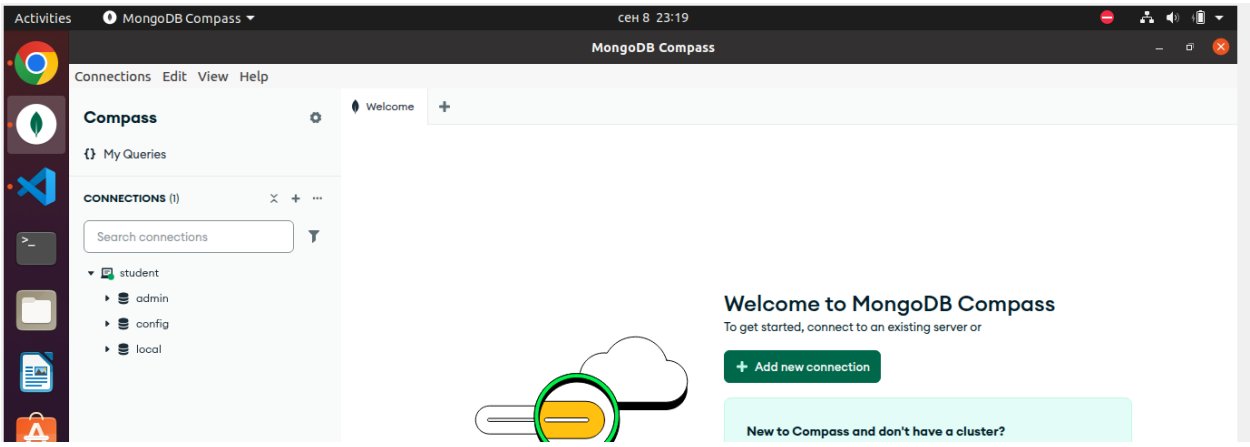
РАБОТА С MONGODB

Для начала работы с MongoDB перехожу к MongoDB Shell:

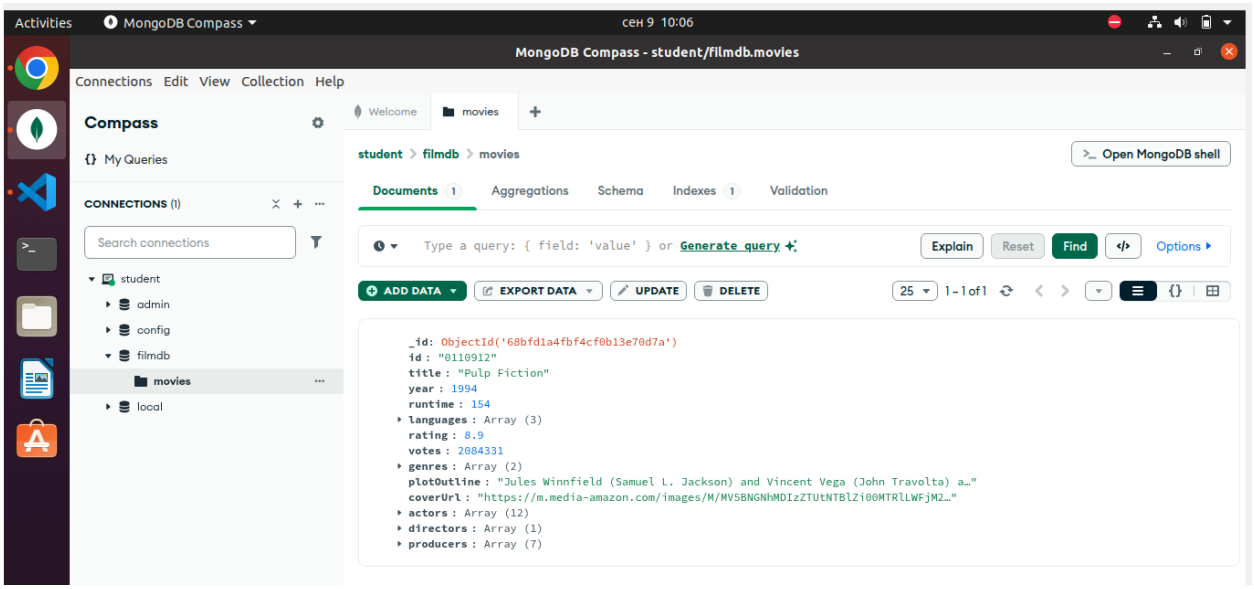
В браузере проверяю подключение:



Для дальнейшей работы перехожу в MongoDB Compass:



Создаю базу данных filmdb и добавляю в нее данные:




```
filmdb > 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 progra
```

```
> db.persons.insertOne (
  {
      "id": 0000246,
      "name": "Bruce Willis",
      "headshot": "https://m.media-amazon.com/images/M/MV5BMjA0MjMzMTE5OF5BMl5BanBnXkFtZTcwMzQ2ODE3Mw@@._V1_UY98_CR8,0,67,
      "birthDate": "1955-03-19",
      "tradeMarks": ['Frequently plays a man who suffered a tragedy, had lost something or had a  crisis of confidence or
```

```
> db.persons.insertOne (
  {
      "id": 0000206,
      "name": "Keanu Reeves",
      "headshot": "https://m.media-amazon.com/images/M/MV5BMjA0MjMzMTE5OF5BMl5BanBnXkFtZTcwMzQ2ODE3Mw@@._V1_UY98_CR8,0,67,9
      "birthDate": "1955-03-19",
      "tradeMarks": ['Intense contemplative gaze',
        'Deep husky voice',
        'Known for playing stoic reserved characters'],
      "actedInMovies": [
          { "movieId": "0133093", "title": "The Matrix"},
          { "movieId": "0234215", "title": "The Matrix Reloaded"},
```

```
> db.persons.insertOne (
 {
     "id": 0000113,
     "name": "Sandra Bullock",
     "headshot": "https://m.media-amazon.com/images/M/MV5BMTI5NDY5NjU3NF5BMl5BanBnXkFtZTcwMzQ0MTMyMw@@._V1_UX67_CR0,0,67,9
     "birthDate": "1964-07-26",
     "actedInMovies": [
         { "movieId": "2737304", "title": "Bird Box"},
         { "movieId": "0120179", "title": "Speed 2: Cruise Control"},
         { "movieId": "0111257", "title": "Speed"},
         { "movieId": "0212346", "title": "Miss Congeniality"}
     ]
```

```
> db.persons.insertOne (
 {
     "id": 0000233,
     "name": "Quentin Tarantino",
     "headshot": "https://m.media-amazon.com/images/M/MV5BMTgyMjI3ODA3Nl5BMl5BanBnXkFtZTcwNzY2MDYxOQ@@._V1_UX67_CR0,0,67,9
     "birthDate": "1963-03-27",
     "tradeMarks": ['Lead characters usually drive General Motors vehicles, particularly Chevrolet and Cadillac, such as J
           'Briefcases and suitcases play an important role in Pulp Fiction (1994), Reservoir Dogs (1992), Jackie Brown (1
           'Makes references to cult movies and television',
```

```
> db.persons.find()
< {
    _id: ObjectId('68c860eb95c8687398729b79'),
    id: 166,
    name: 'Bruce Willis',
    headshot: 'https://m.media-amazon.com/images/M/MV5BMjA0MjMzMTE5OF5BMl5BanBnXkFtZTcwMzQ2ODE3Mw@@._V1_UY98_CR8,0,67,98_AL
    birthDate: '1955-03-19',
    tradeMarks: [
```

```
> 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":
   {"id": "0468569", "title": "The Dark Knight", "genres": ["Action", "Crime", "Drama", "Thriller"], "year": 2008, "rating'
   {"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
   {"id": "0167260", "title": "The Lord of the Rings: The Return of the King", "genres": ["Adventure", "Drama", "Fantasy"]
   {"id": "0060196", "title": "The Good, the Bad and the Ugly", "genres": ["Western"], "year": 1966, "rating": 8.8, "rank":
```

```
> db.movies.deleteOne( { "title": "Fight Club" } )
< {
    acknowledged: true,
    deletedCount: 1
  }
filmdb >
```

Программное взаимодействие с базами данных (Python). Работа с MongoDB через pymongo

Подключаюсь к MongoDB:

[21]:
```python
try:
    client = MongoClient("mongodb://root:abc123!@mongo-1:27017/")
    db = client.student
    collection = db.test_labs

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

✅ Успешное подключение к MongoDB!

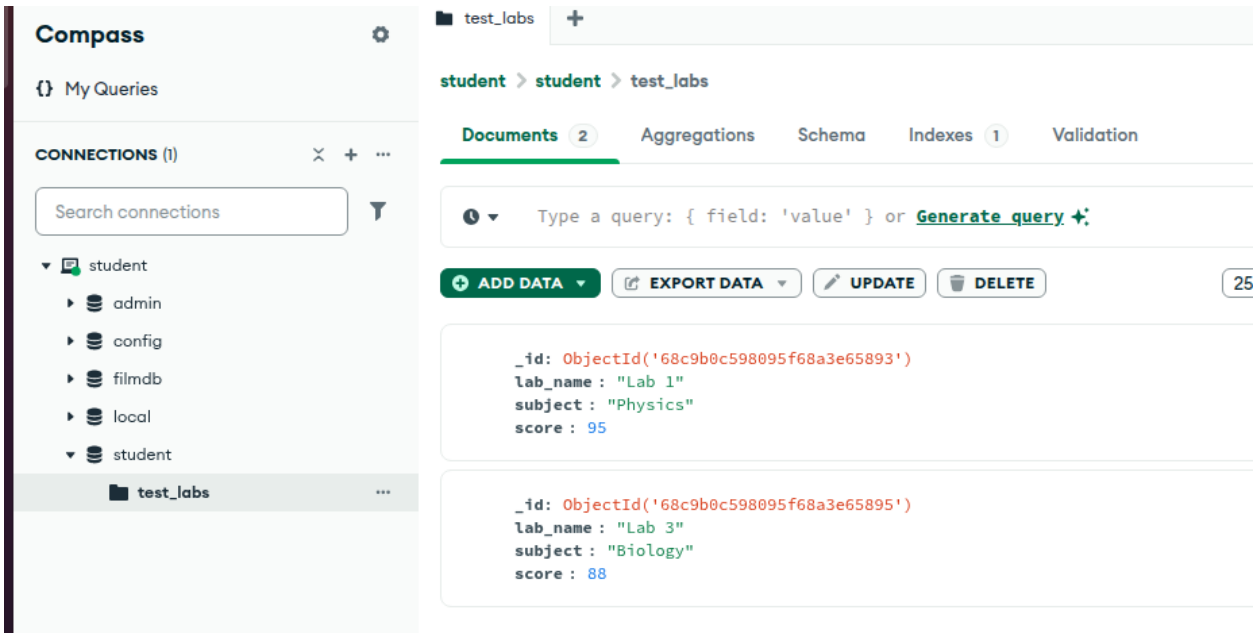
Пробую взаимодействовать с этой СУБД, манипулируя с данные:

```python
try:
    # 3. Выбор базы данных и коллекции
   #db = client[student]
   #collection = db[test_labs]
# Очистка коллекции перед началом работы
   #collection.delete_many({})
# 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)}")
# 5. Чтение данных (Read)
   print("\nСодержимое коллекции:")
   for doc in collection.find():
       print(doc)
# 6. Обновление данных (Update)
   collection.update_one({"subject": "Physics"},  {"$set": {"score": 95}})
   print("\nДокумент после обновления:")
   print(collection.find_one({"subject": "Physics"}))
# 7. Удаление данных (Delete)
   collection.delete_one({"subject": "Chemistry"})
   print(f"\nКоличество документов удаления:{collection.count_documents({})}")
# 8. Удаление коллекции для очистки
   db.drop_collection(collection_name)
```

Результат:

```
Вставлено документов: 3

Содержимое коллекции:
{'_id': ObjectId('68c9b0c598095f68a3e65893'), 'lab_name': 'Lab 1', 'subject': 'Physics', 'score': 85}
{'_id': ObjectId('68c9b0c598095f68a3e65894'), 'lab_name': 'Lab 2', 'subject': 'Chemistry', 'score': 90}
{'_id': ObjectId('68c9b0c598095f68a3e65895'), 'lab_name': 'Lab 3', 'subject': 'Biology', 'score': 88}

Документ после обновления:
{'_id': ObjectId('68c9b0c598095f68a3e65893'), 'lab_name': 'Lab 1', 'subject': 'Physics', 'score': 95}

Количество документов удаления:2
Ошибка: name 'collection_name' is not defined

Подключение к MongoDB закрыто.
```

Проверяю созданную таблицу в MongoDB Compass:



Индивидуальное задание:

Задание: Найти все фильмы, выпущенные в XXI веке (год >= 2001), с рейтингом ниже 8.5.

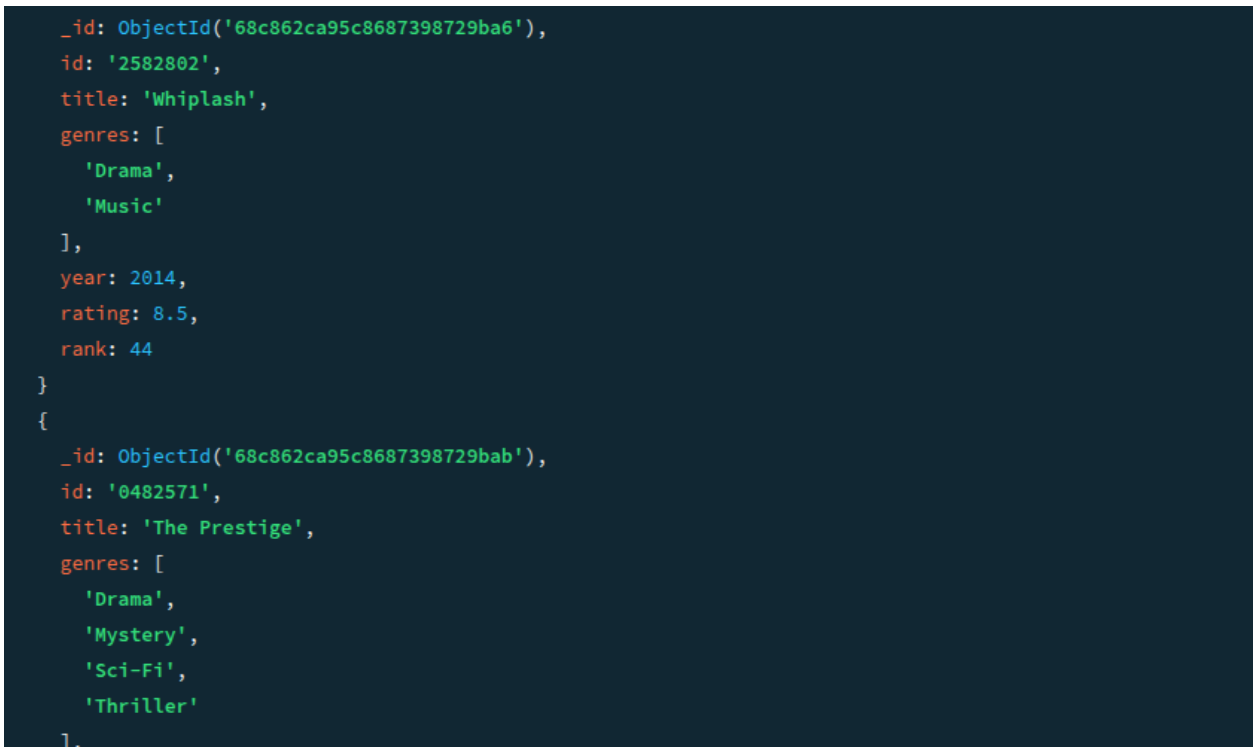
Решение (первый вариант, если условие строго меньше): db.movies.find({ "year": { $gte: 2001 }, "rating": { $lt: 8.5 }})

```
}
> db.movies.find({ "year": { $gte :  2001 } , "rating": { $lt : 8.5 } })
<
filmdb >
```

Как видно, таких фильмов в базе данных нет.

Решение (второй вариант, если условие меньше или равно): db.movies.find({ "year": { $gte: 2001 }, "rating": { $lte: 8.5 }})

```
db.movies.find({ "year": { $gte :  2001 } , "rating": { $lte : 8.5 } })
{
  _id: ObjectId('68c862ca95c8687398729b95'),
  id: '0245429',
  title: 'Spirited Away',
  genres: [
    'Animation',
    'Adventure',
    'Family',
    'Fantasy',
    'Mystery'
  ],
  year: 2001,
  rating: 8.5,
  rank: 27
}
{
  _id: ObjectId('68c862ca95c8687398729b9a'),
  id: '0816692',
  title: 'Interstellar',
  genres: [
    'Adventure',
    'Drama',
    'Sci-Fi'
  ],
  year: 2002,
  rating: 8.5,
  rank: 38
}
{
  _id: ObjectId('68c862ca95c8687398729ba2'),
  id: '1675434',
  title: 'The Intouchables',
  genres: [
    'Biography',
    'Comedy',
    'Drama'
  ],
  year: 2011,
  rating: 8.5,
  rank: 40
}
{
  _id: ObjectId('68c862ca95c8687398729ba3'),
  id: '0407887',
  title: 'The Departed',
```



Здесь уже можно увидеть, что несколько таких фильмов присутствуют в базе данных.

НАЧАЛО РАБОТЫ С REDIS

Для начала работы с Redis перехожу к контейнеру:

```
mgpu@mgpu-vm:~/Downloads/idb/nosql-workshop/01-environment/docker$ docker run -it --rm --network
osql-platform bitnami/redis redis-cli -h redis-1 -p 6379
Unable to find image 'bitnami/redis:latest' locally
latest: Pulling from bitnami/redis
ce4f4d45406a: Pull complete
Digest: sha256:25bf63f3caf75af4628c0dfcf39859ad1ac8abe135be85e99699f9637b16dc28
Status: Downloaded newer image for bitnami/redis:latest
redis 20:30:23.16 INFO  ==>
redis 20:30:23.16 INFO  ==> Welcome to the Bitnami redis container
redis 20:30:23.17 INFO  ==> Subscribe to project updates by watching https://github.com/bitn
ntainers
redis 20:30:23.17 INFO  ==> NOTICE: Starting August 28th, 2025, only a limited subset of ima
arts will remain available for free. Backup will be available for some time at the 'Bitnami
' repository. More info at https://github.com/bitnami/containers/issues/83267
redis 20:30:23.19 INFO  ==>

redis-1:6379> AUTH abc123!
```

Проверяю подключение:

```
redis-1:6379> PING
PONG
redis-1:6379>
```

Индивидуальное задание:

Задание: В упорядоченное множество ranking:posts добавить 5 постов с их рейтингами. Найти все посты с рейтингом от 100 до 500 (ZRANGEBYSCORE).

Решение: ZADD ranking:posts 123 "post 1"

ZADD ranking:posts 132 "post 2"

ZADD ranking:posts 99 "post 3"

ZADD ranking:posts 400 "post 4"

ZADD ranking:posts 505 "post 5"

ZRANGEBYSCORE ranking:posts 100 500

```
ZADD ranking:posts 123 "post 1"
1
ZADD ranking:posts 132 "post 2"
1
ZADD ranking:posts 99 "post 3"
1
ZADD ranking:posts 400 "post 4"
1
ZADD ranking:posts 505 "post 5"
1
```

```
ZRANGEBYSCORE ranking:posts 100 500
1) "post 1"
2) "post 2"
3) "post 4"
```

Можно увидеть, что рейтинги постс 1, 2 и 4 действительно подходят под условия (от 100 до 500)

Вывод: В ходе выполнения работы были успешно изучены и применены на практике три NoSQL-базы данных, каждая из которых продемонстрировала свои уникальные преимущества в зависимости от решаемых задач. Neo4j оказалась идеальным решением для работы со связанными данными, а засчет графового отображения, восприятие информации упрощается. MongoDB позволила эффективно работать с полуструктурированными данными. Были освоены операции создания и обновления документов, запросы с использованием операторов сравнения ($gte, $lt), а также агрегация для анализа данных. Redis продемонстрировал высокую производительность для операций с кэшированием и управления отсортированными множествами.