Федышина София АДЭУ-221

Отчет

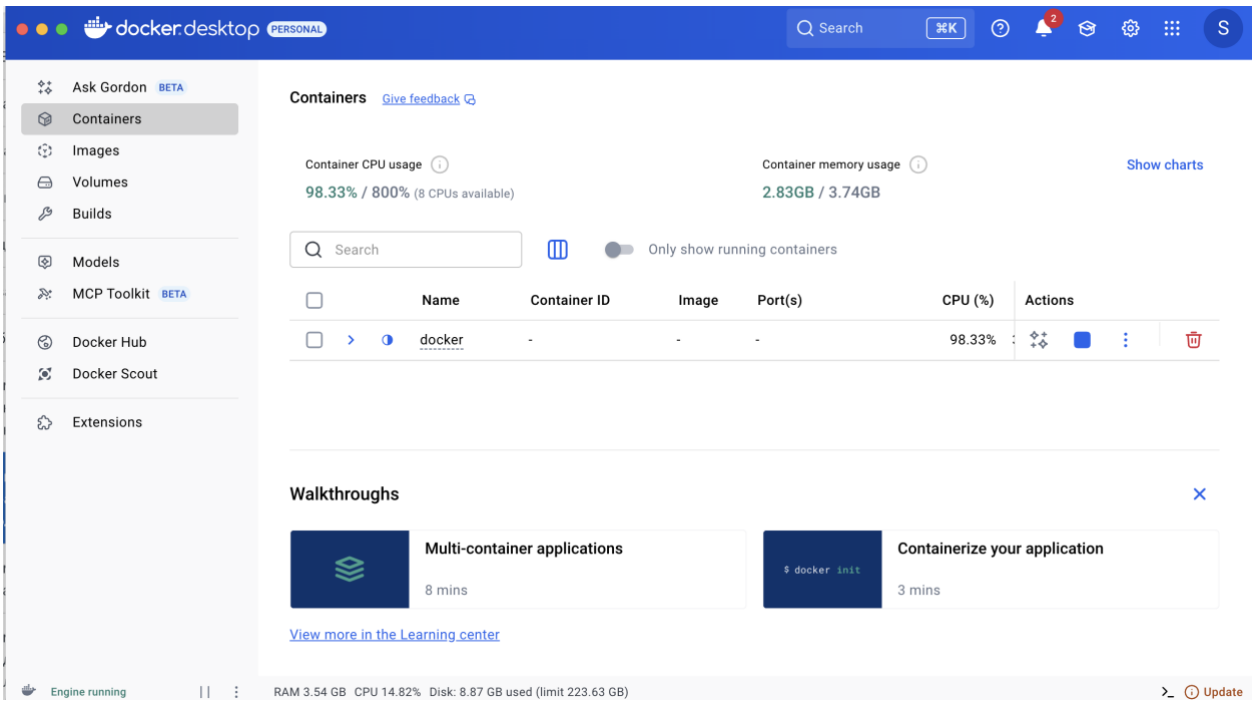
Лабораторная работа 2.1. Изучение методов хранения данных на основе NoSQL

Цель работы: освоить принципы хранения и обработки данных в различных типах NoSQL систем — документо-ориентированных (MongoDB), графовых (Neo4j) и ключ-значение (Redis). В ходе выполнения работы необходимо научиться создавать и наполнять базы данных, выполнять запросы и анализировать результаты, что позволит закрепить навыки работы с нереляционными моделями данных и понять их особенности.

Описание процесса выполнения

Установка

Установила докер и проверила успешность его установки через терминал




```
(base) sofia@MacBook-Air-2 ~ % docker --version
Docker version 28.3.3, build 980b856
(base) sofia@MacBook-Air-2 ~ % docker compose version
Docker Compose version v2.39.2-desktop.1
```

Клонировала данный в задании репозиторий

```
(base) sofia@MacBook-Air-2 ~ % cd ~
(base) sofia@MacBook-Air-2 ~ % git clone https://github.com/BosenkoTM/nosql-workshop.git
Cloning into 'nosql-workshop'...
remote: Enumerating objects: 2483, done.
remote: Counting objects: 100% (225/225), done.
remote: Compressing objects: 100% (200/200), done.
remote: Total 2483 (delta 105), reused 5 (delta 5), pack-reused 2258 (from 3)
Receiving objects: 100% (2483/2483), 177.45 MiB | 7.21 MiB/s, done.
Resolving deltas: 100% (420/420), done.
Updating files: 100% (1407/1407), done.
(base) sofia@MacBook-Air-2 ~ %
```

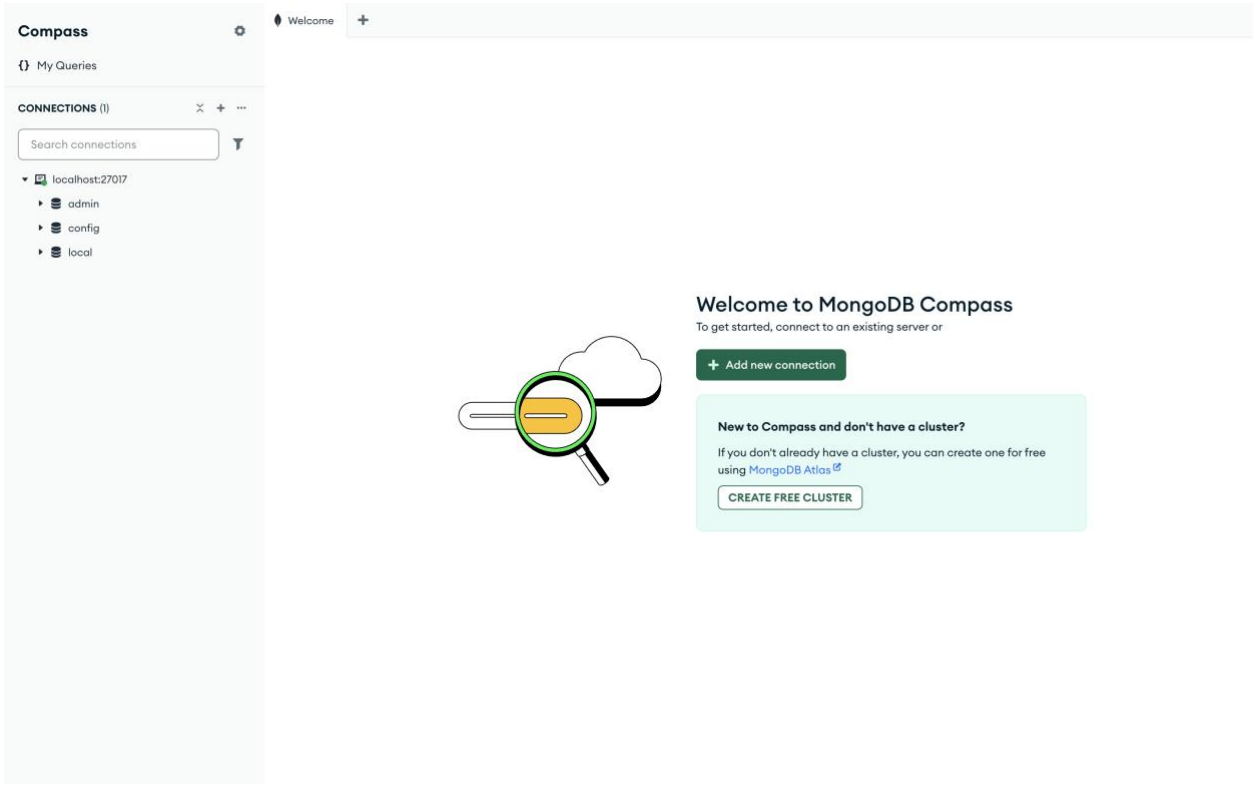
Перешла в нужный каталог, подняла контейнеры и проверила

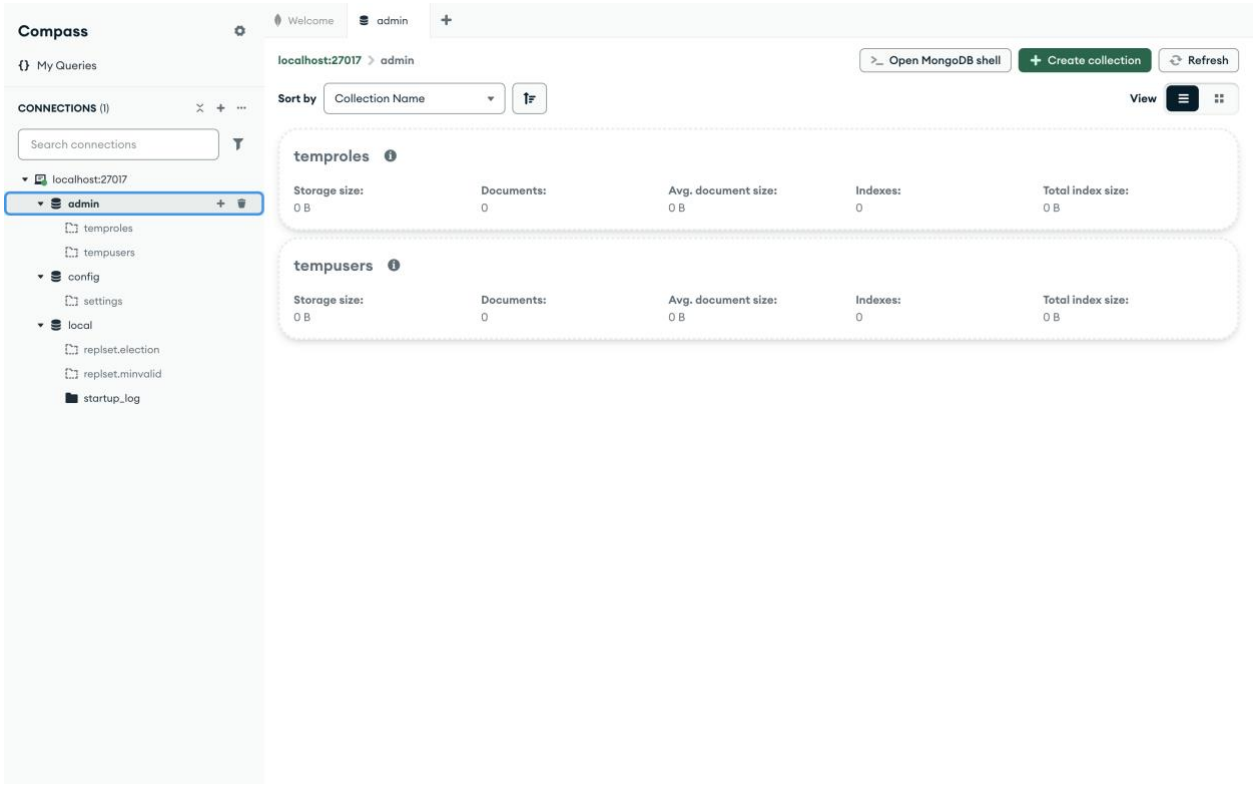
```
(base) sofia@MacBook-Air-2 ~ % cd ~/nosql-workshop/01-environment/docker
(base) sofia@MacBook-Air-2 docker % docker compose up -d
[+] Running 75/75
 ✔ neo4j-1 Pulled                                                    211.4s
 ✔ cassandra-1 Pulled                                                154.4s
 ✔ redis-1 Pulled                                                     71.6s
 ✔ mongo-express Pulled                                              133.6s
 ✔ mongo-1 Pulled                                                    139.5s
 ✔ admin-mongo Pulled                                                155.8s
 ✔ redis-commander Pulled                                             20.1s
 ✔ cassandra-web Pulled                                              106.7s
 ✔ jupyter Pulled                                                    266.4s
```

```
[+] Running 10/10
 ✔ Network nosql-platform      Created                                 0.0s
 ✔ Container redis-commander   Started                                 1.0s
 ✔ Container admin-mongo       Started                                 1.4s
 ✔ Container mongo-express     Started                                 1.1s
 ✔ Container cassandra-web     Started                                 1.1s
 ✔ Container mongo-1           Started                                 1.2s
 ✔ Container redis-1           Started                                 1.3s
 ✔ Container cassandra-1       Started                                 1.6s
 ✔ Container jupyter           Started                                 1.0s
 ✔ Container neo4j-1           Started                                 1.0s
(base) sofia@MacBook-Air-2 docker %
```

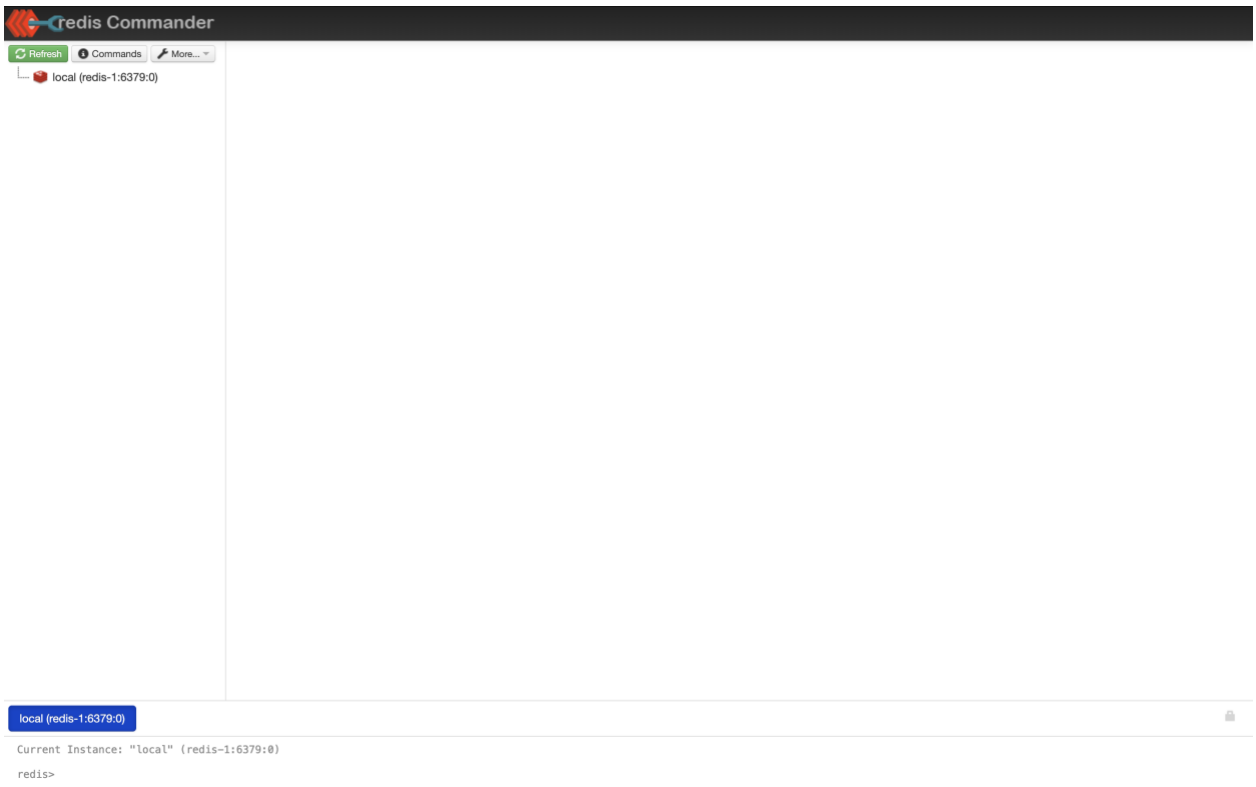
```
(base) sofia@MacBook-Air-2 docker % docker ps
CONTAINER ID   IMAGE                                    COMMAND                  CREATED         STATUS                            PORTS
      NAMES
73fc1eb1a19c   jupyter/minimal-notebook:latest          "tini -g -- bash -c …"   7 minutes ago   Up 7 minutes (healthy)            0.0.0.0:14040->4040/tcp, [::]:14040->4040/tcp, 0.0.0.0:14041->4041/tcp, [:
:]:14041->4041/tcp, 0.0.0.0:14042->4042/tcp, [::]:14042->4042/tcp, 0.0.0.0:14043->4043/tcp, [::]:14043->4043/tcp, 0.0.0.0:14044->4044/tcp, [::]:14044->4044/tcp, 0.0.0.0:28888->8888/tcp, [::]:28888->8888/t
cp    jupyter
f040ab44304f   neo4j:5.19                               "tini -g -- /startup…"   7 minutes ago   Up 7 minutes (healthy)            0.0.0.0:7474->7474/tcp, [::]:7474->7474/tcp, 0.0.0.0:7687->7687/tcp, [::]:
7687->7687/tcp
      neo4j-1
9e811b39744e   mongo:7.0                                "docker-entrypoint.s…"   7 minutes ago   Up 7 minutes                      0.0.0.0:27017->27017/tcp, [::]:27017->27017/tcp

      mongo-1
32f2d48de1be   mongo-express:latest                     "/sbin/tini -- /dock…"   7 minutes ago   Up 7 minutes                      0.0.0.0:28203->8081/tcp, [::]:28203->8081/tcp

      mongo-express
4f16ce81e1df   bitnami/cassandra:4.1                    "/opt/bitnami/script…"   7 minutes ago   Up 7 minutes                      0.0.0.0:7199->7199/tcp, [::]:7199->7199/tcp, 0.0.0.0:9160->9160/tcp, [::]:
9160->9160/tcp, 0.0.0.0:29042->9042/tcp, [::]:29042->9042/tcp
      cassandra-1
50ca08d57fc4   bitnami/redis:7.2                        "/opt/bitnami/script…"   7 minutes ago   Up 7 minutes (healthy)            0.0.0.0:6379->6379/tcp, [::]:6379->6379/tcp

      redis-1
9133b6b00ae5   adicom/admin-mongo:latest                "node app.js"            7 minutes ago   Up 7 minutes                      0.0.0.0:28204->1234/tcp, [::]:28204->1234/tcp

      admin-mongo
9c2fed2abad5   rediscommander/redis-commander:latest    "/usr/bin/dumb-init …"   7 minutes ago   Up 7 minutes (unhealthy)          0.0.0.0:28119->8081/tcp, [::]:28119->8081/tcp

      redis-commander
d87669a46482   trivadis/cassandra-web                   "/entrypoint.sh"         7 minutes ago   Restarting (139) 22 seconds ago

      cassandra-web
(base) sofia@MacBook-Air-2 docker %
```

Установила MongoDB Compass и создала подключение

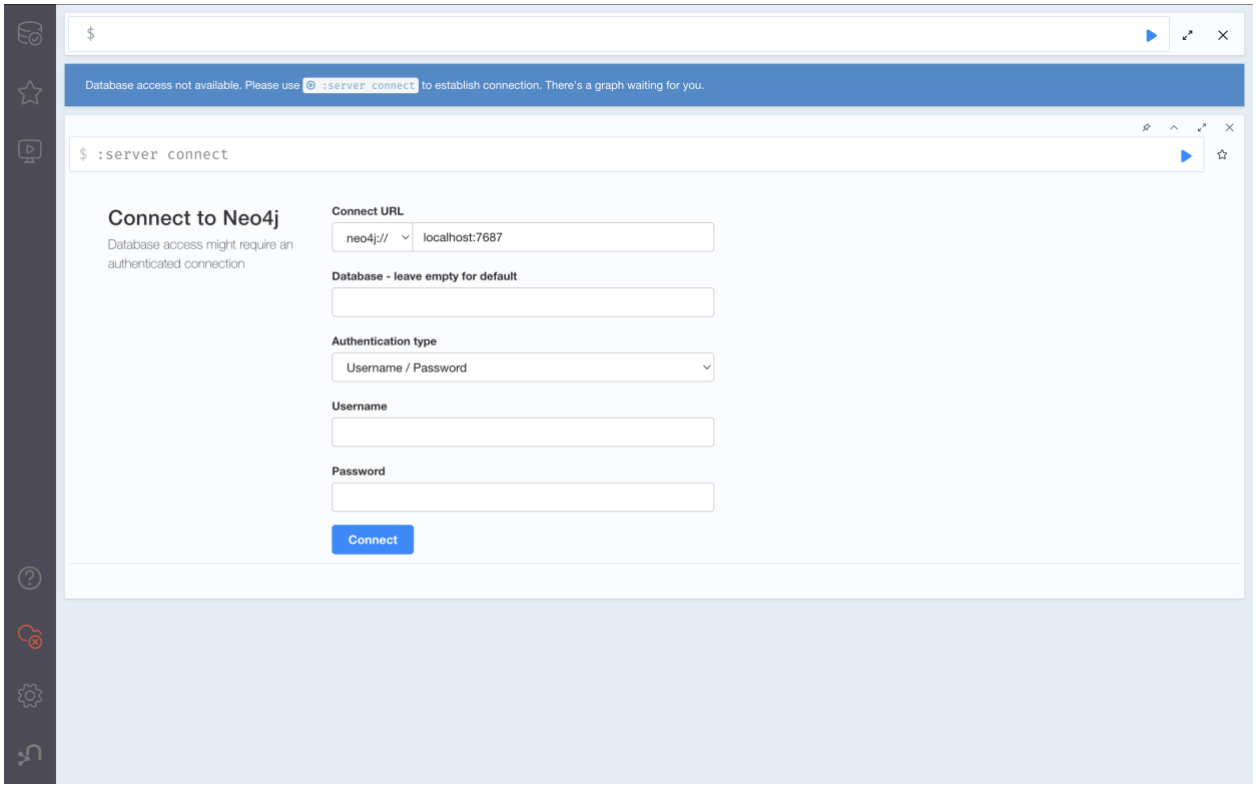


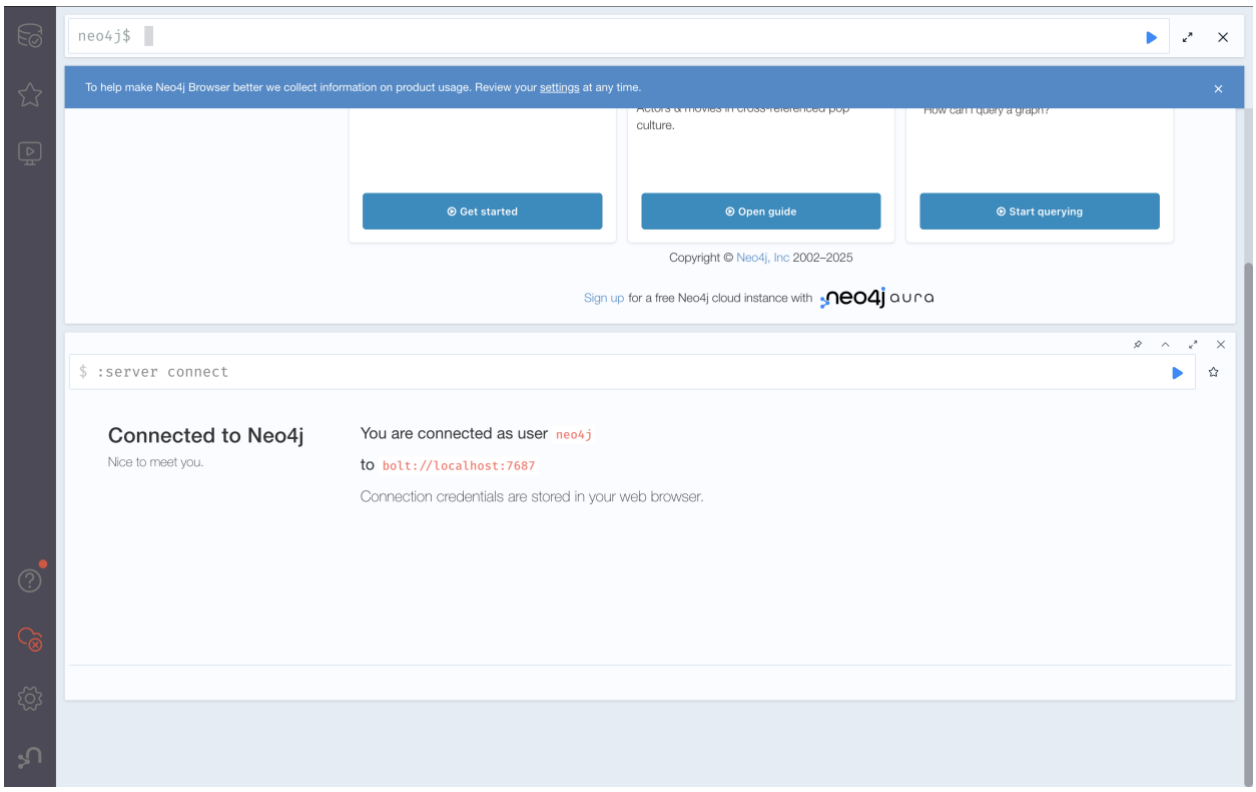


Открыла в браузере Redis Commander с помощью ссылки http://localhost:8082



Открыла в браузере Neo4j с помощью ссылки http://localhost:7474, создала соединение





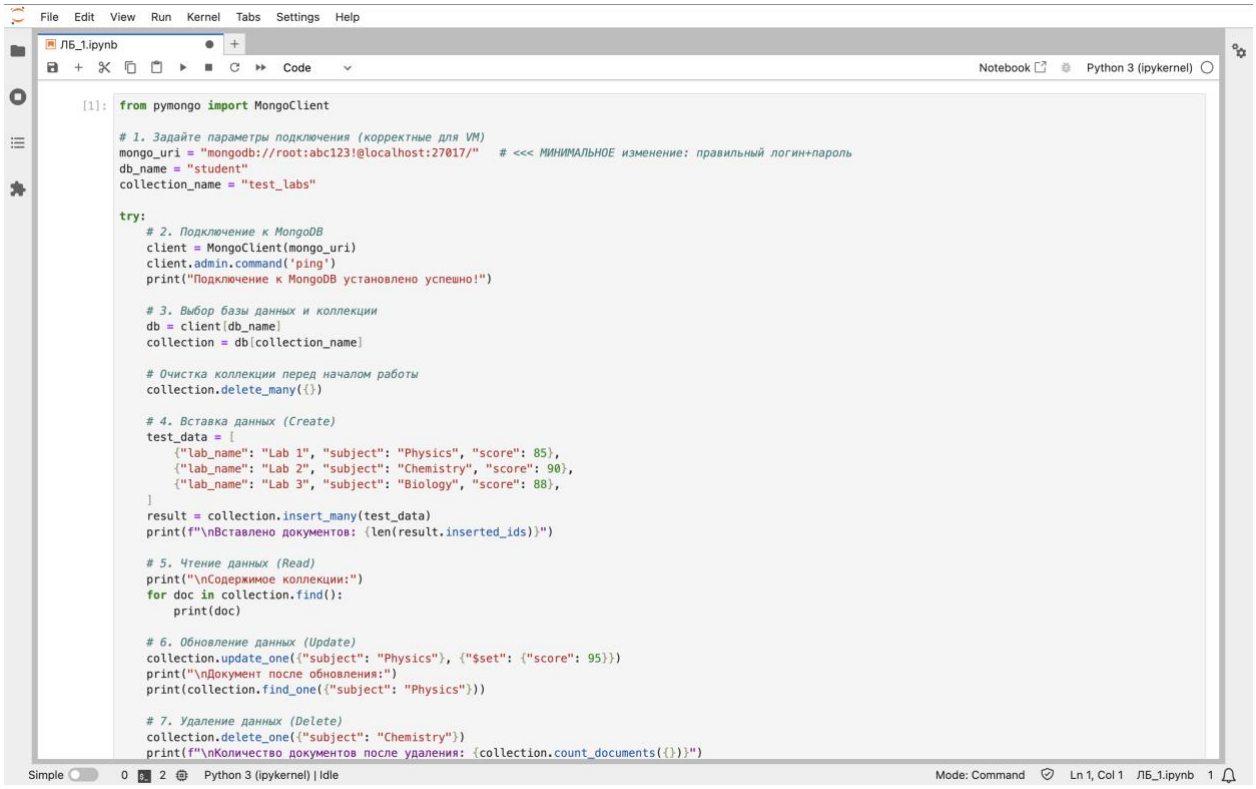
Устанавила необходимые библиотеки питон

```
(base) sofia@MacBook-Air-2 docker % conda activate nosql_lab
pip install pymongo redis neo4j


EnvironmentNameNotFound: Could not find conda environment: nosql_lab
You can list all discoverable environments with `conda info --envs`.


Collecting pymongo
  Downloading pymongo-4.14.1-cp311-cp311-macosx_11_0_arm64.whl.metadata (22 kB)
Collecting redis
  Downloading redis-6.4.0-py3-none-any.whl.metadata (10 kB)
Collecting neo4j
  Downloading neo4j-5.28.2-py3-none-any.whl.metadata (5.9 kB)
Collecting dnspython<3.0.0,>=1.16.0 (from pymongo)
  Downloading dnspython-2.7.0-py3-none-any.whl.metadata (5.8 kB)
Requirement already satisfied: pytz in /opt/anaconda3/lib/python3.11/site-packages (from neo4j) (2023.3.post1)
Downloading pymongo-4.14.1-cp311-cp311-macosx_11_0_arm64.whl (859 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 859.7/859.7 kB 3.1 MB/s eta 0:00:00
Downloading redis-6.4.0-py3-none-any.whl (279 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 279.8/279.8 kB 12.2 MB/s eta 0:00:00
Downloading neo4j-5.28.2-py3-none-any.whl (313 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 313.2/313.2 kB 12.1 MB/s eta 0:00:00
Downloading dnspython-2.7.0-py3-none-any.whl (313 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 313.6/313.6 kB 11.4 MB/s eta 0:00:00
Installing collected packages: redis, neo4j, dnspython, pymongo
Successfully installed dnspython-2.7.0 neo4j-5.28.2 pymongo-4.14.1 redis-6.4.0
(base) sofia@MacBook-Air-2 docker %
```

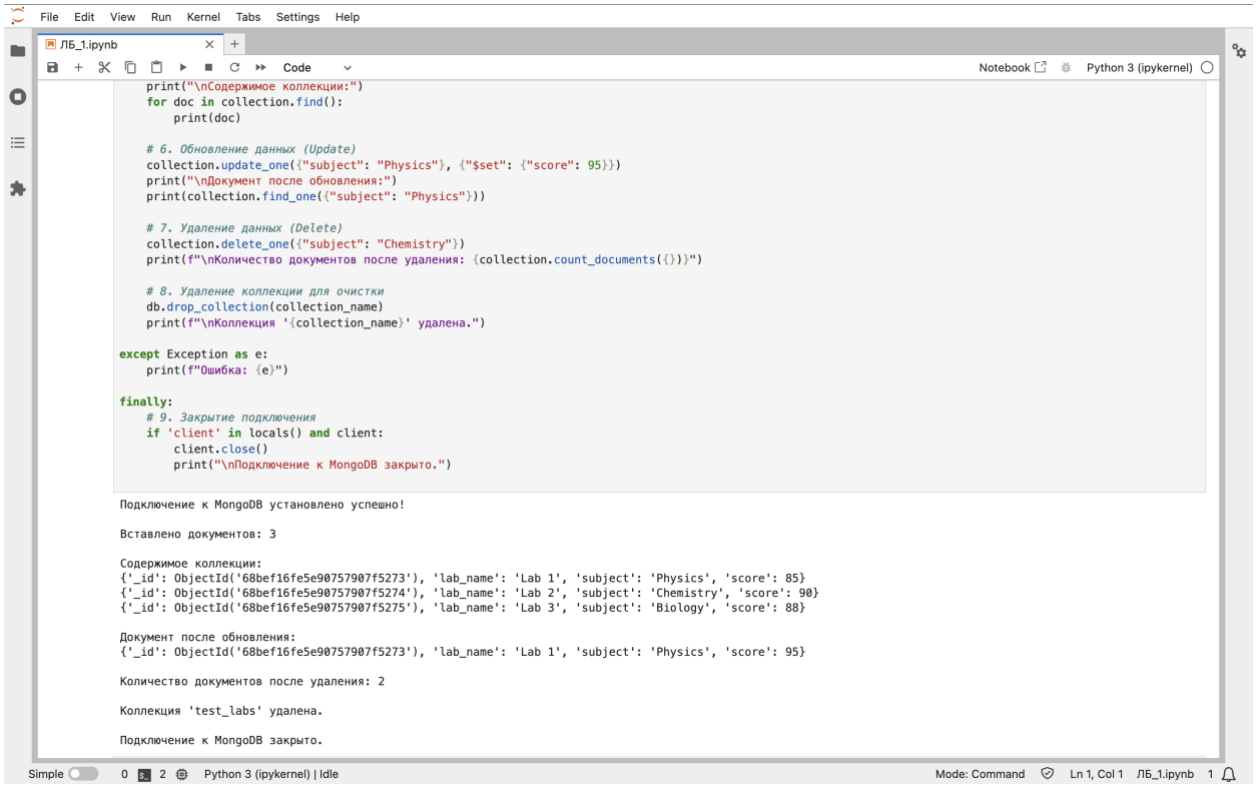
Подключение на питоне

Работа с MongoDB через pymongo

Скопировала и выполнила код, данный в задании, в ячейке Jupyter. Он демонстрирует полный цикл CRUD-операций: подключение, создание данных, чтение, обновление и удаление.



```
from pymongo import MongoClient

# 1. Задайте параметры подключения (корректные для VM)
mongo_uri = "mongodb://root:abc123!@localhost:27017/"   # <<< МИНИМАЛЬНОЕ изменение: правильный логин+пароль
db_name = "student"
collection_name = "test_labs"

try:
    # 2. Подключение к MongoDB
    client = MongoClient(mongo_uri)
    client.admin.command('ping')
    print("Подключение к MongoDB установлено успешно!")

    # 3. Выбор базы данных и коллекции
    db = client[db_name]
    collection = db[collection_name]

    # 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 после удаления: {collection.count_documents({})}")
```

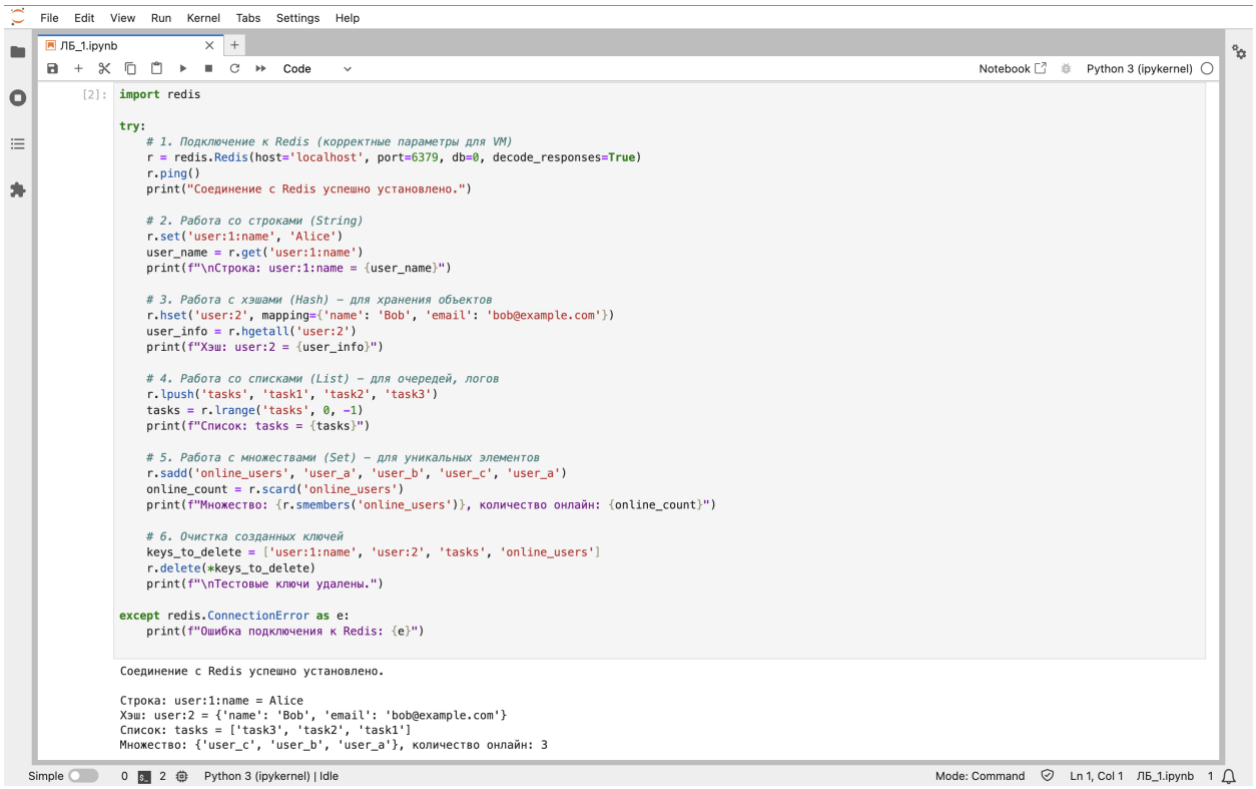


```
    print("\nСодержимое коллекции:")
    for doc in collection.find():
        print(doc)

    # 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 после удаления: {collection.count_documents({})}")

    # 8. Удаление коллекции для очистки
    db.drop_collection(collection_name)
    print(f"\nКоллекция '{collection_name}' удалена.")

except Exception as e:
    print(f"Ошибка: {e}")

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

```
Подключение к MongoDB установлено успешно!

Вставлено документов: 3

Содержимое коллекции:
{'_id': ObjectId('68bef16fe5e90757907f5273'), 'lab_name': 'Lab 1', 'subject': 'Physics', 'score': 85}
{'_id': ObjectId('68bef16fe5e90757907f5274'), 'lab_name': 'Lab 2', 'subject': 'Chemistry', 'score': 90}
{'_id': ObjectId('68bef16fe5e90757907f5275'), 'lab_name': 'Lab 3', 'subject': 'Biology', 'score': 88}

Документ после обновления:
{'_id': ObjectId('68bef16fe5e90757907f5273'), 'lab_name': 'Lab 1', 'subject': 'Physics', 'score': 95}

Количество документов после удаления: 2

Коллекция 'test_labs' удалена.

Подключение к MongoDB закрыто.
```

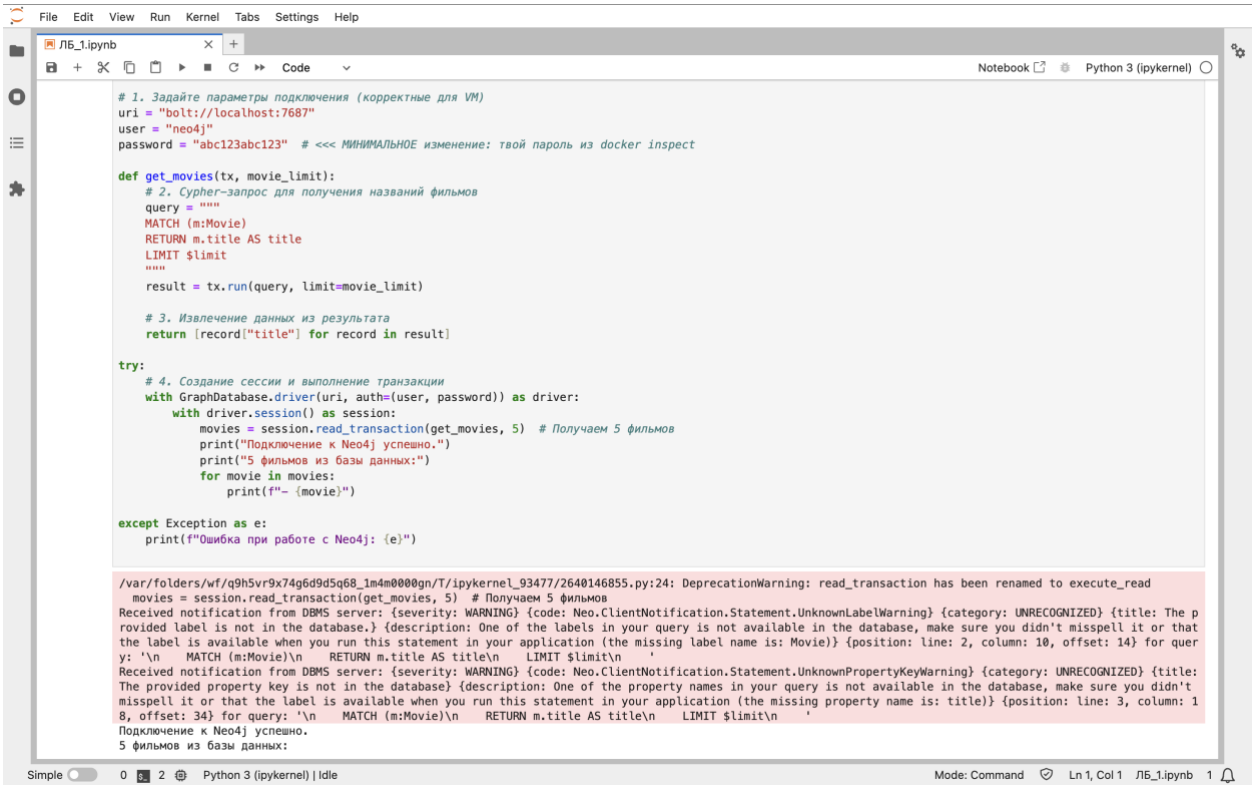
Работа с Redis через redis-py

Следующий код демонстрирует работу с основными типами данных в Redis. Выполнила его в новой ячейке Jupyter.



Работа с Neo4j через neo4j-driver

Данный код показывает, как выполнить Cypher-запрос к базе данных Neo4j и обработать результаты.



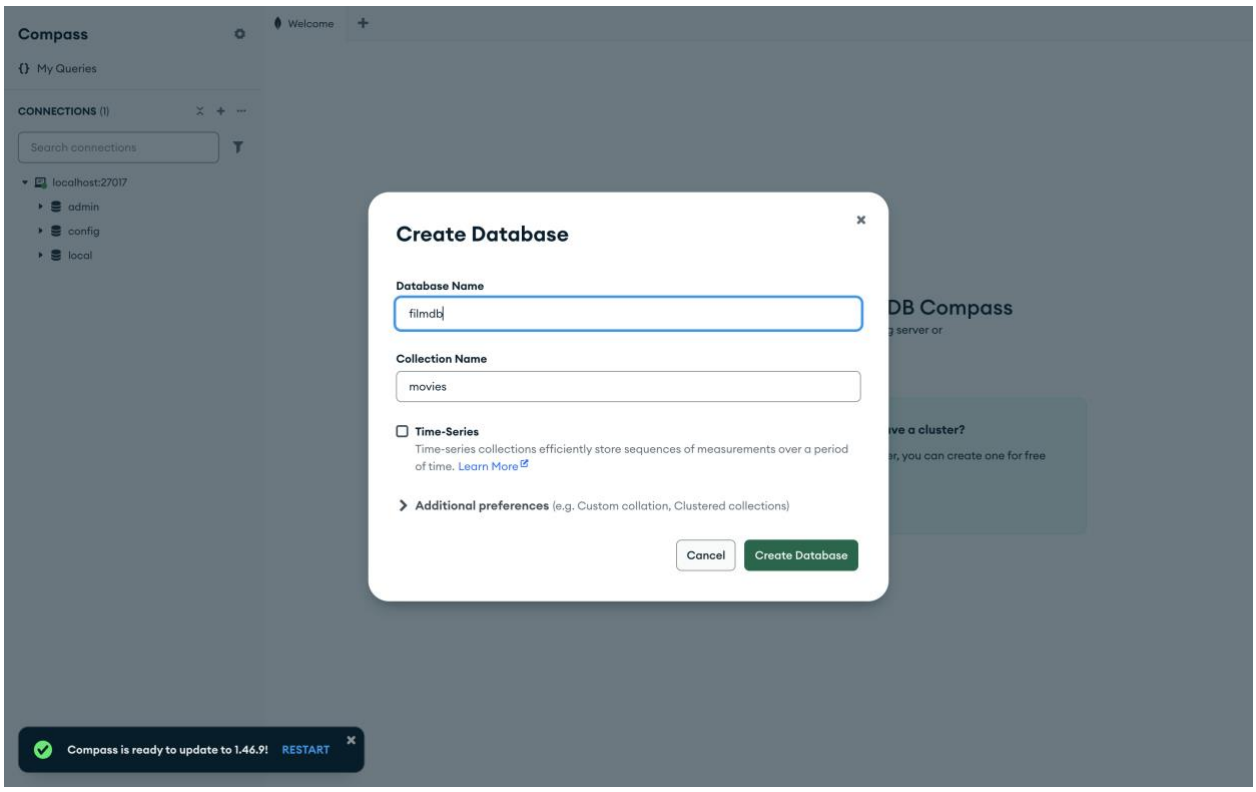
В результате код выдал предупреждение о том, что в базе Neo4j нет узлов с меткой Movie. Код выполняется, но возвращать нечего, поэтому список фильмов пустой.

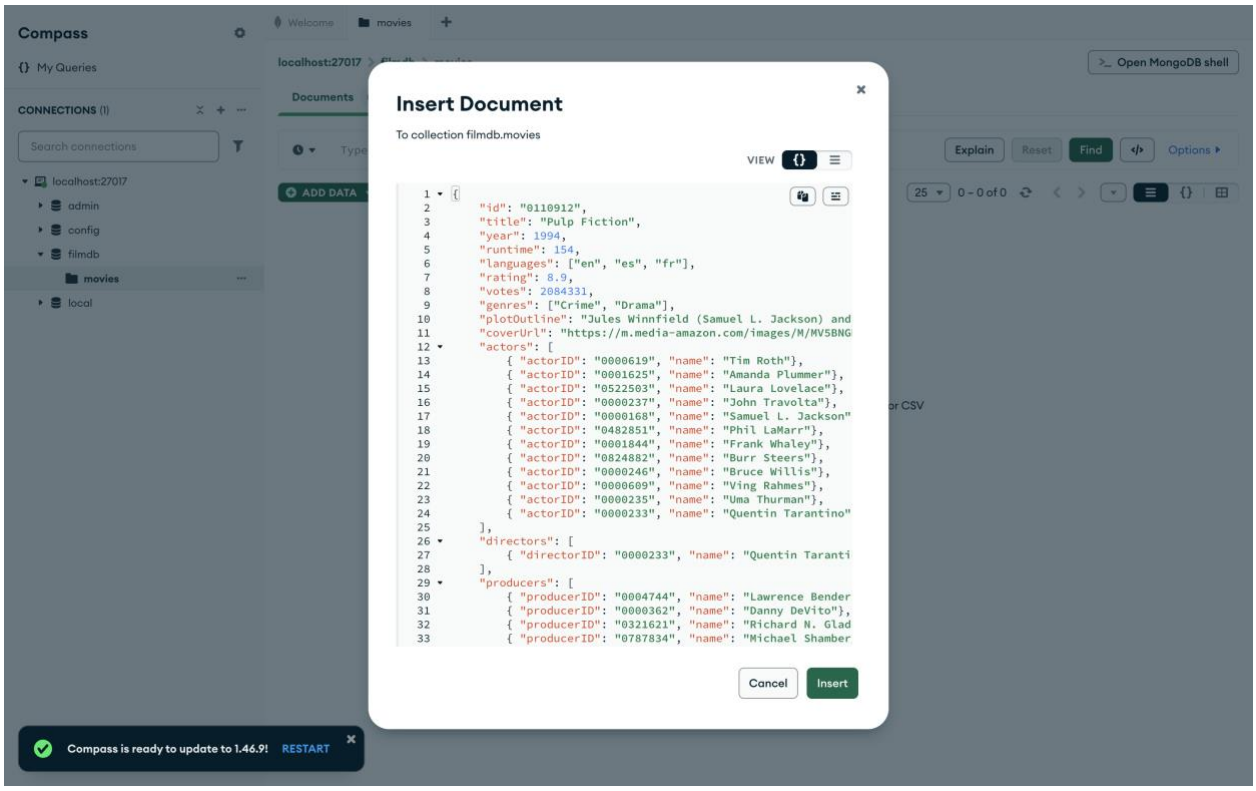
Работа с MongoDB

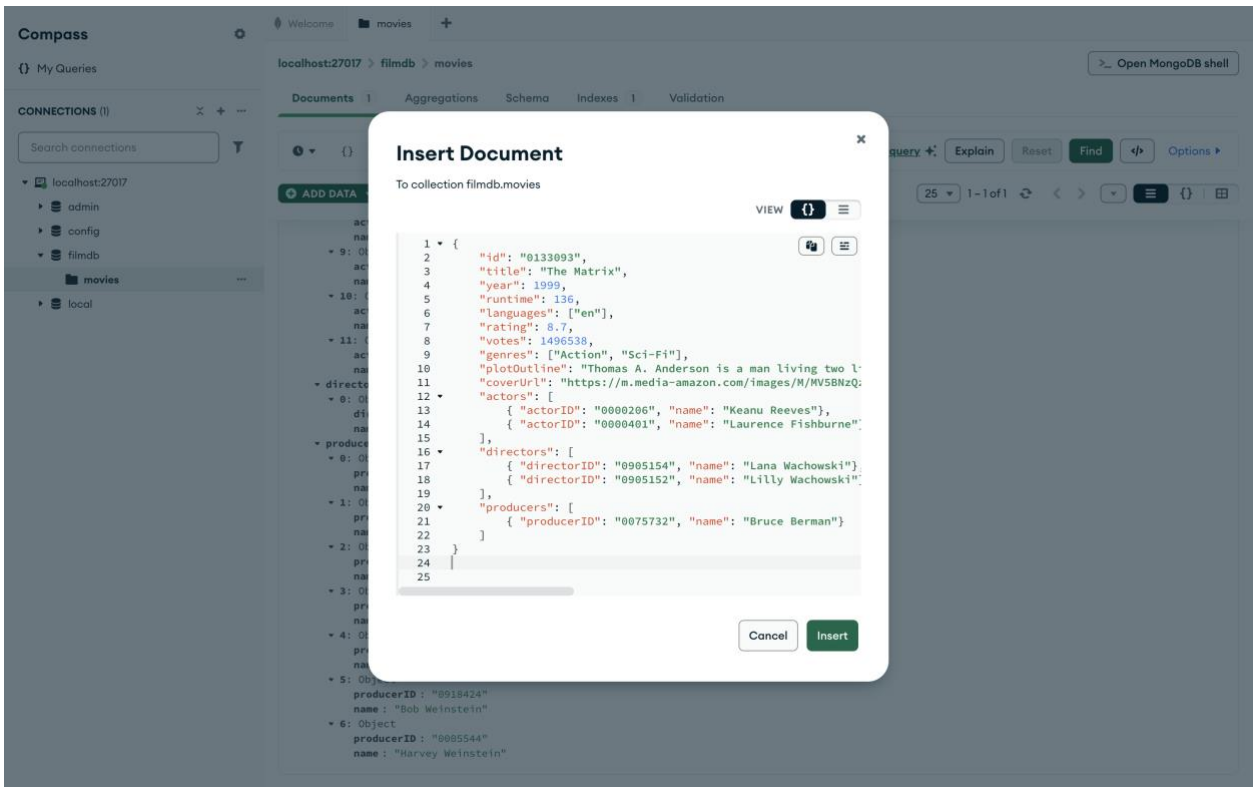
Загрузка данных

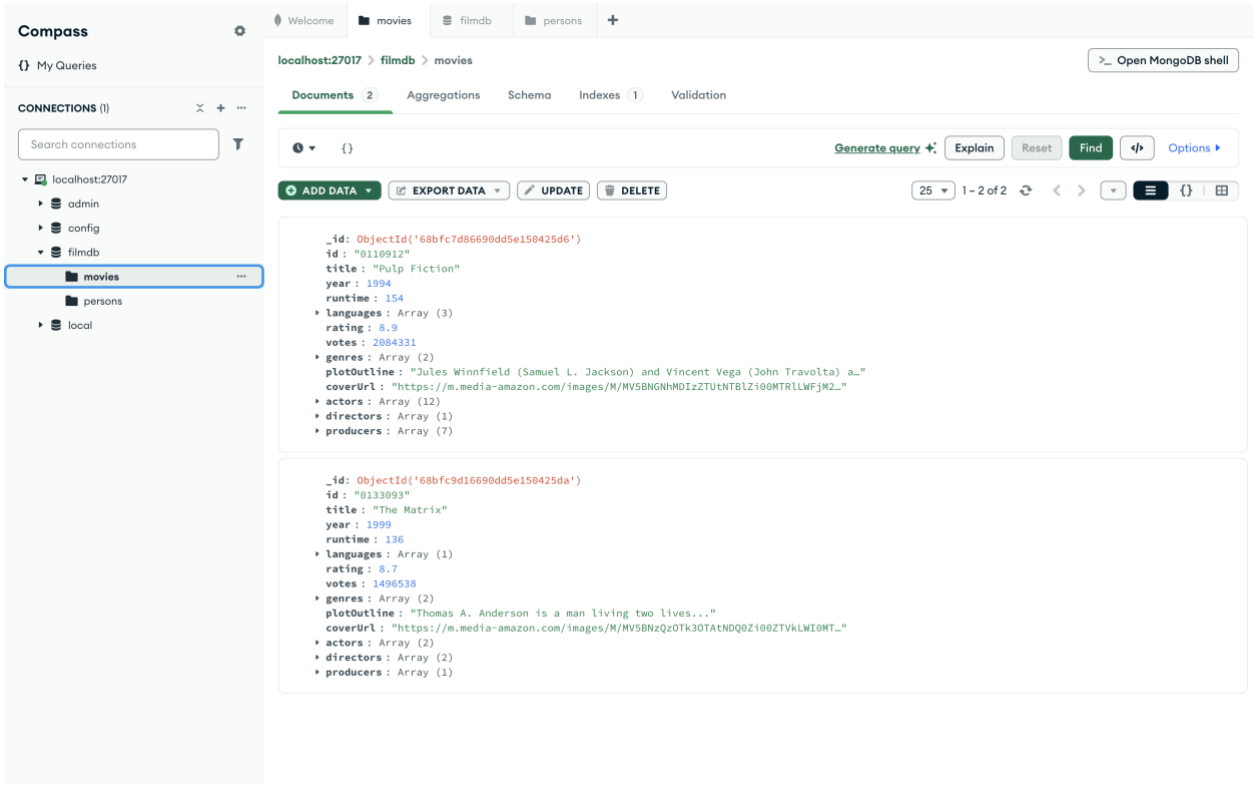
Загрузила данные в монго (взяты с гитхаба)

Коллекция фильмов

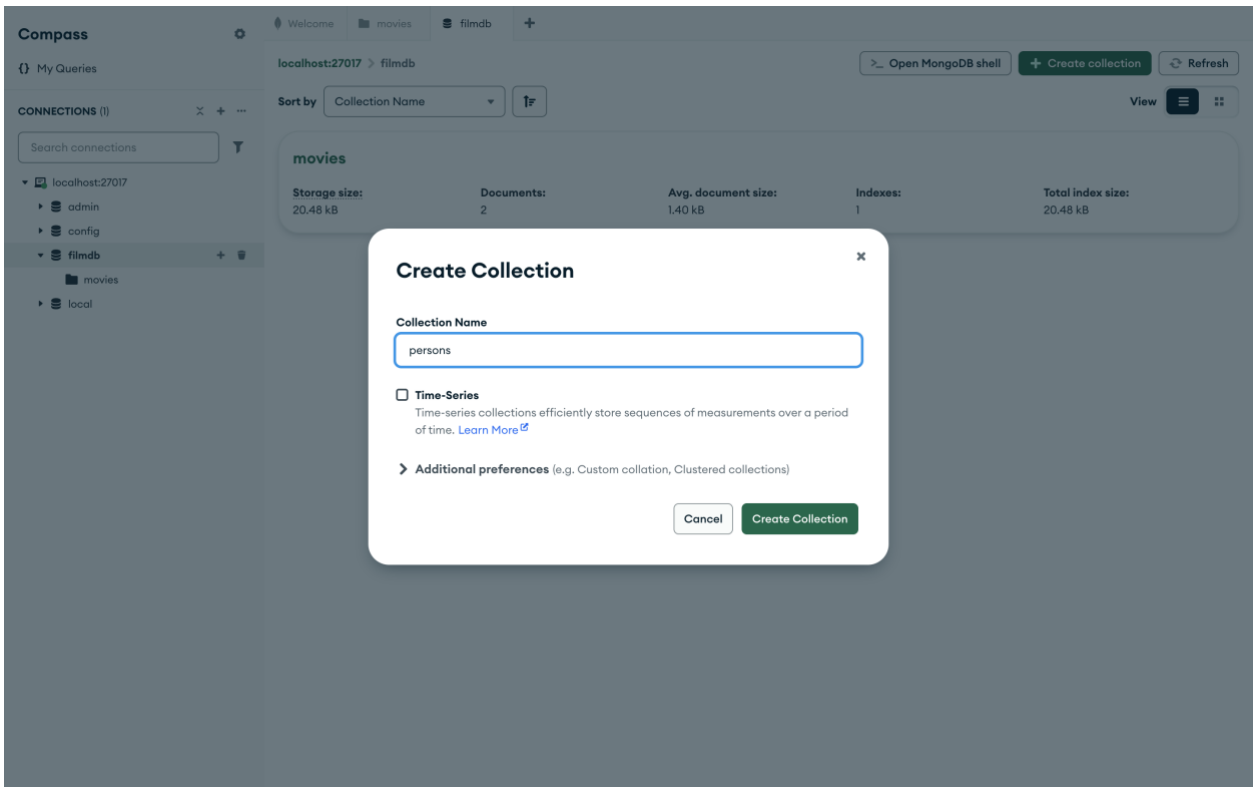

Compass
My Queries
CONNECTIONS (1)
Search connections
localhost:27017
admin
config
local
Welcome
Create Database
Database Name
filmdb
Collection Name
movies
Time-Series
Time-series collections efficiently store sequences of measurements over a period of time. Learn More
Additional preferences (e.g. Custom collation, Clustered collections)
Cancel
Create Database
DB Compass
server or
ve a cluster?
r, you can create one for free
Compass is ready to update to 1.46.9!
RESTART

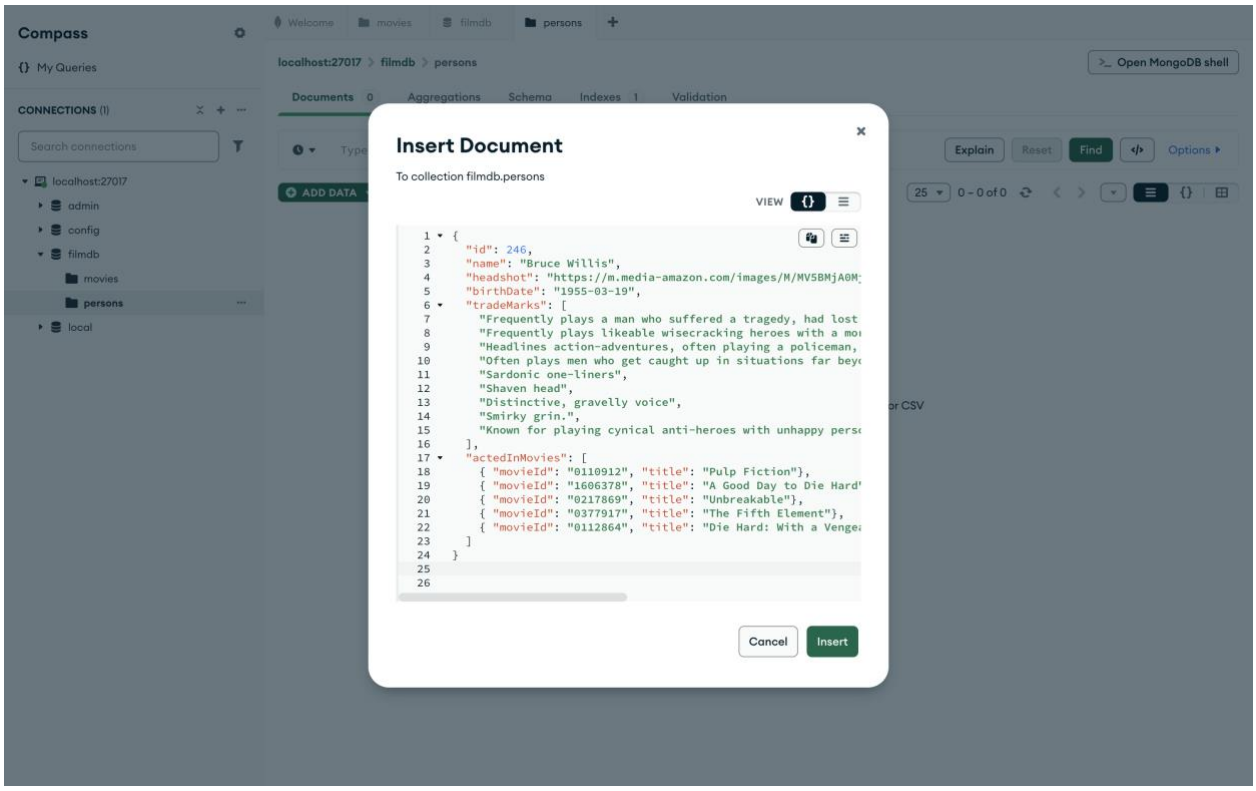

Compass
My Queries
CONNECTIONS (1)
Search connections
localhost:27017
admin
config
filmdb
movies
local
Welcome
movies
Open MongoDB shell
Documents
ADD DATA
Explain
Reset
Find
Options
0 - 0 of 0
or CSV
Insert Document
To collection filmdb.movies
VIEW
{
"id": "0110912",
"title": "Pulp Fiction",
"year": 1994,
"runtime": 154,
"languages": ["en", "es", "fr"],
"rating": 8.9,
"votes": 2084331,
"genres": ["Crime", "Drama"],
"plotOutline": "Jules Winnfield (Samuel L. Jackson) and
"coverUrl": "https://m.media-amazon.com/images/M/MV5BNG
"actors": [
{ "actorID": "0000619", "name": "Tim Roth"},
{ "actorID": "0001625", "name": "Amanda Plummer"},
{ "actorID": "0522503", "name": "Laura Lovelace"},
{ "actorID": "0000237", "name": "John Travolta"},
{ "actorID": "0000168", "name": "Samuel L. Jackson"
{ "actorID": "0482851", "name": "Phil LaMarr"},
{ "actorID": "0001844", "name": "Frank Whaley"},
{ "actorID": "0824882", "name": "Burr Steers"},
{ "actorID": "0000246", "name": "Bruce Willis"},
{ "actorID": "0000609", "name": "Ving Rahmes"},
{ "actorID": "0000235", "name": "Uma Thurman"},
{ "actorID": "0000233", "name": "Quentin Tarantino"
],
"directors": [
{ "directorID": "0000233", "name": "Quentin Taranti
],
"producers": [
{ "producerID": "0004744", "name": "Lawrence Bender
{ "producerID": "0000362", "name": "Danny DeVito"},
{ "producerID": "0321621", "name": "Richard N. Glad
{ "producerID": "0787834", "name": "Michael Shamber
Cancel
Insert
Compass is ready to update to 1.46.9!
RESTART

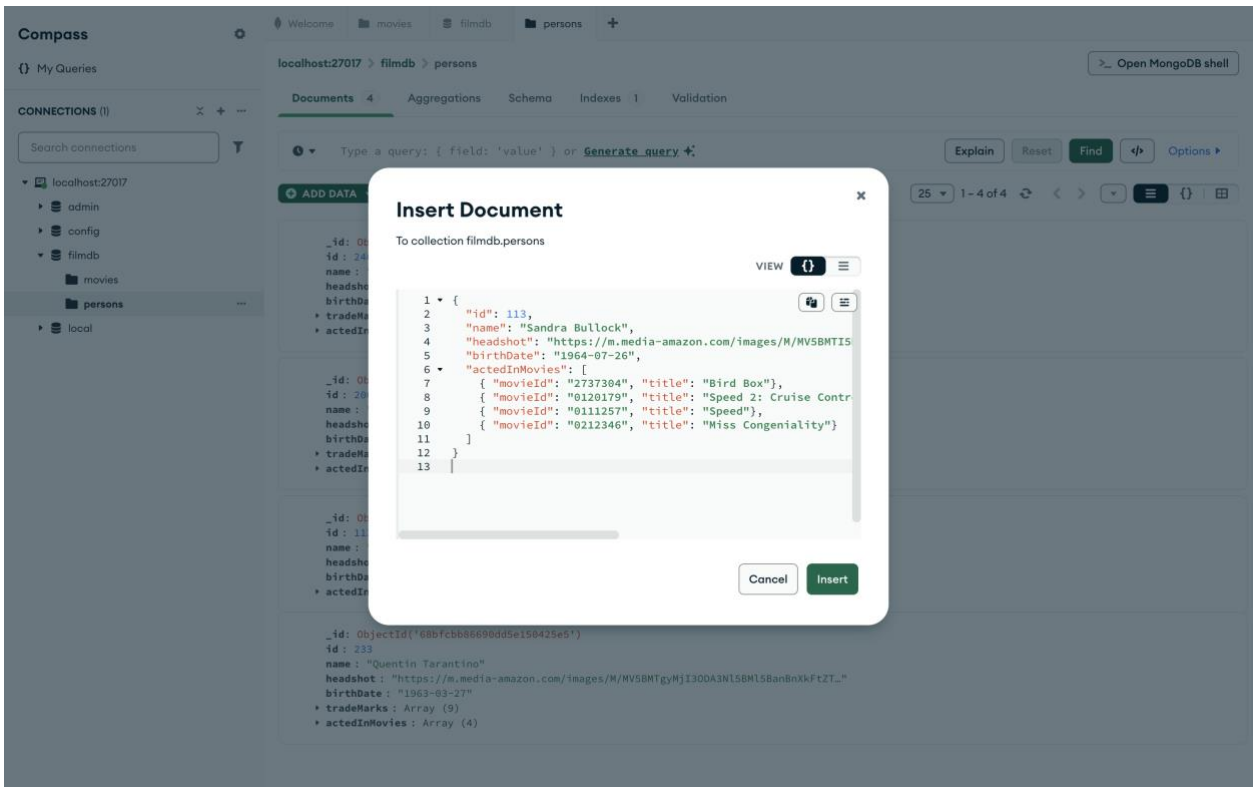


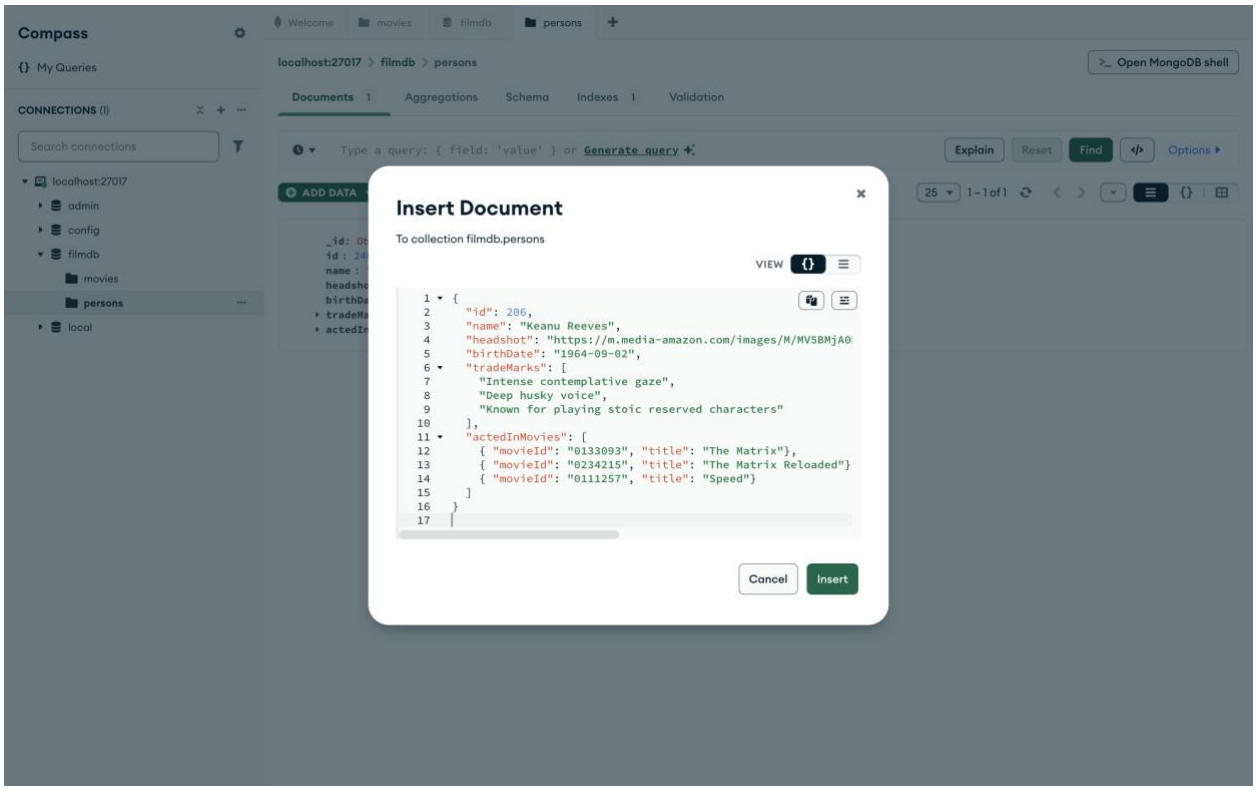


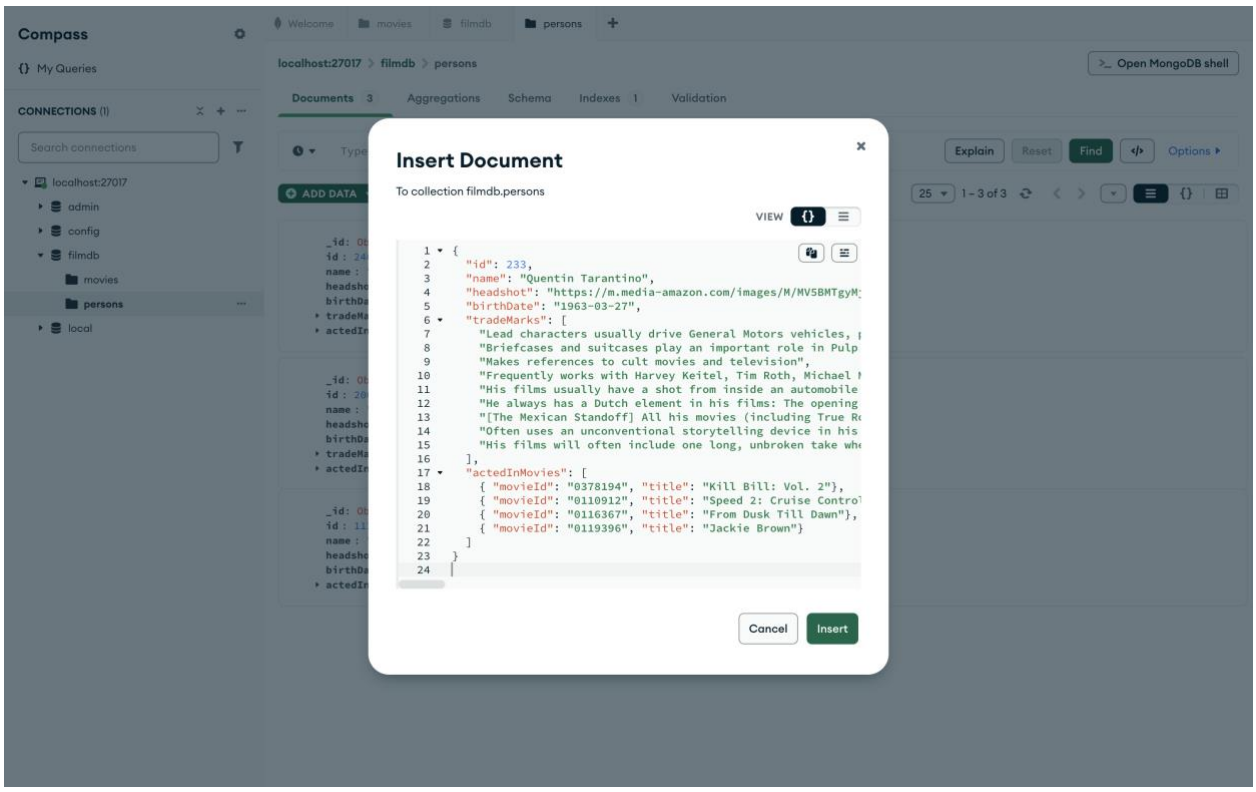
Создала документы актеров в коллекции persons

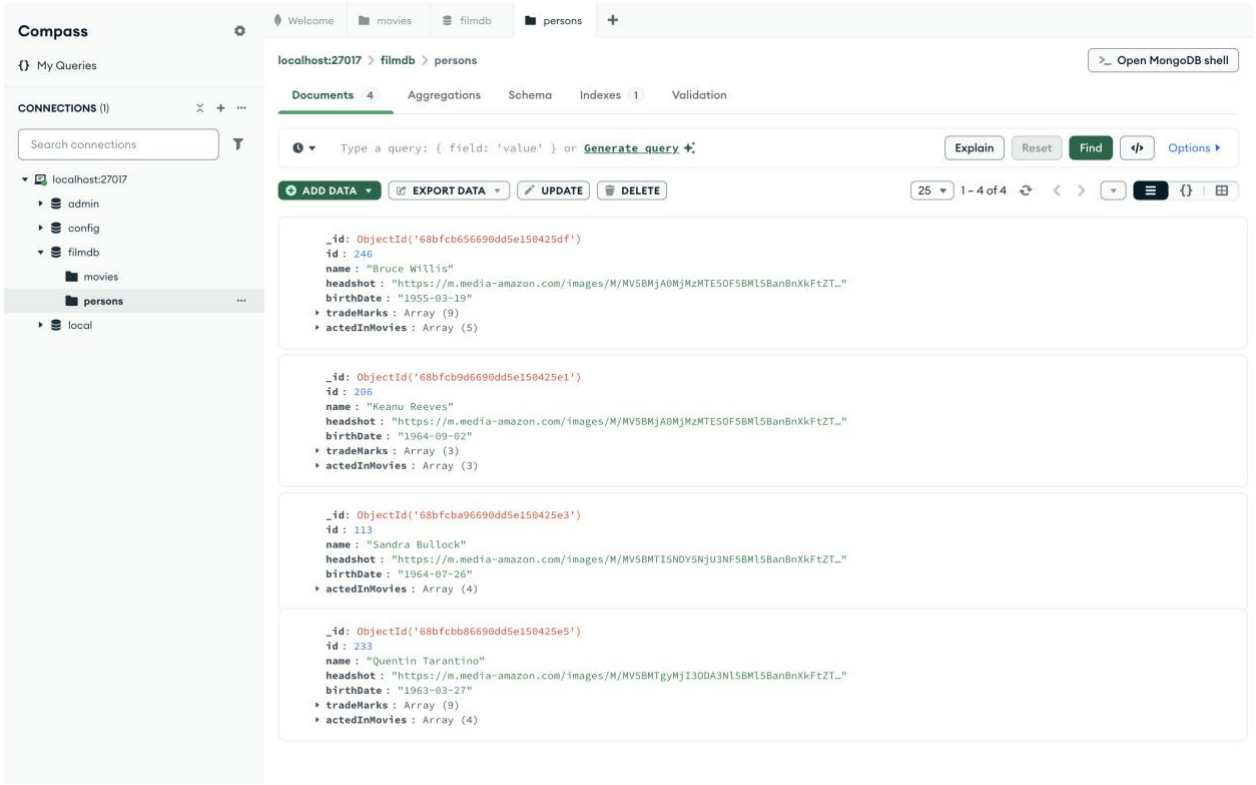

Compass
My Queries
CONNECTIONS (1)
Search connections
localhost:27017
admin
config
filmdb
movies
local
Welcome
movies
filmdb
localhost:27017 > filmdb
Open MongoDB shell
Create collection
Refresh
Sort by
Collection Name
View
movies
Storage size:
20.48 kB
Documents:
2
Avg. document size:
140 kB
Indexes:
1
Total index size:
20.48 kB
Create Collection
Collection Name
persons
Time-Series
Time-series collections efficiently store sequences of measurements over a period of time. Learn More
Additional preferences (e.g. Custom collation, Clustered collections)
Cancel
Create Collection


Compass
My Queries
CONNECTIONS (1)
Search connections
localhost:27017
admin
config
filmdb
movies
persons
local
Welcome
movies
filmdb
persons
localhost:27017 > filmdb > persons
Open MongoDB shell
Documents 0
Aggregations
Schema
Indexes 1
Validation
ADD DATA
Explain
Reset
Find
Options
0 - 0 of 0
Insert Document
To collection filmdb.persons
VIEW
{
"id": 246,
"name": "Bruce Willis",
"headshot": "https://m.media-amazon.com/images/M/MV5BMjA6M
"birthDate": "1955-03-19",
"tradeMarks": [
"Frequently plays a man who suffered a tragedy, had lost
"Frequently plays likeable wisecracking heroes with a mo
"Headlines action-adventures, often playing a policeman,
"Often plays men who get caught up in situations far bey
"Sardonic one-liners",
"Shaven head",
"Distinctive, gravelly voice",
"Smirky grin.",
"Known for playing cynical anti-heroes with unhappy pers
],
"actedInMovies": [
{ "movieId": "0110912", "title": "Pulp Fiction"},
{ "movieId": "1606378", "title": "A Good Day to Die Hard"
{ "movieId": "0217869", "title": "Unbreakable"},
{ "movieId": "0377917", "title": "The Fifth Element"},
{ "movieId": "0112864", "title": "Die Hard: With a Venge
]
}
Cancel
Insert


Compass
My Queries
CONNECTIONS (1)
Search connections
localhost:27017
admin
config
filmdb
movies
persons
local
Welcome
movies
filmdb
persons
localhost:27017 > filmdb > persons
Open MongoDB shell
Documents 4
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
25 1-4 of 4
Insert Document
To collection filmdb.persons
VIEW
{
"id": 113,
"name": "Sandra Bullock",
"headshot": "https://m.media-amazon.com/images/M/MV5BMTIS
"birthDate": "1964-07-26",
"actedInMovies": [
{ "movieId": "2737304", "title": "Bird Box"},
{ "movieId": "0120179", "title": "Speed 2: Cruise Contr
{ "movieId": "0111257", "title": "Speed"},
{ "movieId": "0212346", "title": "Miss Congeniality"}
]
}
Cancel
Insert
_id: ObjectId('68bfcbb86690dd5e150425e5')
id : 233
name : "Quentin Tarantino"
headshot : "https://m.media-amazon.com/images/M/MV5BMTgyMjI3ODA3NlS5BMlSBanBnXkFtZT…"
birthDate : "1963-03-27"
tradeMarks : Array (9)
actedInMovies : Array (4)


Compass
My Queries
CONNECTIONS (1)
Search connections
localhost:27017
admin
config
filmdb
movies
persons
local
Welcome
movies
filmdb
persons
localhost:27017 > filmdb > persons
Open MongoDB shell
Documents 1
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
25 1-1 of 1
Insert Document
To collection filmdb.persons
VIEW
{
"id": 206,
"name": "Keanu Reeves",
"headshot": "https://m.media-amazon.com/images/M/MV5BMjA0
"birthDate": "1964-09-02",
"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"}
{ "movieId": "0111257", "title": "Speed"}
]
}
Cancel
Insert





Подключение к среде MongoDB

Запрос документов с помощью селектора запросов

Использовала метод insertMany для добавления нескольких документов JSON одновременно. Предварительно создала подключение к среде MongoDB.

```
(base) sofia@MacBook-Air-2 ~ % docker exec -it mongo-1 mongosh -u root -p abc123! --authenticationDatabase admin

Current Mongosh Log ID: 68bfcea2154255260cfa334f
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&authSource=admin&appName=mongosh+2.5.7
Using MongoDB:          7.0.23
Using Mongosh:          2.5.7

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-09-08T15:04:00.390+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2025-09-08T15:04:03.300+00:00: Unable to tell whether transparent hugepages are disabled for this process via prctl, as a result unable to verify the state of transparent hugepages on this system.
   2025-09-08T15:04:03.304+00:00: vm.max_map_count is too low
------
```

```
...
switched to db filmdb
filmdb> db.movies.countDocuments()
2
filmdb> 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": 8.5, "rank": 27},
...     {"id": "0120815", "title": "Saving Private Ryan", "genres": ["Drama", "War"], "year": 1998, "rating": 8.5, "rank": 28},
...     {"id": "0114814", "title": "The Usual Suspects", "genres": ["Crime", "Mystery", "Thriller"], "year": 1995, "rating": 8.5, "rank": 29},
...     {"id": "0110413", "title": "L\u00e9on: The Professional", "genres": ["Action", "Crime", "Drama", "Thriller"], "year": 1994, "rating": 8.5, "rank": 30},
...     {"id": "0120689", "title": "The Green Mile", "genres": ["Crime", "Drama", "Fantasy", "Mystery"], "year": 1999, "rating": 8.5, "rank": 31},
...     {"id": "0816692", "title": "Interstellar", "genres": ["Adventure", "Drama", "Sci-Fi"], "year": 2014, "rating": 8.5, "rank": 32},
...     {"id": "0054215", "title": "Psycho", "genres": ["Horror", "Mystery", "Thriller"], "year": 1960, "rating": 8.5, "rank": 33},
...     {"id": "0120586", "title": "American History X", "genres": ["Drama"], "year": 1998, "rating": 8.5, "rank": 34},
...     {"id": "0021749", "title": "City Lights", "genres": ["Comedy", "Drama", "Romance"], "year": 1931, "rating": 8.5, "rank": 35},
...     {"id": "0034583", "title": "Casablanca", "genres": ["Drama", "Romance", "War"], "year": 1942, "rating": 8.5, "rank": 36},
...     {"id": "0064116", "title": "Once Upon a Time in the West", "genres": ["Western"], "year": 1968, "rating": 8.5, "rank": 37},
...     {"id": "0253474", "title": "The Pianist", "genres": ["Biography", "Drama", "Music", "War"], "year": 2002, "rating": 8.5, "rank": 38},
...     {"id": "0027977", "title": "Modern Times", "genres": ["Comedy", "Drama", "Family", "Romance"], "year": 1936, "rating": 8.5, "rank": 39},
...     {"id": "1675434", "title": "The Intouchables", "genres": ["Biography", "Comedy", "Drama"], "year": 2011, "rating": 8.5, "rank": 40},
...     {"id": "0407887", "title": "The Departed", "genres": ["Crime", "Drama", "Thriller"], "year": 2006, "rating": 8.5, "rank": 41},
...     {"id": "0088763", "title": "Back to the Future", "genres": ["Adventure", "Comedy", "Sci-Fi"], "year": 1985, "rating": 8.5, "rank": 42},
...     {"id": "0103064", "title": "Terminator 2: Judgment Day", "genres": ["Action", "Sci-Fi"], "year": 1991, "rating": 8.5, "rank": 43},
...     {"id": "2582802", "title": "Whiplash", "genres": ["Drama", "Music"], "year": 2014, "rating": 8.5, "rank": 44},
...     {"id": "0110357", "title": "The Lion King", "genres": ["Animation", "Adventure", "Drama", "Family", "Musical"], "year": 1994, "rating": 8.5, "rank": 45},
...     {"id": "0047396", "title": "Rear Window", "genres": ["Mystery", "Thriller"], "year": 1954, "rating": 8.5, "rank": 46},
...     {"id": "0082971", "title": "Raiders of the Lost Ark", "genres": ["Action", "Adventure"], "year": 1981, "rating": 8.5, "rank": 47},
...     {"id": "0172495", "title": "Gladiator", "genres": ["Action", "Adventure", "Drama"], "year": 2000, "rating": 8.5, "rank": 48},
...     {"id": "0482571", "title": "The Prestige", "genres": ["Drama", "Mystery", "Sci-Fi", "Thriller"], "year": 2006, "rating": 8.5, "rank": 49},
...     {"id": "0078788", "title": "Apocalypse Now", "genres": ["Drama", "War"], "year": 1979, "rating": 8.4, "rank": 50}
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68bfcf05154255260cfa3350'),
    '1': ObjectId('68bfcf05154255260cfa3351'),
    '2': ObjectId('68bfcf05154255260cfa3352'),
    '3': ObjectId('68bfcf05154255260cfa3353'),
    '4': ObjectId('68bfcf05154255260cfa3354'),
    '5': ObjectId('68bfcf05154255260cfa3355'),
```

После выполнения мультивставки убедилась, что в нашей коллекции movies на самом деле имеется 50 фильмов.

```
...     {"id": "0110357", "title": "The Lion King", "genres": ["Animation", "Adventure", "Drama", "Family", "Musical"], "year": 1994, "rating": 8.5, "rank": 45},
...     {"id": "0047396", "title": "Rear Window", "genres": ["Mystery", "Thriller"], "year": 1954, "rating": 8.5, "rank": 46},
...     {"id": "0082971", "title": "Raiders of the Lost Ark", "genres": ["Action", "Adventure"], "year": 1981, "rating": 8.5, "rank": 47},
...     {"id": "0172495", "title": "Gladiator", "genres": ["Action", "Adventure", "Drama"], "year": 2000, "rating": 8.5, "rank": 48},
...     {"id": "0482571", "title": "The Prestige", "genres": ["Drama", "Mystery", "Sci-Fi", "Thriller"], "year": 2006, "rating": 8.5, "rank": 49},
...     {"id": "0078788", "title": "Apocalypse Now", "genres": ["Drama", "War"], "year": 1979, "rating": 8.4, "rank": 50}
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68bfcf05154255260cfa3350'),
    '1': ObjectId('68bfcf05154255260cfa3351'),
    '2': ObjectId('68bfcf05154255260cfa3352'),
    '3': ObjectId('68bfcf05154255260cfa3353'),
    '4': ObjectId('68bfcf05154255260cfa3354'),
    '5': ObjectId('68bfcf05154255260cfa3355'),
    '6': ObjectId('68bfcf05154255260cfa3356'),
    '7': ObjectId('68bfcf05154255260cfa3357'),
    '8': ObjectId('68bfcf05154255260cfa3358'),
    '9': ObjectId('68bfcf05154255260cfa3359'),
    '10': ObjectId('68bfcf05154255260cfa335a'),
    '11': ObjectId('68bfcf05154255260cfa335b'),
    '12': ObjectId('68bfcf05154255260cfa335c'),
    '13': ObjectId('68bfcf05154255260cfa335d'),
    '14': ObjectId('68bfcf05154255260cfa335e'),
    '15': ObjectId('68bfcf05154255260cfa335f'),
    '16': ObjectId('68bfcf05154255260cfa3360'),
    '17': ObjectId('68bfcf05154255260cfa3361'),
    '18': ObjectId('68bfcf05154255260cfa3362'),
    '19': ObjectId('68bfcf05154255260cfa3363'),
    '20': ObjectId('68bfcf05154255260cfa3364'),
    '21': ObjectId('68bfcf05154255260cfa3365'),
    '22': ObjectId('68bfcf05154255260cfa3366'),
    '23': ObjectId('68bfcf05154255260cfa3367'),
    '24': ObjectId('68bfcf05154255260cfa3368'),
    '25': ObjectId('68bfcf05154255260cfa3369'),
    '26': ObjectId('68bfcf05154255260cfa336a'),
    '27': ObjectId('68bfcf05154255260cfa336b'),
    '28': ObjectId('68bfcf05154255260cfa336c'),
    '29': ObjectId('68bfcf05154255260cfa336d'),
    '30': ObjectId('68bfcf05154255260cfa336e'),
    '31': ObjectId('68bfcf05154255260cfa336f'),
    '32': ObjectId('68bfcf05154255260cfa3370'),
    '33': ObjectId('68bfcf05154255260cfa3371'),
    '34': ObjectId('68bfcf05154255260cfa3372'),
    '35': ObjectId('68bfcf05154255260cfa3373'),
    '36': ObjectId('68bfcf05154255260cfa3374'),
    '37': ObjectId('68bfcf05154255260cfa3375'),
    '38': ObjectId('68bfcf05154255260cfa3376'),
    '39': ObjectId('68bfcf05154255260cfa3377'),
    '40': ObjectId('68bfcf05154255260cfa3378'),
    '41': ObjectId('68bfcf05154255260cfa3379'),
    '42': ObjectId('68bfcf05154255260cfa337a'),
    '43': ObjectId('68bfcf05154255260cfa337b'),
    '44': ObjectId('68bfcf05154255260cfa337c'),
    '45': ObjectId('68bfcf05154255260cfa337d'),
    '46': ObjectId('68bfcf05154255260cfa337e'),
    '47': ObjectId('68bfcf05154255260cfa337f')
  }
}
filmdb> db.movies.countDocuments()
50
filmdb>
```

Проверка:

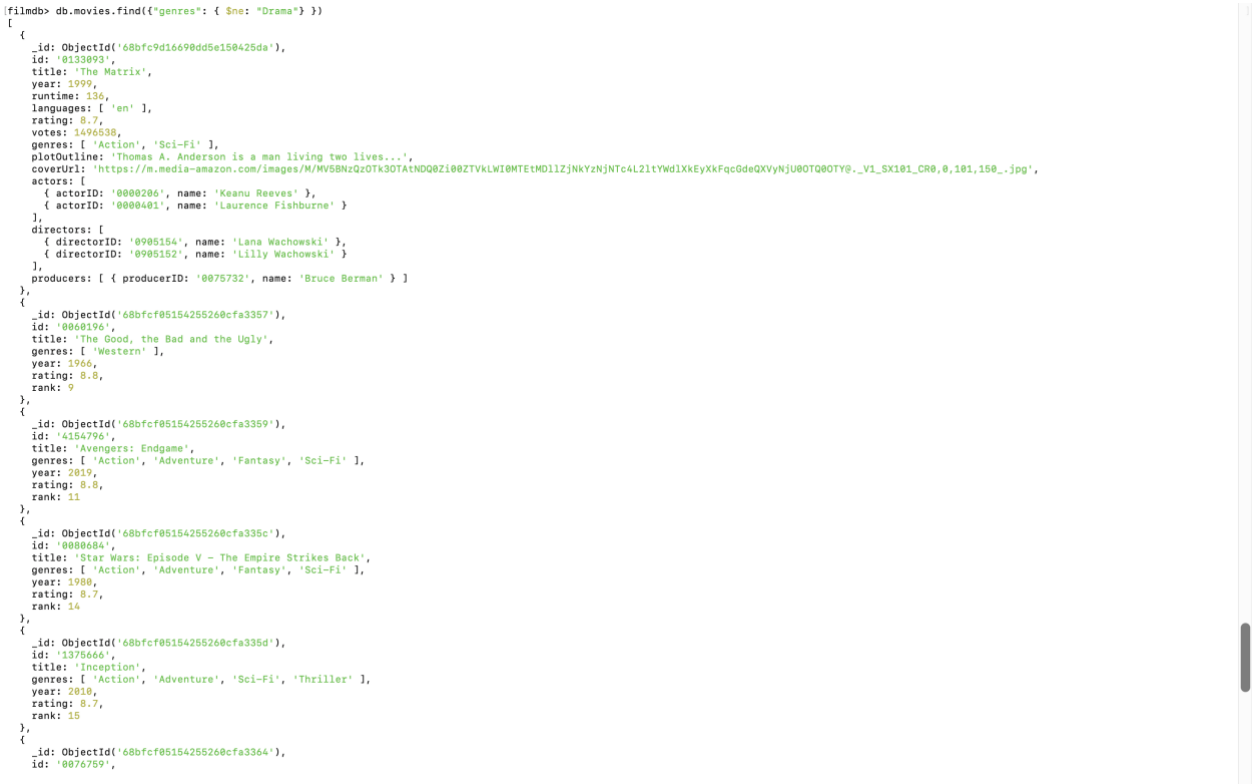
Чтобы получить все Family фильмы, выполнила:

```
filmdb> db.movies.find({"genres": "Family"})
[
  {
    _id: ObjectId('68bfcf05154255260cfa3366'),
    id: '0038650',
    title: "It's a Wonderful Life",
    genres: [ 'Drama', 'Family', 'Fantasy' ],
    year: 1946,
    rating: 8.6,
    rank: 25
  },
  {
    _id: ObjectId('68bfcf05154255260cfa3368'),
    id: '0245429',
    title: 'Spirited Away',
    genres: [ 'Animation', 'Adventure', 'Family', 'Fantasy', 'Mystery' ],
    year: 2001,
    rating: 8.5,
    rank: 27
  },
  {
    _id: ObjectId('68bfcf05154255260cfa3374'),
    id: '0027977',
    title: 'Modern Times',
    genres: [ 'Comedy', 'Drama', 'Family', 'Romance' ],
    year: 1936,
    rating: 8.5,
    rank: 39
  },
  {
    _id: ObjectId('68bfcf05154255260cfa337a'),
    id: '0110357',
    title: 'The Lion King',
    genres: [ 'Animation', 'Adventure', 'Drama', 'Family', 'Musical' ],
    year: 1994,
    rating: 8.5,
    rank: 45
  }
]
filmdb>
```

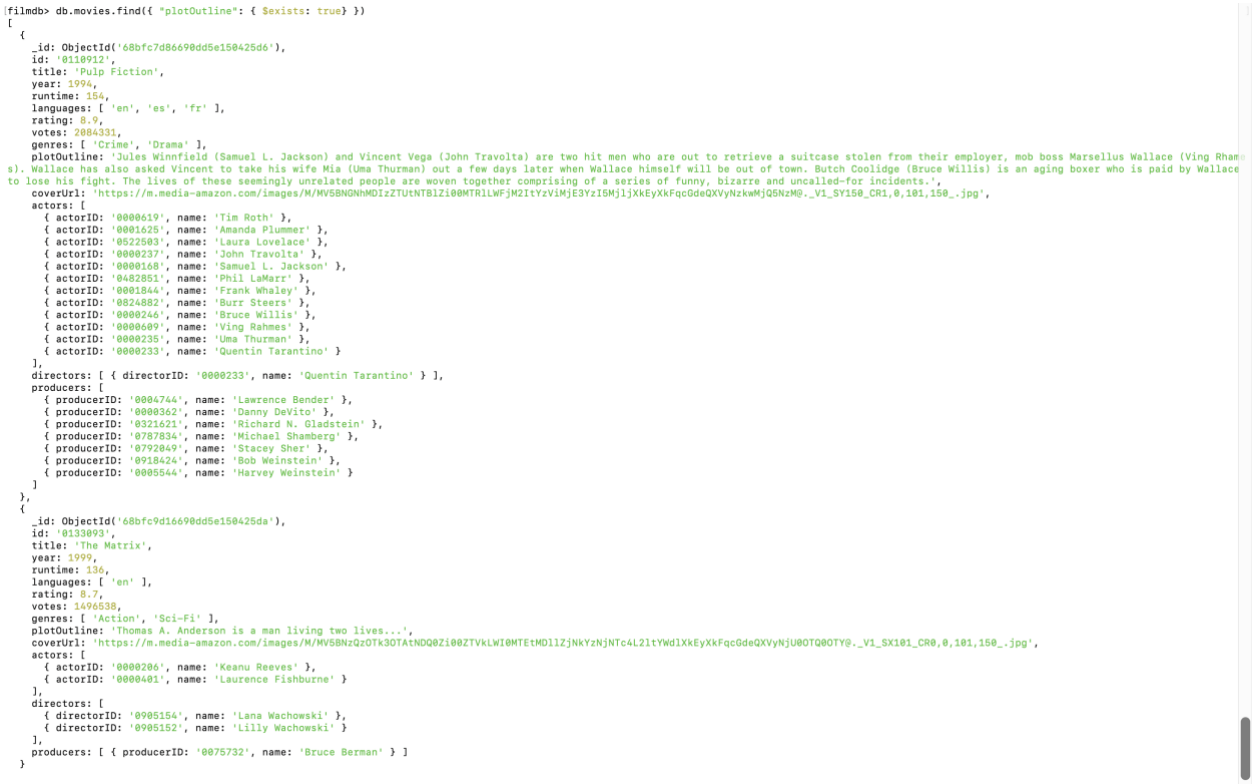
Чтобы хотим получить все фильмы, которые были опубликованы в 2010 году и позже, мы можем сделала следующее:

```
filmdb> db.movies.find({"genres":"Action", "year": { $gte :  2010 } })
[
  {
    _id: ObjectId('68bfcf05154255260cfa3359'),
    id: '4154796',
    title: 'Avengers: Endgame',
    genres: [ 'Action', 'Adventure', 'Fantasy', 'Sci-Fi' ],
    year: 2019,
    rating: 8.8,
    rank: 11
  },
  {
    _id: ObjectId('68bfcf05154255260cfa335d'),
    id: '1375666',
    title: 'Inception',
    genres: [ 'Action', 'Adventure', 'Sci-Fi', 'Thriller' ],
    year: 2010,
    rating: 8.7,
    rank: 15
  }
]
filmdb>
```

Чтобы найти все фильмы, которые не относятся к жанру Drama



Оператор $exists может использоваться для проверки наличия или отсутствия поля.



```
filmdb> db.movies.find({ "plotOutline": { $exists: true} })
[
  {
    _id: ObjectId('68bfc7d86690dd5e150425d6'),
    id: '0110912',
    title: 'Pulp Fiction',
    year: 1994,
    runtime: 154,
    languages: [ 'en', 'es', 'fr' ],
    rating: 8.9,
    votes: 2084331,
    genres: [ 'Crime', 'Drama' ],
    plotOutline: 'Jules Winnfield (Samuel L. Jackson) and Vincent Vega (John Travolta) are two hit men who are out to retrieve a suitcase stolen from their employer, mob boss Marsellus Wallace (Ving Rham
s). Wallace has also asked Vincent to take his wife Mia (Uma Thurman) out a few days later when Wallace himself will be out of town. Butch Coolidge (Bruce Willis) is an aging boxer who is paid by Wallace
to lose his fight. The lives of these seemingly unrelated people are woven together comprising of a series of funny, bizarre and uncalled-for incidents.',
    coverUrl: 'https://m.media-amazon.com/images/M/MV5BNGNhMDIzZTUtNTBlZi00MTRlLWFjM2ItYzViMjE3YzI5MjljXkEyXkFqcGdeQXVyNzkwMjQ5NzM@._V1_SY150_CR1,0,101,150_.jpg',
    actors: [
      { actorID: '0000619', name: 'Tim Roth' },
      { actorID: '0001625', name: 'Amanda Plummer' },
      { actorID: '0522503', name: 'Laura Lovelace' },
      { actorID: '0000237', name: 'John Travolta' },
      { actorID: '0000168', name: 'Samuel L. Jackson' },
      { actorID: '0482851', name: 'Phil LaMarr' },
      { actorID: '0001844', name: 'Frank Whaley' },
      { actorID: '0824882', name: 'Burr Steers' },
      { actorID: '0000246', name: 'Bruce Willis' },
      { actorID: '0000609', name: 'Ving Rahmes' },
      { actorID: '0000235', name: 'Uma Thurman' },
      { actorID: '0000233', name: 'Quentin Tarantino' }
    ],
    directors: [ { directorID: '0000233', name: 'Quentin Tarantino' } ],
    producers: [
      { producerID: '0004744', name: 'Lawrence Bender' },
      { producerID: '0000362', name: 'Danny DeVito' },
      { producerID: '0321621', name: 'Richard N. Gladstein' },
      { producerID: '0787834', name: 'Michael Shamberg' },
      { producerID: '0792049', name: 'Stacey Sher' },
      { producerID: '0918424', name: 'Bob Weinstein' },
      { producerID: '0005544', name: 'Harvey Weinstein' }
    ]
  },
  {
    _id: ObjectId('68bfc9d16690dd5e150425da'),
    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...',
    coverUrl: 'https://m.media-amazon.com/images/M/MV5BNzQzOTk3OTAtNDQ0Zi00ZTVkLWI0MTEtMDllZjNkYzNjNTc4L2ltYWdlXkEyXkFqcGdeQXVyNjU0OTQ0OTY@._V1_SX101_CR0,0,101,150_.jpg',
    actors: [
      { actorID: '0000206', name: 'Keanu Reeves' },
      { actorID: '0000401', name: 'Laurence Fishburne' }
    ],
    directors: [
      { directorID: '0905154', name: 'Lana Wachowski' },
      { directorID: '0905152', name: 'Lilly Wachowski' }
    ],
    producers: [ { producerID: '0075732', name: 'Bruce Berman' } ]
  }
```

Мы видим, что только два фильма имеют установленное свойство plotOutline.

Оператор $in можно использовать для сопоставления одного из нескольких значений, которые мы передаем как массив.

```
filmdb> db.movies.find({ "genres": { $in: ['Family', 'Mistery']} })
[
  {
    _id: ObjectId('68bfcf05154255260cfa3366'),
    id: '0038650',
    title: "It's a Wonderful Life",
    genres: [ 'Drama', 'Family', 'Fantasy' ],
    year: 1946,
    rating: 8.6,
    rank: 25
  },
  {
    _id: ObjectId('68bfcf05154255260cfa3368'),
    id: '0245429',
    title: 'Spirited Away',
    genres: [ 'Animation', 'Adventure', 'Family', 'Fantasy', 'Mystery' ],
    year: 2001,
    rating: 8.5,
    rank: 27
  },
  {
    _id: ObjectId('68bfcf05154255260cfa3374'),
    id: '0027977',
    title: 'Modern Times',
    genres: [ 'Comedy', 'Drama', 'Family', 'Romance' ],
    year: 1936,
    rating: 8.5,
    rank: 39
  },
  {
    _id: ObjectId('68bfcf05154255260cfa337a'),
    id: '0110357',
    title: 'The Lion King',
    genres: [ 'Animation', 'Adventure', 'Drama', 'Family', 'Musical' ],
    year: 1994,
    rating: 8.5,
    rank: 45
  }
]
filmdb>
```

который возвращает все фильмы в жанре Family или Mistery.

Чтобы найти все фильмы жанра Music OR которые были выпущены в 2012 году или позже

```
filmdb> db.movies.find({ $or: [ { "genres":"Music" },  { "year": { $gte :  2012 } } ] })
[
  {
    _id: ObjectId('68bfcf05154255260cfa3359'),
    id: '4154796',
    title: 'Avengers: Endgame',
    genres: [ 'Action', 'Adventure', 'Fantasy', 'Sci-Fi' ],
    year: 2019,
    rating: 8.8,
    rank: 11
  },
  {
    _id: ObjectId('68bfcf05154255260cfa336d'),
    id: '0816692',
    title: 'Interstellar',
    genres: [ 'Adventure', 'Drama', 'Sci-Fi' ],
    year: 2014,
    rating: 8.5,
    rank: 32
  },
  {
    _id: ObjectId('68bfcf05154255260cfa3373'),
    id: '0253474',
    title: 'The Pianist',
    genres: [ 'Biography', 'Drama', 'Music', 'War' ],
    year: 2002,
    rating: 8.5,
    rank: 38
  },
  {
    _id: ObjectId('68bfcf05154255260cfa3379'),
    id: '2582802',
    title: 'Whiplash',
    genres: [ 'Drama', 'Music' ],
    year: 2014,
    rating: 8.5,
    rank: 44
  }
]
filmdb>
```

Чтобы найти все фильмы жанра Action AND, которые были выпущены в 2010 году или позже OR имеют рейтинг выше 8.8

```
filmdb> db.movies.find({ "genres":"Action", $or: [ { "year": { $gte :  2010 } },  { "rating": { $gt :  8.8 } } ] })
[
  {
    _id: ObjectId('68bfcf05154255260cfa3353'),
    id: '0468569',
    title: 'The Dark Knight',
    genres: [ 'Action', 'Crime', 'Drama', 'Thriller' ],
    year: 2008,
    rating: 9,
    rank: 4
  },
  {
    _id: ObjectId('68bfcf05154255260cfa3359'),
    id: '4154796',
    title: 'Avengers: Endgame',
    genres: [ 'Action', 'Adventure', 'Fantasy', 'Sci-Fi' ],
    year: 2019,
    rating: 8.8,
    rank: 11
  },
  {
    _id: ObjectId('68bfcf05154255260cfa335d'),
    id: '1375666',
    title: 'Inception',
    genres: [ 'Action', 'Adventure', 'Sci-Fi', 'Thriller' ],
    year: 2010,
    rating: 8.7,
    rank: 15
  }
]
filmdb>
```

ObjectId, который MongoDB сгенерировал для нашего поля _id, можно выбрать следующим образом:

```
filmdb> db.movies.find({_id: ObjectId("68bfce26315fd46642fa3350")})
...
```

Обновление документов

В своей простейшей форме updateOne() принимает два параметра: селектор (where) для использования и обновления, которые нужно применить к полям. Допустим, мы хотим изменить рейтинг фильма Fight Club на 9

```
filmdb> db.movies.updateOne ( {title: 'Fight Club'} , { $set: {rating: 9} } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
filmdb>
```

Если мы хотим увеличить количество голосов для фильма "The Matrix", которое в настоящее время установлено на 1496538, как мы можем легко увидеть с помощью find

```
filmdb> db.movies.find( {title: 'The Matrix'}, {"votes":1})
[ { _id: ObjectId('68bfc9d16690dd5e150425da'), votes: 1496538 } ]
filmdb> db.movies.updateOne( {title: 'The Matrix'} , { $inc: {votes: 1} } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
filmdb>
```

В данном случае результат говорит нам, что операция успешно обновила один документ с названием "The Matrix", увеличив значение его поля votes на 1.

проверить новый результат, используя тот же find второй раз

```
filmdb> db.movies.find( {title: 'The Matrix'}, {"votes":1})
[ { _id: ObjectId('68bfc9d16690dd5e150425da'), votes: 1496539 } ]
filmdb>
```

Оптимизация производительности с помощью индексов

Давайте добавим индекс по названию фильмов. Для возрастающего индекса по полю укажите значение 1; для убывающего индекса укажите значение -1.

если мы теперь выполним запрос по названию, будет использован индекс

```
filmdb> db.movies.createIndex( {title: 1} );
title_1
filmdb> db.movies.find({ title: "The Matrix" }).explain();
...
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'filmdb.movies',
    indexFilterSet: false,
    parsedQuery: { title: { '$eq': 'The Matrix' } },
    queryHash: '553F28EB',
    planCacheKey: 'DA190D56',
    optimizationTimeMillis: 10,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { title: 1 },
        indexName: 'title_1',
        isMultiKey: false,
        multiKeyPaths: { title: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { title: [ '["The Matrix", "The Matrix"]' ] }
      }
    },
    rejectedPlans: []
  },
  command: { find: 'movies', filter: { title: 'The Matrix' }, '$db': 'filmdb' },
  serverInfo: {
    host: 'mongo-1',
    port: 27017,
    version: '7.0.23',
    gitVersion: '78d6d71385be23831b5971993af60bcafed785bc'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'forceClassicEngine'
  },
  ok: 1
}
filmdb>
```

Давайте добавим индекс по полю id, чтобы убедиться, что оно уникально.

```
filmdb> db.movies.createIndex( {id: 1}, {unique: true} );
id_1
filmdb>
```

Если мы теперь попытаемся добавить один из фильмов второй раз, мы получим ошибку:

```
filmdb> db.movies.insertOne( {"id": "0111161", "title": "The Shawshank Redemption", "genres": ["Drama"], "year": 1994, "rating": 9.2, "rank": 1} )
MongoServerError: E11000 duplicate key error collection: filmdb.movies index: id_1 dup key: { id: "0111161" }
filmdb>
```

Мы можем просмотреть индексы, которые у нас есть в настоящее время в коллекции movies, используя db.movies.getIndexes():

```
filmdb> db.movies.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { title: 1 }, name: 'title_1' },
  { v: 2, key: { id: 1 }, name: 'id_1', unique: true }
]
filmdb>
```

Индекс может быть удален с помощью команды dropIndex.

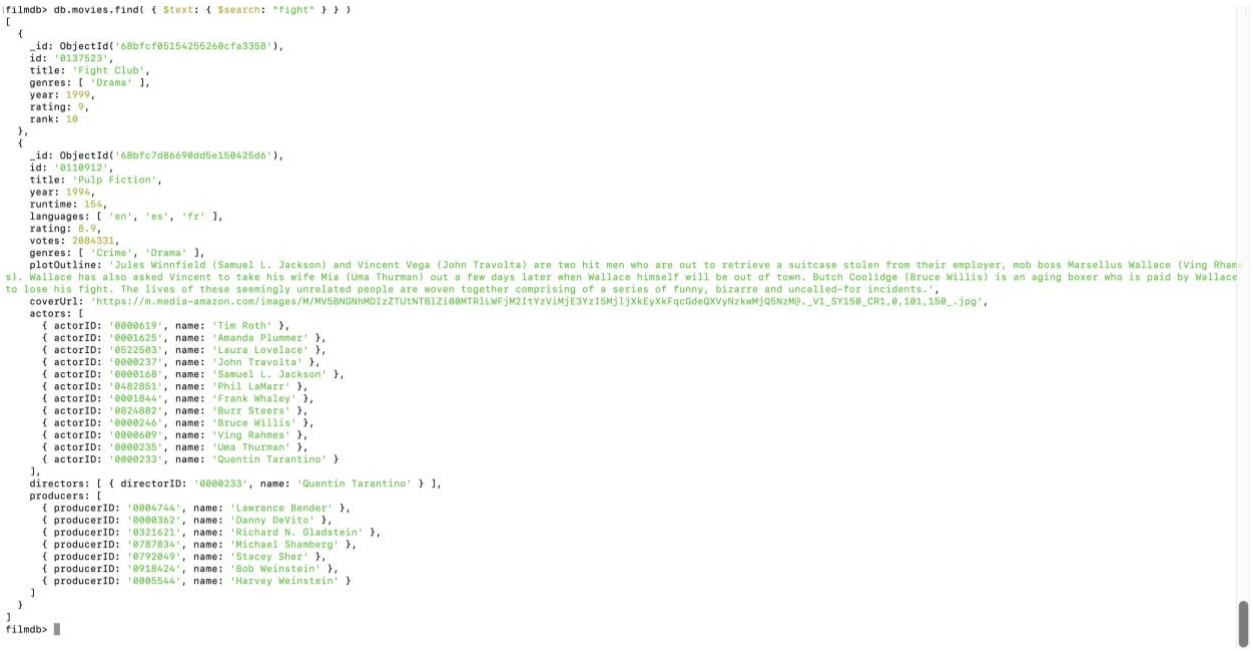
```
filmdb> db.movies.dropIndex( {title: 1} );
{ nIndexesWas: 3, ok: 1 }
filmdb>
```

Текстовый поиск

Например, вы можете выполнить следующее в оболочке mongo, чтобы разрешить текстовый поиск по полям title и plotOutline:

```
filmdb> db.movies.createIndex ( { title: "text", plotOutline: "text" } )
title_text_plotOutline_text
```

Теперь давайте выполним текстовый поиск по термину "fight"



Агрегация данных

мы хотим увидеть, сколько фильмов у нас есть для разных рейтингов

```
filmdb> db.movies.aggregate( [{$group:{_id:'$rating', total: { $sum:1 }}}])
 [
   { _id: 8.6, total: 7 },
   { _id: 8.9, total: 4 },
   { _id: 8.4, total: 1 },
   { _id: 8.8, total: 3 },
   { _id: 8.5, total: 23 },
   { _id: 8.7, total: 7 },
   { _id: 9, total: 3 },
   { _id: 9.2, total: 2 }
 ]
 filmdb>
```

В следующем примере мы группируем по genres и подсчитываем количество фильмов для каждого жанра. Поскольку поле genres является массивом, сначала мы должны использовать $unwind для разворачивания массива. Мы также возвращаем минимальный, максимальный и средний рейтинг для каждой группы. Результат сортируется по количеству фильмов в жанре в порядке убывания.

```
filmdb> db.movies.aggregate([
...                     {$match: {year:{$gt:2000}}},
...                     {$unwind: "$genres" },
...                     {$group: {_id:'$genres',
...                         number :{ $sum:1 },
...                         minRating:{$min:'$rating'},
...                         maxRating:{$max:'$rating'},
...                         avgRating:{$avg:'$rating'}
...                     }},
...                     {$sort:{number:-1}} ])
[
  {
    _id: 'Drama',
    number: 11,
    minRating: 8.5,
    maxRating: 9,
    avgRating: 8.636363636363637
  },
  {
    _id: 'Adventure',
    number: 7,
    minRating: 8.5,
    maxRating: 8.9,
    avgRating: 8.7
  },
  {
    _id: 'Fantasy',
    number: 5,
    minRating: 8.5,
    maxRating: 8.9,
    avgRating: 8.74
  },
  {
    _id: 'Sci-Fi',
    number: 4,
    minRating: 8.5,
    maxRating: 8.8,
    avgRating: 8.625
  },
  {
    _id: 'Thriller',
    number: 4,
    minRating: 8.5,
    maxRating: 9,
    avgRating: 8.675
  },
  {
    _id: 'Crime',
    number: 3,
    minRating: 8.5,
    maxRating: 9,
    avgRating: 8.700000000000001
  },
  {
    _id: 'Action',
    number: 3,
    minRating: 8.7,
    maxRating: 9,
    avgRating: 8.833333333333334
  },
  {
    _id: 'Mystery',
    number: 2,
```

Удаление документов

Если мы хотим удалить конкретный документ, например фильм "Fight Club", мы можем выполнить

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

Мы видим, что, как и ожидалось, один фильм был удален.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ 1

Задание 1 (MongoDB)

вариант 17

Найти все фильмы, у которых в массиве genres ровно 2 элемента ($size).

Через терминал:

```
filmdb> db.movies.find({ genres: { $size: 2 } })
...
[
  {
    _id: ObjectId('68bfc7d86690dd5e150425d6'),
    id: '0110912',
    title: 'Pulp Fiction',
    year: 1994,
    runtime: 154,
    languages: [ 'en', 'es', 'fr' ],
    rating: 8.9,
    votes: 2084331,
    genres: [ 'Crime', 'Drama' ],
    plotOutline: 'Jules Winnfield (Samuel L. Jackson) and Vincent Vega (John Travolta) are two hit men who are out to retrieve a suitcase stolen from their employer, mob boss Marsellus Wallace (Ving Rhames). Wallace has also asked Vincent to take his wife Mia (Uma Thurman) out a few days later when Wallace himself will be out of town. Butch Coolidge (Bruce Willis) is an aging boxer who is paid by Wallace to lose his fight. The lives of these seemingly unrelated people are woven together comprising of a series of funny, bizarre and uncalled-for incidents.',
    coverUrl: 'https://m.media-amazon.com/images/M/MV5BNGNhMDIzZTUtNTBlZi00MTRlLWFjM2ItYzViMjE3YzI5MjljXkEyXkFqcGdeQXVyNzkwMjQ5NzM@._V1_SY150_CR1,0,101,150_.jpg',
    actors: [
      { actorID: '0000619', name: 'Tim Roth' },
      { actorID: '0001625', name: 'Amanda Plummer' },
      { actorID: '0522503', name: 'Laura Lovelace' },
      { actorID: '0000237', name: 'John Travolta' },
      { actorID: '0000168', name: 'Samuel L. Jackson' },
      { actorID: '0482851', name: 'Phil LaMarr' },
      { actorID: '0001844', name: 'Frank Whaley' },
      { actorID: '0824882', name: 'Burr Steers' },
      { actorID: '0000246', name: 'Bruce Willis' },
      { actorID: '0000609', name: 'Ving Rahmes' },
      { actorID: '0000235', name: 'Uma Thurman' },
      { actorID: '0000233', name: 'Quentin Tarantino' }
    ],
    directors: [ { directorID: '0000233', name: 'Quentin Tarantino' } ],
    producers: [
      { producerID: '0004744', name: 'Lawrence Bender' },
      { producerID: '0000362', name: 'Danny DeVito' },
      { producerID: '0321621', name: 'Richard N. Gladstein' },
      { producerID: '0787834', name: 'Michael Shamberg' },
      { producerID: '0792049', name: 'Stacey Sher' },
      { producerID: '0918424', name: 'Bob Weinstein' },
      { producerID: '0005544', name: 'Harvey Weinstein' }
    ]
  },
  {
    _id: ObjectId('68bfc9d16690dd5e150425da'),
    id: '0133093',
    title: 'The Matrix',
    year: 1999,
    runtime: 136,
    languages: [ 'en' ],
    rating: 8.7,
    votes: 1496539,
    genres: [ 'Action', 'Sci-Fi' ],
    plotOutline: 'Thomas A. Anderson is a man living two lives...',
    coverUrl: 'https://m.media-amazon.com/images/M/MV5BNzQzOTk3OTAtNDQ0Zi00ZTVkLWI0MTEtMDllZjNkYzNjNTc4L2ltYWdlXkEyXkFqcGdeQXVyNjU0OTQ0OTY@._V1_SX101_CR0,0,101,150_.jpg',
    actors: [
      { actorID: '0000206', name: 'Keanu Reeves' },
      { actorID: '0000401', name: 'Laurence Fishburne' }
    ],
    directors: [
      { directorID: '0905154', name: 'Lana Wachowski' },
      { directorID: '0905152', name: 'Lilly Wachowski' }
    ],
    producers: [ { producerID: '0075732', name: 'Bruce Berman' } ]
  },
  {
    _id: ObjectId('68bff63f154255260cfa3382'),
    id: '0068646',
    title: 'The Godfather',
    genres: [ 'Crime', 'Drama' ],
    year: 1972,
    rating: 9.2,
    rank: 2
  },
  {
    _id: ObjectId('68bff63f154255260cfa3383'),
    id: '0071562',
    title: 'The Godfather: Part II',
    genres: [ 'Crime', 'Drama' ],
    year: 1974,
    rating: 9,
    rank: 3
  },
  {
    _id: ObjectId('68bff63f154255260cfa338c'),
    id: '0109830',
    title: 'Forrest Gump',
    genres: [ 'Drama', 'Romance' ],
    year: 1994,
    rating: 8.7,
    rank: 13
  },
  {
    _id: ObjectId('68bff63f154255260cfa3392'),
    id: '0047478',
    title: 'Seven Samurai',
    genres: [ 'Adventure', 'Drama' ],
    year: 1954,
    rating: 8.6,
    rank: 20
  },
  {
    _id: ObjectId('68bff63f154255260cfa3394'),
    id: '0317248',
    title: 'City of God',
    genres: [ 'Crime', 'Drama' ],
    year: 2002,
    rating: 8.6,
    rank: 22
  },
  {
    _id: ObjectId('68bff63f154255260cfa339a'),
    id: '0120815',
    title: 'Saving Private Ryan',
    genres: [ 'Drama', 'War' ],
    year: 1998,
    rating: 8.5,
    rank: 28
  },
  {
    _id: ObjectId('68bff63f154255260cfa33a9'),
    id: '0103064',
    title: 'Terminator 2: Judgment Day',
    genres: [ 'Action', 'Sci-Fi' ],
    year: 1991,
    rating: 8.5,
    rank: 43
  },
  {
```

```
    id: '0317248',
    title: 'City of God',
    genres: [ 'Crime', 'Drama' ],
    year: 2002,
    rating: 8.6,
    rank: 22
  },
  {
    _id: ObjectId('68bff63f154255260cfa339a'),
    id: '0120815',
    title: 'Saving Private Ryan',
    genres: [ 'Drama', 'War' ],
    year: 1998,
    rating: 8.5,
    rank: 28
  },
  {
    _id: ObjectId('68bff63f154255260cfa33a9'),
    id: '0103064',
    title: 'Terminator 2: Judgment Day',
    genres: [ 'Action', 'Sci-Fi' ],
    year: 1991,
    rating: 8.5,
    rank: 43
  },
  {
    _id: ObjectId('68bff63f154255260cfa33aa'),
    id: '2582802',
    title: 'Whiplash',
    genres: [ 'Drama', 'Music' ],
    year: 2014,
    rating: 8.5,
    rank: 44
  },
  {
    _id: ObjectId('68bff63f154255260cfa33ac'),
    id: '0047396',
    title: 'Rear Window',
    genres: [ 'Mystery', 'Thriller' ],
    year: 1954,
    rating: 8.5,
    rank: 46
  },
  {
    _id: ObjectId('68bff63f154255260cfa33ad'),
    id: '0082971',
    title: 'Raiders of the Lost Ark',
    genres: [ 'Action', 'Adventure' ],
    year: 1981,
    rating: 8.5,
    rank: 47
  },
  {
    _id: ObjectId('68bff63f154255260cfa33b0'),
    id: '0078788',
    title: 'Apocalypse Now',
    genres: [ 'Drama', 'War' ],
    year: 1979,
    rating: 8.4,
    rank: 50
  }
]
filmdb>
```

На питоне:

```python
from pymongo import MongoClient

client = MongoClient("mongodb://root:abc123!@localhost:27017/")
db = client.filmdb
movies_collection = db.movies

query = { "genres": { "$size": 2 } }
results = movies_collection.find(query)

for movie in results:
    title = movie.get("title", "No Title")
    genres = movie.get("genres", [])
    print(f"{title} - Жанры: {genres}")
```

```
Pulp Fiction - Жанры: ['Crime', 'Drama']
The Matrix - Жанры: ['Action', 'Sci-Fi']
The Godfather - Жанры: ['Crime', 'Drama']
The Godfather: Part II - Жанры: ['Crime', 'Drama']
Forrest Gump - Жанры: ['Drama', 'Romance']
Seven Samurai - Жанры: ['Adventure', 'Drama']
City of God - Жанры: ['Crime', 'Drama']
Saving Private Ryan - Жанры: ['Drama', 'War']
Terminator 2: Judgment Day - Жанры: ['Action', 'Sci-Fi']
Whiplash - Жанры: ['Drama', 'Music']
Rear Window - Жанры: ['Mystery', 'Thriller']
Raiders of the Lost Ark - Жанры: ['Action', 'Adventure']
Apocalypse Now - Жанры: ['Drama', 'War']
```

Начало работы с Redis

Использование утилиты командной строки Redis

Открыла еще одно окно терминала и ввела следующую команду, чтобы запустить Redis CLI в другом docker-контейнере:

```
(base) sofia@MacBook-Air-2 ~ % 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
fd86129b0db3: Pull complete
Digest: sha256:25bf63f3caf75af4628c0dfcf39859ad1ac8abe135be85e99699f9637b16dc28
Status: Downloaded newer image for bitnami/redis:latest
redis 10:15:49.53 INFO  ==>
redis 10:15:49.53 INFO  ==> Welcome to the Bitnami redis container
redis 10:15:49.53 INFO  ==> Subscribe to project updates by watching https://github.com/bitnami/containers
redis 10:15:49.53 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' reposi
tory. More info at https://github.com/bitnami/containers/issues/83267
redis 10:15:49.54 INFO  ==>
```

Для проверки соединения ввела простую команду PING

Ввела help, чтобы увидеть версию установленного Redis.

```
redis-1:6379> PING
PONG
redis-1:6379> help
redis-cli 8.2.1
To get help about Redis commands type:
      "help @<group>" to get a list of commands in <group>
      "help <command>" for help on <command>
      "help <tab>" to get a list of possible help topics
      "quit" to exit

To set redis-cli preferences:
      ":set hints" enable online hints
      ":set nohints" disable online hints
Set your preferences in ~/.redisclirc
redis-1:6379>
```

Структура данных "Строка" (String)

```
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

  GETDEL key
  summary: Returns the string value of a key after deleting the key.
  since: 6.2.0

  GETEX key [EX seconds|PX milliseconds|EXAT unix-time-seconds|PXAT unix-time-milliseconds|PERSIST]
  summary: Returns the string value of a key after setting its expiration time.
  since: 6.2.0

  GETRANGE key start end
  summary: Returns a substring of the string stored at a key.
  since: 2.4.0

  GETSET key value
  summary: Returns the previous string value of a key after setting it to a new value.
  since: 1.0.0

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

  INCRBY key increment
  summary: Increments the integer value of a key by a number. Uses 0 as initial value if the key doesn't exist.
  since: 1.0.0

  INCRBYFLOAT key increment
  summary: Increment the floating point value of a key by a number. Uses 0 as initial value if the key doesn't exist.
  since: 2.6.0

  LCS key1 key2 [LEN] [IDX] [MINMATCHLEN min-match-len] [WITHMATCHLEN]
  summary: Finds the longest common substring.
  since: 7.0.0

  MGET key [key ...]
  summary: Atomically returns the string values of one or more keys.
  since: 1.0.0

  MSET key value [key value ...]
  summary: Atomically creates or modifies the string values of one or more keys.
  since: 1.0.1

  MSETNX key value [key value ...]
  summary: Atomically modifies the string values of one or more keys only when all keys don't exist.
  since: 1.0.1

  PSETEX key milliseconds value
```

Работа с ключами

Можно использовать команду SET для сохранения значения “redis-server” по ключу “server:name”

Redis сохранит данные на постоянной основе, поэтому позже можем спросить: «Какое значение хранится по ключу server:name?»

и Redis ответит “redis-server”.

```
redis-1:6379> SET server:name "redis-server"
OK
redis-1:6379> GET server:name
"redis-server"
redis-1:6379> EXISTS server:name
(integer) 1
redis-1:6379> KEYS server*
1) "server:name"
redis-1:6379> KEYS *
1) "server:name"
2) "task_queue"
redis-1:6379>
```

Операции Get и Set

Сначала установим значение для ключа connections с помощью команды SET

Проверим значение с помощью команды GET

Попробуем перезаписать его с помощью другой команды SET

Посмотрим, что произойдет, если использовать команду SETNX

Если используем SETNX для ключа, который еще не существует, получим другой ответ

Используем MSET для установки нескольких пар «ключ-значение» и обратную команду MGET, чтобы получить несколько значений для нескольких ключей.

```
redis-1:6379> SET connections 10
OK
redis-1:6379> GET connections
"10"
redis-1:6379> SET connections 20
OK
redis-1:6379> GET connections
"20"
redis-1:6379> SETNX connections 30
(integer) 0
redis-1:6379> GET connections
"20"
redis-1:6379> SETNX newkey 30
(integer) 1
redis-1:6379> MSET key1 10 key2 20 key3 30
OK
redis-1:6379> MGET key1 key3
1) "10"
2) "30"
redis-1:6379>
```

Операции инкремента и декремента

Инициализируем значение connections равным 10, а затем используем INCR, чтобы увеличить его на единицу.

Видим, что в ответ получаем новое значение счетчика.

Увеличим его на 10, используя команду INCRBY.

Выполним обратную операцию: уменьшим значение счетчика. С помощью команды DECR счетчик уменьшается на единицу.

А затем с помощью DECRBY укажем величину, на которую уменьшим значение ключа (используем 10).

Удалим пару «ключ-значение» и посмотрим, что произойдет, если используем INCR для несуществующего ключа.

Мы видим, что INCR автоматически начинает со значения 0 и увеличивает его на 1, что и является результатом, который получили.

```
redis-1:6379> SET connections 10
OK
redis-1:6379> INCR connections
(integer) 11
redis-1:6379> INCRBY connections 10
(integer) 21
redis-1:6379> DECR connections
(integer) 20
redis-1:6379> DECRBY connections 10
(integer) 10
redis-1:6379> DEL connections
(integer) 1
redis-1:6379> EXISTS connections
(integer) 0
redis-1:6379> INCR connections
(integer) 1
redis-1:6379>
```

Срок действия (Expiration) и время жизни (TTL)

В Redis можно указать, что ключ должен существовать только в течение определенного времени. Это достигается с помощью команд EXPIRE и TTL. Сначала установим новую пару «ключ-значение».

А затем установим срок его действия в 2 минуты (120 секунд) с помощью команды EXPIRE.

Это устанавливает, что ключ resource:lock будет удален через 120 секунд. Можно проверить, как долго ключ будет существовать, с помощью команды TTL. Она возвращает количество секунд до его удаления. Подождав 114 секунд и выполнив ту же команду снова, видим, что он был удален.

Значение -2 для TTL ключа означает, что ключ (больше) не существует. Проверяем с помощью команды EXISTS.
Если вы SET-ите новое значение для ключа, его TTL будет сброшен. Создать значение сразу со сроком действия. Это можно сделать либо с помощью специальной команды SETEX, либо с помощью SET и опции EX.
Мы видим, что время жизни было установлено при создании.

Теперь используем команду SET, чтобы обновить значение.
Мы видим, что время жизни было сброшено.

```
redis-1:6379> SET resource:lock "Redis Demo"
OK
redis-1:6379> EXPIRE resource:lock 120
(integer) 1
redis-1:6379> TTL resource:lock
(integer) 114
redis-1:6379> TTL resource:lock
(integer) 60
redis-1:6379> TTL resource:lock
(integer) 11
redis-1:6379> TTL resource:lock
(integer) 3
redis-1:6379> TTL resource:lock
(integer) -2
redis-1:6379> EXISTS resource:lock
(integer) 0
redis-1:6379> SET resource:lock "Redis Demo 1" EX 120
OK
redis-1:6379> TTL resource:lock
(integer) 114
redis-1:6379> SET resource:lock "Redis Demo 2"
OK
redis-1:6379> TTL resource:lock
(integer) -1
redis-1:6379>
```

Структуры данных "Список" (List)

Добавить новый элемент в конец несуществующего списка с именем skills с помощью команды RPUSH.

Мы видим, что теперь список содержит 1 элемент. Давайте добавим еще один навык в список skills.

Теперь посмотрим на значения, которые в данный момент находятся в списке skills. Можем использовать команду GET?
Команда GET относится к группе String и не может быть использована для структур list. В таком случае используют команду LRANGE.

LPUSH добавляет новое значение в начало списка.

LRANGE возвращает подмножество списка. Она принимает индекс первого элемента, который требуется получить, в качестве первого параметра, и

индекс последнего элемента, который требуется получить, в качестве второго параметра. Значение -1 для второго параметра означает получение элементов до конца списка.
LLEN возвращает текущую длину списка.
LPOP удаляет первый элемент из списка и возвращает его.
RPOP удаляет последний элемент из списка и возвращает его.
Примечание**:** теперь в списке остался только один элемент

```
redis-1:6379> RPUSH skills "Oracle RDBMS"
(integer) 1
redis-1:6379> RPUSH skills "Redis"
(integer) 2
redis-1:6379> GET skills
(error) WRONGTYPE Operation against a key holding the wrong kind of value
redis-1:6379> LRANGE skills 0 -1
1) "Oracle RDBMS"
2) "Redis"
redis-1:6379> LPUSH skills "SQL Server"
(integer) 3
redis-1:6379> LRANGE skills 0 -1
1) "SQL Server"
2) "Oracle RDBMS"
3) "Redis"
redis-1:6379> LRANGE skills 0 -1
1) "SQL Server"
2) "Oracle RDBMS"
3) "Redis"
redis-1:6379> LRANGE skills 0 1
1) "SQL Server"
2) "Oracle RDBMS"
redis-1:6379> LRANGE skills 1 2
1) "Oracle RDBMS"
2) "Redis"
redis-1:6379> LLEN skills
(integer) 3
redis-1:6379> LPOP skills
"SQL Server"
redis-1:6379> RPOP skills
"Redis"
redis-1:6379> LLEN skills
(integer) 1
redis-1:6379> LRANGE skills 0 -1
1) "Oracle RDBMS"
redis-1:6379>
```

Структуры данных "Множество" (Set)

SADD добавляет заданное значение в множество.

SMEMBERS возвращает список всех элементов этого множества.

SREM удаляет заданное значение из множества.

SISMEMBER проверяет, находится ли заданное значение в множестве.

Cassandra является элементом nosql:products, а MongoDB — нет (поэтому результат 0).

SUNION объединяет два или более множества и возвращает список всех элементов.

Сначала создадим еще одно множество продуктов RDBMS
Теперь создадим объединение двух множеств

SUNIONSTORE объединяет два или более множества и сохраняет результат в новое множество.

SINTER находит пересечение двух или более множеств и возвращает список пересекающихся элементов.

```
redis-1:6379> SADD nosql:products "Cassandra"
(integer) 1
redis-1:6379> SADD nosql:products "Redis"
(integer) 1
redis-1:6379> SADD nosql:products "MongoDB"
(integer) 1
redis-1:6379> SMEMBERS nosql:products
1) "Cassandra"
2) "Redis"
3) "MongoDB"
redis-1:6379> SREM nosql:products "MongoDB"
(integer) 1
redis-1:6379> SMEMBERS nosql:products
1) "Cassandra"
2) "Redis"
redis-1:6379> SISMEMBER nosql:products "Cassandra"
(integer) 1
redis-1:6379> SISMEMBER nosql:products "MongoDB"
(integer) 0
redis-1:6379> SADD rdbms:products "Oracle"
(integer) 1
redis-1:6379> SADD rdbms:products "SQL Server"
(integer) 1
redis-1:6379> SUNION rdbms:products nosql:products
1) "Oracle"
2) "SQL Server"
3) "Cassandra"
4) "Redis"
redis-1:6379> SUNIONSTORE database:products rdbms:products nosql:products
(integer) 4
redis-1:6379> SMEMBERS database:products
1) "Oracle"
2) "SQL Server"
3) "Cassandra"
4) "Redis"
redis-1:6379> SADD favorite:products "Cassandra"
(integer) 1
redis-1:6379> SADD favorite:products "Oracle"
(integer) 1
redis-1:6379> SINTER database:products favorite:products
1) "Cassandra"
2) "Oracle"
redis-1:6379>
```

Структуры данных "Упорядоченное множество" (Sorted Set)

ZADD добавляет один или несколько элементов в упорядоченное множество.

ZRANGE возвращает диапазон элементов в упорядоченном множестве по индексу (отсортировано от меньшего к большему, т.е. по возрастанию), опционально также возвращает оценки. Индекс начинается с 0.

ZREVRANGE возвращает диапазон элементов в упорядоченном множестве по индексу (отсортировано от большего к меньшему, т.е. по убыванию), опционально также возвращает оценки. Индекс начинается с 0.

```
[redis-1:6379> ZADD pioneers 1940 "Alan Kay"
 (integer) 1
[redis-1:6379> ZADD pioneers 1906 "Grace Hopper"
 (integer) 1
[redis-1:6379> ZADD pioneers 1953 "Richard Stallman"
 (integer) 1
[redis-1:6379> ZADD pioneers 1965 "Yukihiro Matsumoto"
 (integer) 1
[redis-1:6379> ZADD pioneers 1916 "Claude Shannon"
 (integer) 1
[redis-1:6379> ZADD pioneers 1969 "Linus Torvalds"
 (integer) 1
[redis-1:6379> ZADD pioneers 1957 "Sophie Wilson"
 (integer) 1
 redis-1:6379> ZADD pioneers 1912 "Alan Turing"
 (integer) 1
[redis-1:6379> ZRANGE pioneers 2 4
 1) "Claude Shannon"
 2) "Alan Kay"
 3) "Richard Stallman"
[redis-1:6379> ZRANGE pioneers 2 4 WITHSCORES
 1) "Claude Shannon"
 2) "1916"
 3) "Alan Kay"
 4) "1940"
 5) "Richard Stallman"
 6) "1953"
[redis-1:6379> ZREVRANGE pioneers 0 2
 1) "Linus Torvalds"
 2) "Yukihiro Matsumoto"
 3) "Sophie Wilson"
 redis-1:6379>
```

Структуры данных "Хэш" (Hash)

HSET устанавливает значение для поля в хэше, хранящемся по заданному ключу.

Чтобы получить сохраненные данные, используйте команду HGETALL

Вы также можете установить несколько полей одновременно с помощью команды HMSET.

Давайте убедимся, что это сработало и был создан новый хэш.

Если требуется получить значение только одного поля, это также возможно с помощью команды HGET.

Числовые значения в полях хэша обрабатываются точно так же, как и в простых строках, и существуют операции для атомарного увеличения этого значения.

```
redis-1:6379> HSET user:1000 name "John Smith"
(integer) 1
redis-1:6379> HSET user:1000 email "john.smith@example.com"
(integer) 1
redis-1:6379>
redis-1:6379> HSET user:1000 password "s3cret"
(integer) 1
redis-1:6379> HGETALL user:1000
1) "name"
2) "John Smith"
3) "email"
4) "john.smith@example.com"
5) "password"
6) "s3cret"
redis-1:6379> HMSET user:1001 name "Mary Jones" password "hidden" email "mjones@example.com"
OK
redis-1:6379> HGETALL user:1001
1) "name"
2) "Mary Jones"
3) "password"
4) "hidden"
5) "email"
6) "mjones@example.com"
redis-1:6379> HGET user:1001 name
"Mary Jones"
redis-1:6379> HSET user:1000 visits 10
(integer) 1
redis-1:6379> HINCRBY user:1000 visits 1
(integer) 11
redis-1:6379> HINCRBY user:1000 visits 10
(integer) 21
redis-1:6379> HDEL user:1000 visits
(integer) 1
redis-1:6379>
```

Дополнительные возможности Redis

Геопространственные данные

Redis поддерживает работу с географическими координатами через команды GEO

```
redis-1:6379> GEOADD cities:russia 37.6176 55.7558 "Moscow"
(integer) 0
redis-1:6379> GEOADD cities:russia 30.3351 59.9311 "Saint Petersburg"
(integer) 0
redis-1:6379> GEOADD cities:russia 82.9346 55.0084 "Novosibirsk"
(integer) 0
redis-1:6379> GEORADIUS cities:russia 37.6176 55.7558 500 km WITHDIST
1) 1) "Moscow"
   2) "0.0002"
```

Python

Установка библиотеки

```
(base) sofia@MacBook-Air-2 ~ % conda install redis-py
Channels:
 - defaults
 - conda-forge
Platform: osx-arm64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/anaconda3

  added / updated specs:
    - redis-py


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    async-timeout-5.0.1        |  py311hca03da5_0          16 KB
    ca-certificates-2025.7.15  |       hca03da5_0         127 KB
    certifi-2025.8.3           |  py311hca03da5_0         161 KB
    conda-24.11.3              |  py311hca03da5_0         1.2 MB
    redis-py-5.2.0             |  py311hca03da5_0         527 KB
    ------------------------------------------------------------
                                           Total:         2.0 MB

The following NEW packages will be INSTALLED:

  async-timeout      pkgs/main/osx-arm64::async-timeout-5.0.1-py311hca03da5_0
  redis-py           pkgs/main/osx-arm64::redis-py-5.2.0-py311hca03da5_0

The following packages will be UPDATED:

  ca-certificates                    2024.9.24-hca03da5_0 --> 2025.7.15-hca03da5_0
  certifi                        2024.8.30-py311hca03da5_0 --> 2025.8.3-py311hca03da5_0
  conda                             24.9.2-py311hca03da5_0 --> 24.11.3-py311hca03da5_0


Proceed ([y]/n)? y


Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(base) sofia@MacBook-Air-2 ~ %
```

Подключение к Redis из Python и проверка соединения

```
import redis
r = redis.Redis(host='localhost', port=6379, db=0)
```

```
!pip install redis
```

```
Requirement already satisfied: redis in /opt/anaconda3/lib/python3.11/site-packages (6.4.0)
```

```
r.ping()
```

```
True
```

```
r.set('foo','bar')
```

```
True
```

Полный пример для Jupyter:

```
import redis

# Создаем подключение (используем localhost, т.к. Jupyter на хосте VM)
# decode_responses=True автоматически преобразует ответы из байтов в строки
# r = redis.Redis(host='localhost', port=6379, db=0, decode_responses=True)
r = redis.Redis(host='localhost', port=6379, db=0, decode_responses=True)

# 1. Проверяем соединение
try:
    if r.ping():
        print("Соединение с Redis установлено!")
except redis.exceptions.ConnectionError as e:
    print(f"Не удалось подключиться к Redis: {e}")

# 2. Устанавливаем значение
key = 'message:1'
value = 'Hello from Python!'
result = r.set(key, value)
print(f"Команда SET для ключа '{key}' выполнена успешно: {result}")

# 3. Получаем значение обратно
retrieved_value = r.get(key)
print(f"Полученное значение для ключа '{key}': {retrieved_value}")

# 4. Удаляем ключ для очистки
r.delete(key)
print(f"Ключ '{key}' удален.")
```

```
Соединение с Redis установлено!
Команда SET для ключа 'message:1' выполнена успешно: True
Полученное значение для ключа 'message:1': Hello from Python!
Ключ 'message:1' удален.
```

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Задание 3 (Redis)

Сохранить в хэш request:xyz данные о ip, user_agent, timestamp. Получить только ip и user_agent с помощью HMGET.

На питоне:

```python
import redis
from datetime import datetime

# подключение к Redis
r = redis.Redis(host="localhost", port=6379, db=0)

# сохраняем данные в хэш
r.hset("request:xyz", mapping={
    "ip": "192.168.0.1",
    "user_agent": "Mozilla/5.0",
    "timestamp": datetime.now().isoformat()
})

# получаем только ip и user_agent
ip, user_agent = r.hmget("request:xyz", "ip", "user_agent")

print("IP:", ip.decode())
print("User-Agent:", user_agent.decode())
```

```
IP: 192.168.0.1
User-Agent: Mozilla/5.0
```

Проверим через терминал: сначала выведем все содержимое хэша, а потом только ip и user_agent с помощью HMGET.

```
redis-1:6379> HGETALL request:xyz
 1) "ip"
 2) "192.168.0.1"
 3) "user_agent"
 4) "Mozilla/5.0"
 5) "timestamp"
 6) "2025-09-09T15:23:39.551179"
redis-1:6379> HMGET request:xyz ip user_agent
 1) "192.168.0.1"
 2) "Mozilla/5.0"
 redis-1:6379>
```

Только при помощи терминала:

```
redis-1:6379> DEL request:xyz
(integer) 1
redis-1:6379> HSET request:xyz ip "192.168.1.100" user_agent "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) Chrome/127.0.0.1 Safari/537.36" timestamp "1725891234"
(integer) 3
redis-1:6379> HGETALL request:xyz
1) "timestamp"
2) "1725891234"
3) "user_agent"
4) "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) Chrome/127.0.0.1 Safari/537.36"
5) "ip"
6) "192.168.1.100"
redis-1:6379> HMGET request:xyz ip user_agent
1) "192.168.1.100"
2) "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) Chrome/127.0.0.1 Safari/537.36"
redis-1:6379>
```

Я удалила старый ключ, заново записала в Redis хэш с ip, user_agent и timestamp, проверила все данные и выбрала отдельно только ip и user_agent.

Работа с Neo4J

Подключение к Cypher Shell

:help, чтобы получить список доступных команд

```
(base) sofia@MacBook-Air-2 ~ % sudo docker exec -ti neo4j-1 ./bin/cypher-shell -u neo4j -p abc123abc123
Password:
Connected to Neo4j using Bolt protocol version 5.4 at neo4j://localhost:7687 as user neo4j.
Type :help for a list of available commands or :exit to exit the shell.
Note that Cypher queries must end with a semicolon.
neo4j@neo4j> :help

Available commands:
  :begin       Open a transaction
  :commit      Commit the currently open transaction
  :connect     Connects to a database
  :disconnect  Disconnects from database
  :exit        Exit the logger
  :help        Show this help message
  :history     Statement history
  :impersonate Impersonate user
  :param       Set the value of a query parameter
  :rollback    Rollback the currently open transaction
  :source      Executes Cypher statements from a file
  :sysinfo     Neo4j system information
  :use         Set the active database

For help on a specific command type:
    :help command

Keyboard shortcuts:
    Up and down arrows to access statement history.
    Tab for autocompletion of commands, hit twice to select suggestion from list using arrow keys.

For help on cypher please visit:
    https://neo4j.com/docs/cypher-manual/current/

neo4j@neo4j>
```

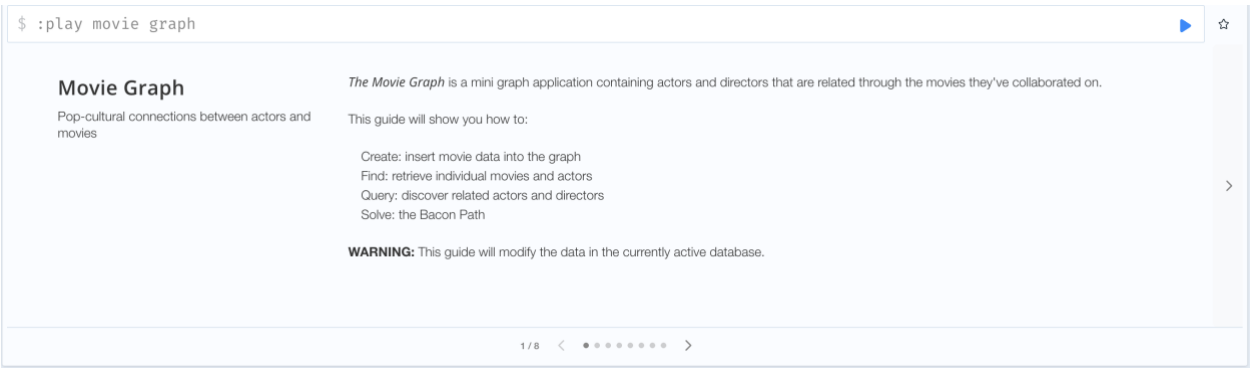
Подключение через Neo4J Browser

подробно описано в разделе Установка в самом начале

Граф фильмов

Культурные связи между актёрами и фильмами

В верхней панели ввела :play movie graph, чтобы начать работать с учебным пособием Movie Graph (Граф фильмов).



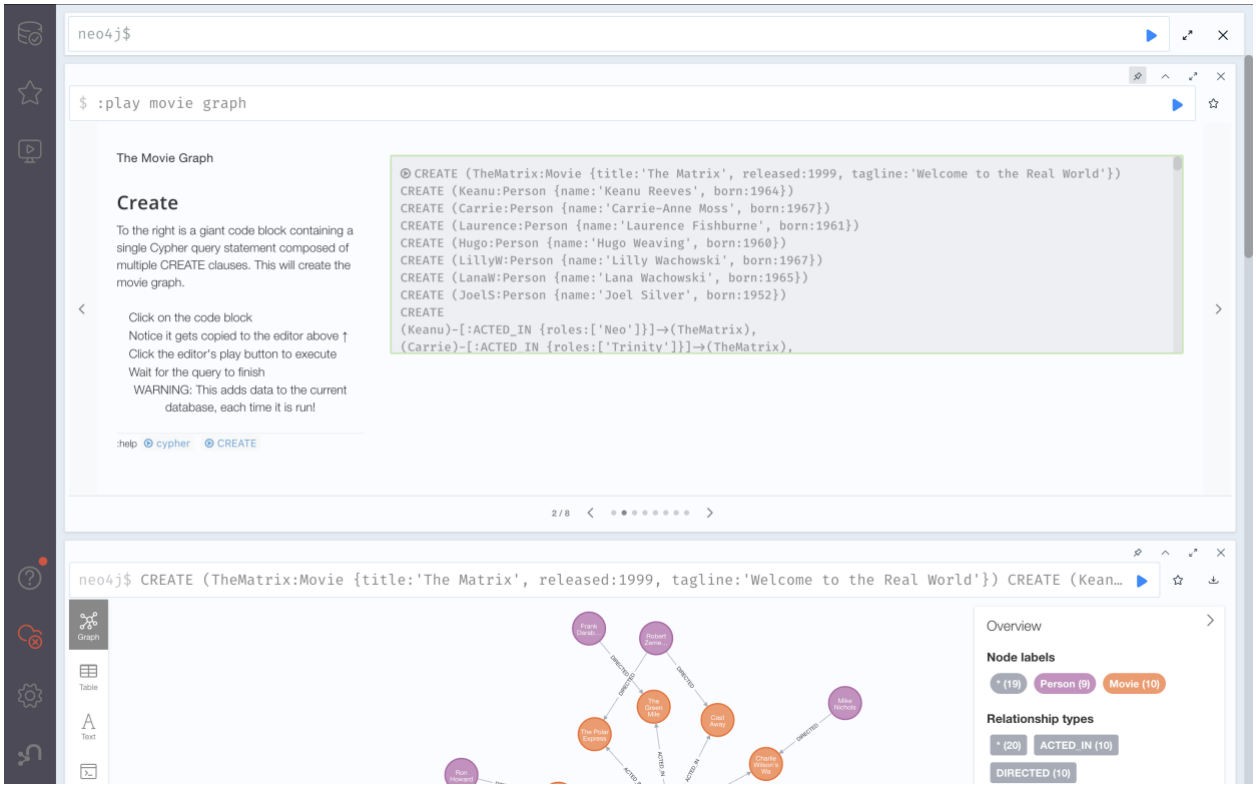
Загрузка графа фильмов

Использовала закрепление (-pin) на панели Movie Graph, чтобы закрепить эту панель сверху.

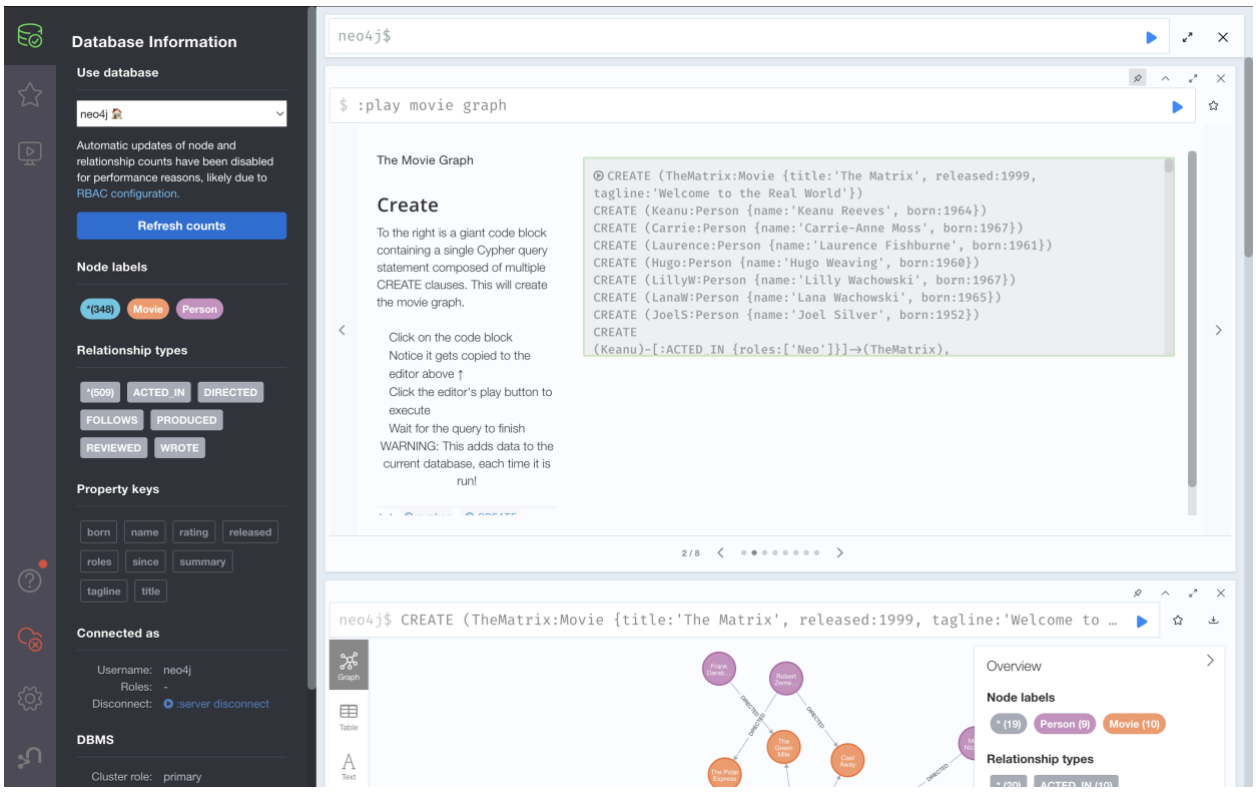
Нажала на следующий шаг в навигационной панели внизу, чтобы перейти к 2/8.

Нажала на маленькую стрелку слева от команды CREATE .., и она должна появиться в верхней панели.

Выполнила команду CREATE .., чтобы создать граф фильмов и добавить некоторые пробные данные о фильмах.



Прежде чем продолжить со следующим шагом, давайте посмотрим, как выглядит граф. Нажала на иконку базы данных в левом верхнем углу браузера Neo4j.

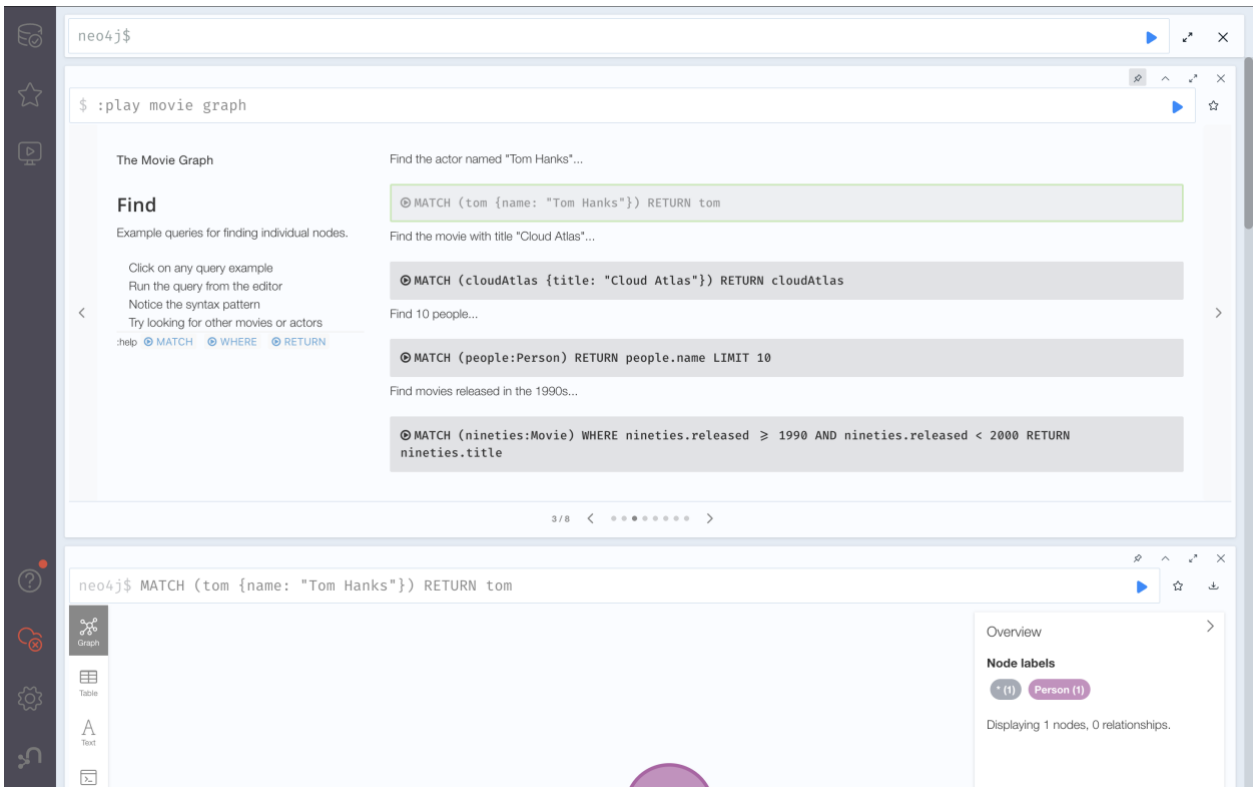


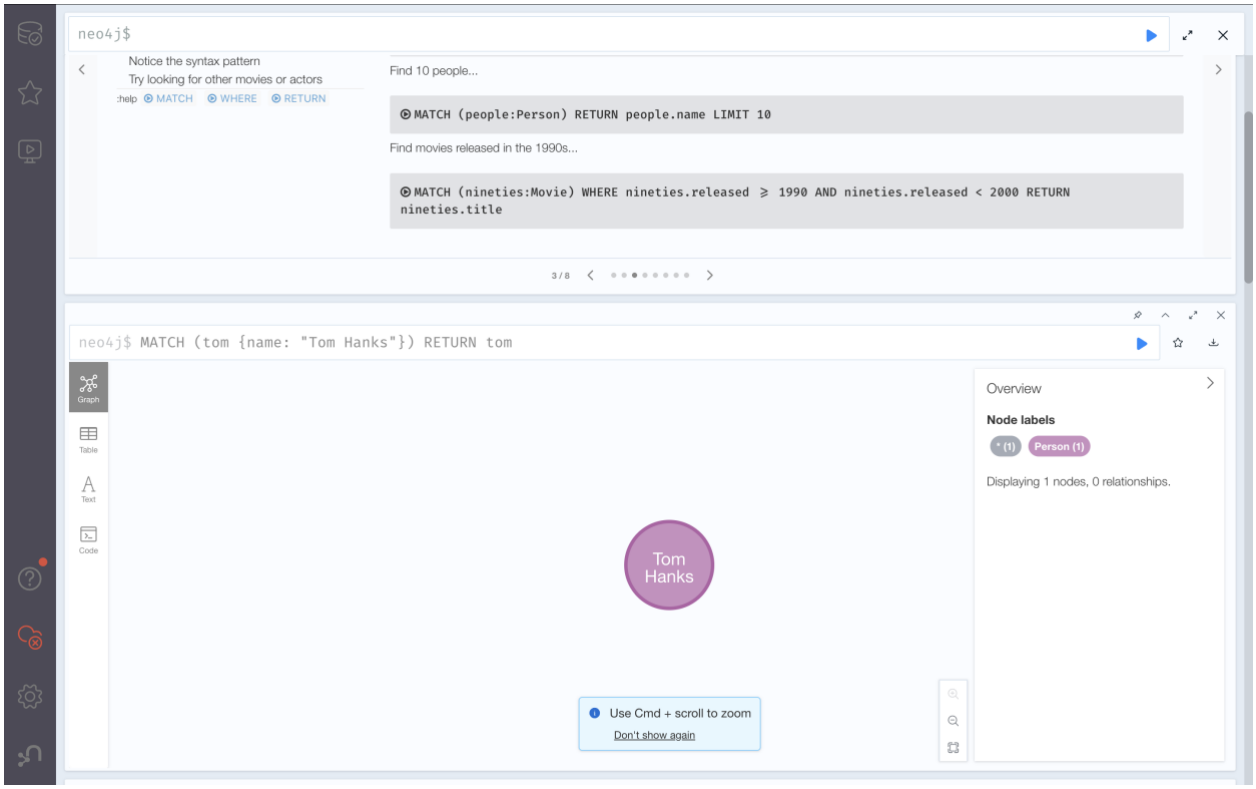
Мы можем увидеть различные Node Labels, Relationship Types и Property Keys, которые были созданы для графа фильмов, а также сколько их создано для каждого типа.

Примеры запросов

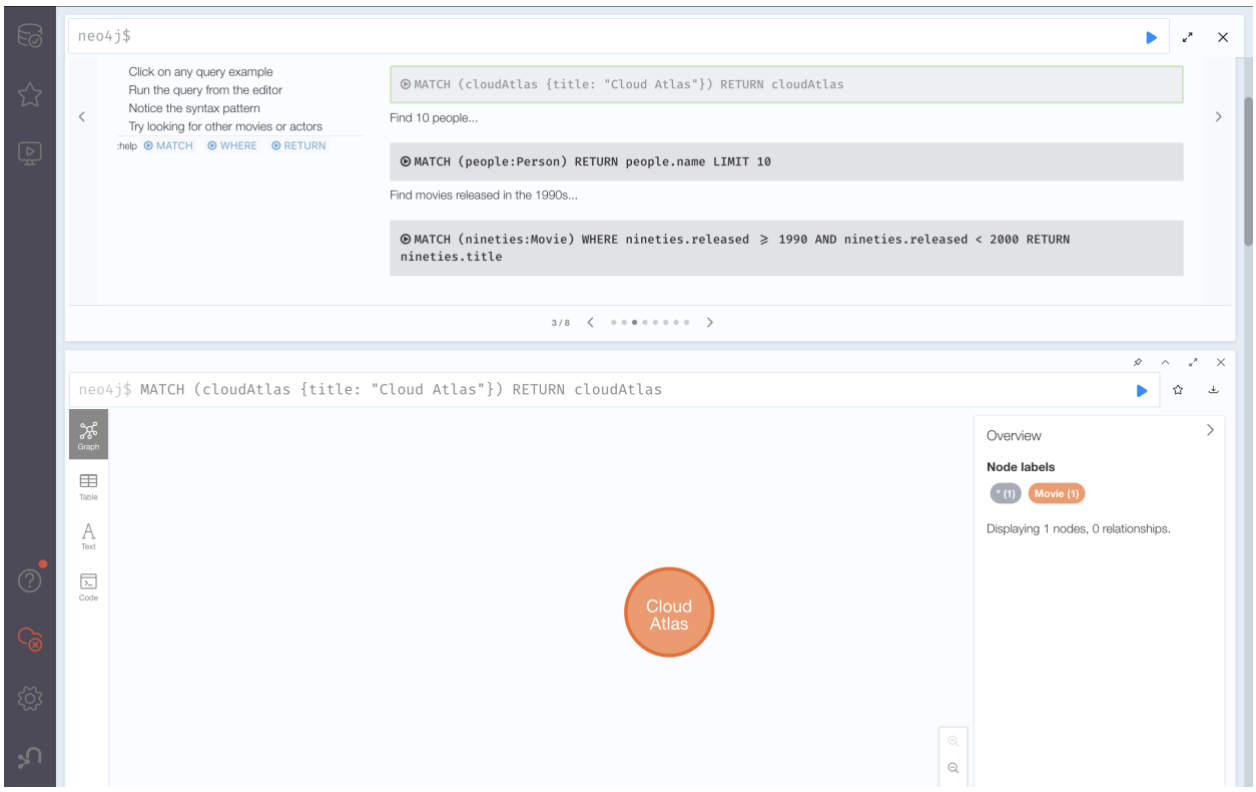
Перешла к шагу 3/8, чтобы найти несколько примеров операторов Cypher для поиска информации в графе.

Первый запрос находит актёра по имени "Том Хэнкс".



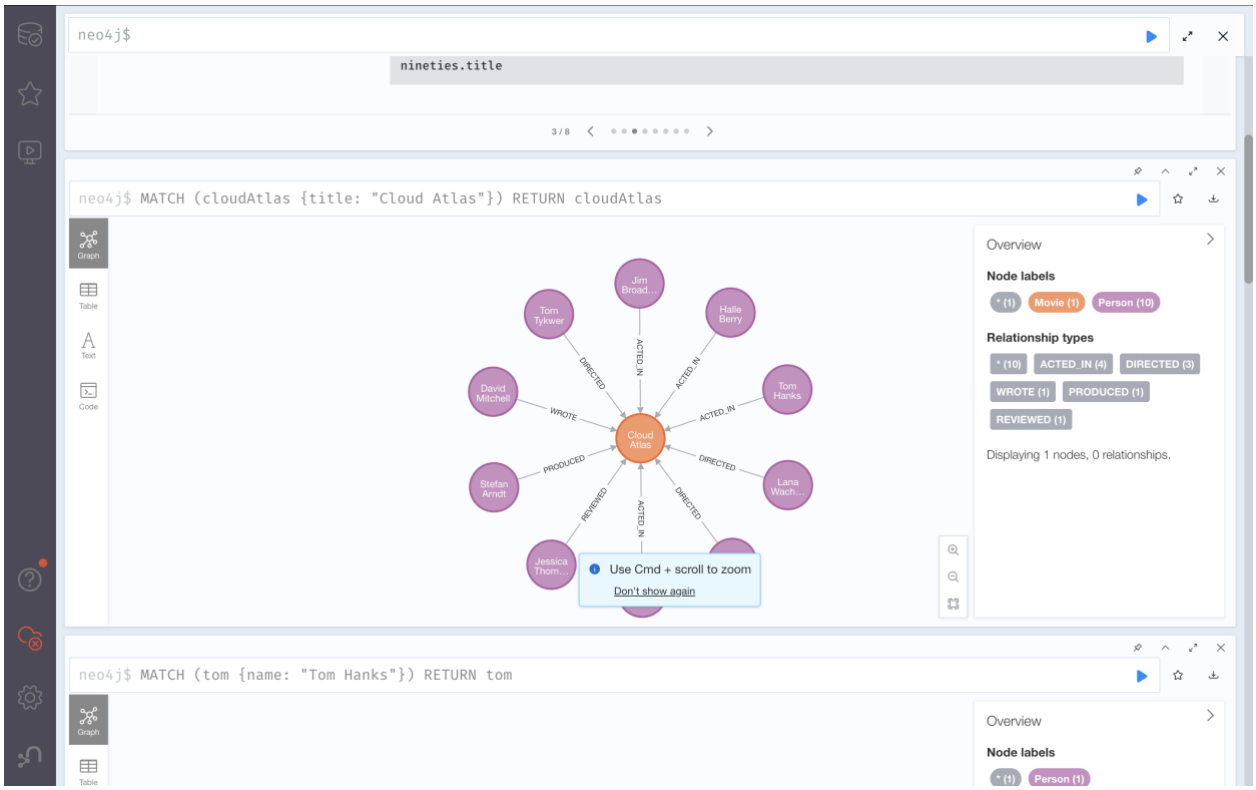


Теперь найдём фильм с названием "Cloud Atlas"



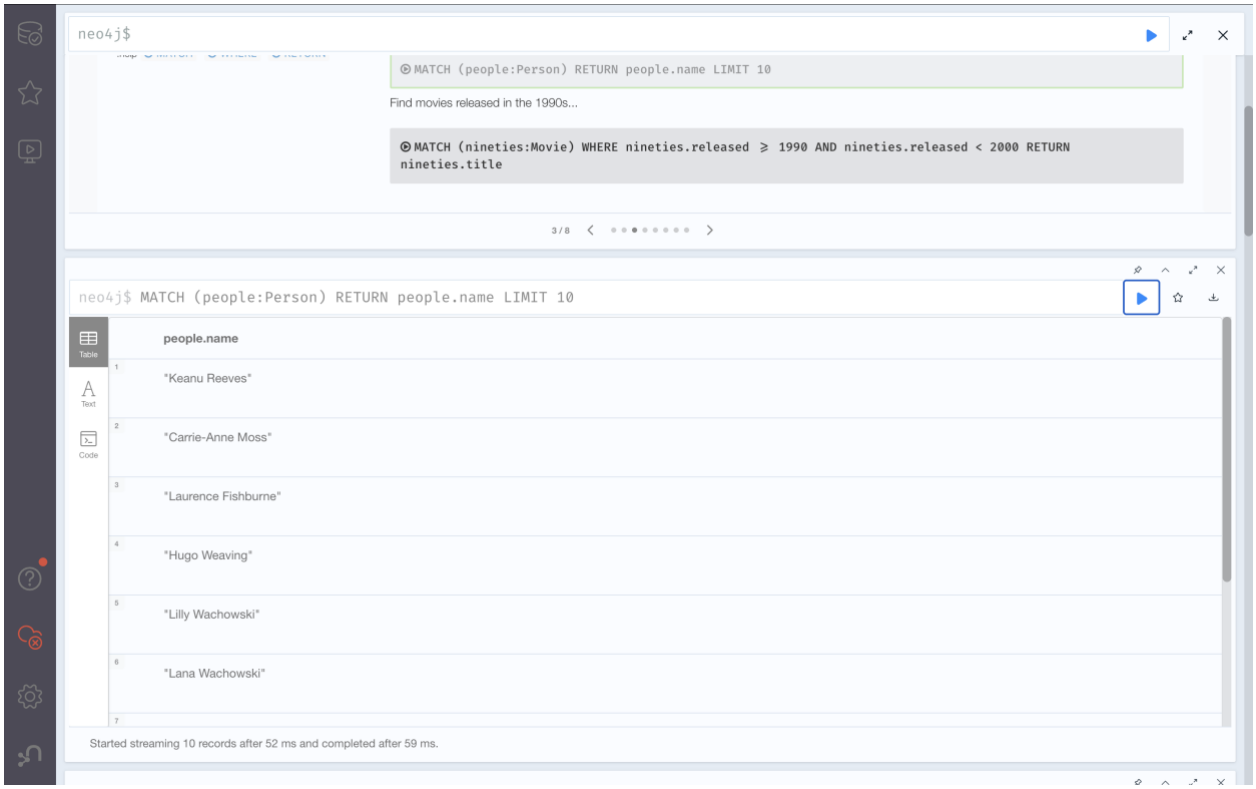
Результат похож на предыдущий, но в этот раз возвращается другой тип узла — Movie (Movie node), и поэтому он показан другим цветом. Вы можете кликнуть по узлу и развернуть связь с/на узел фильма "Cloud Atlas".

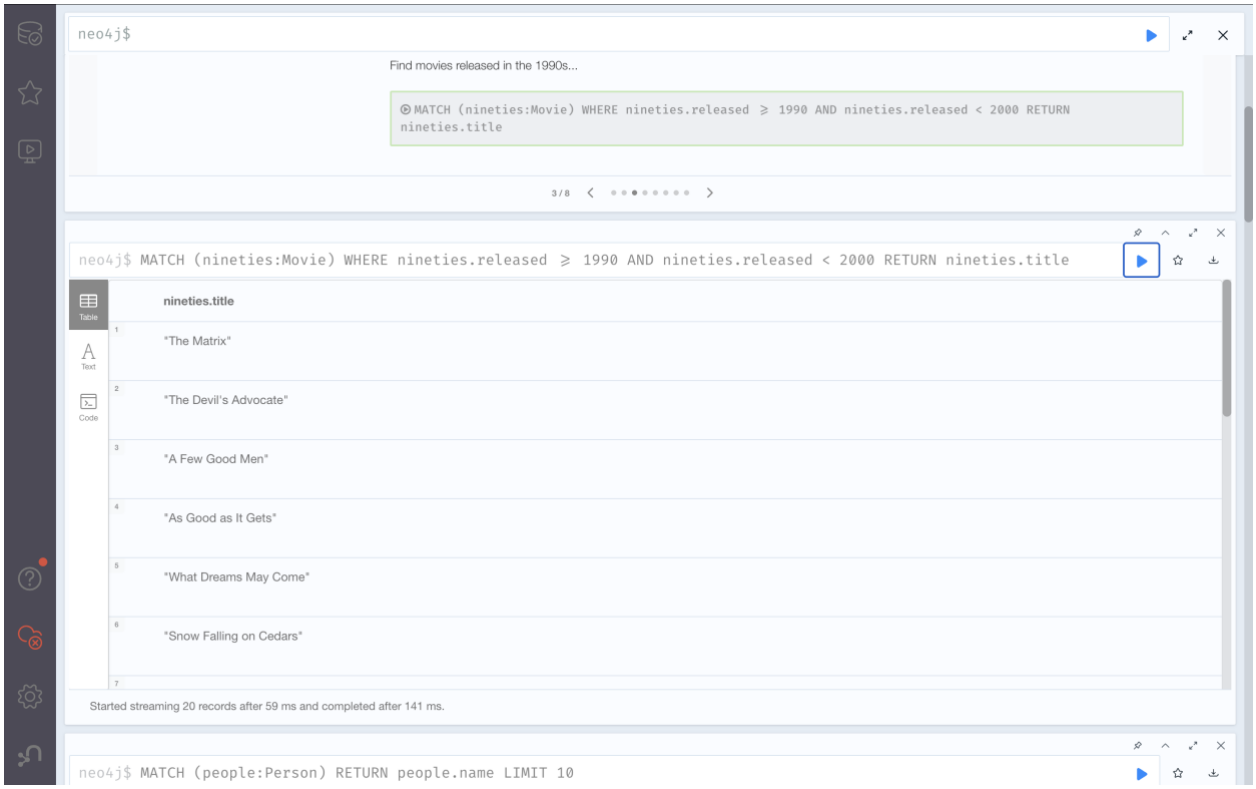
Нажмите на кнопку "Развернуть" (expand), и вы увидите все узлы, связанные с узлом фильма.



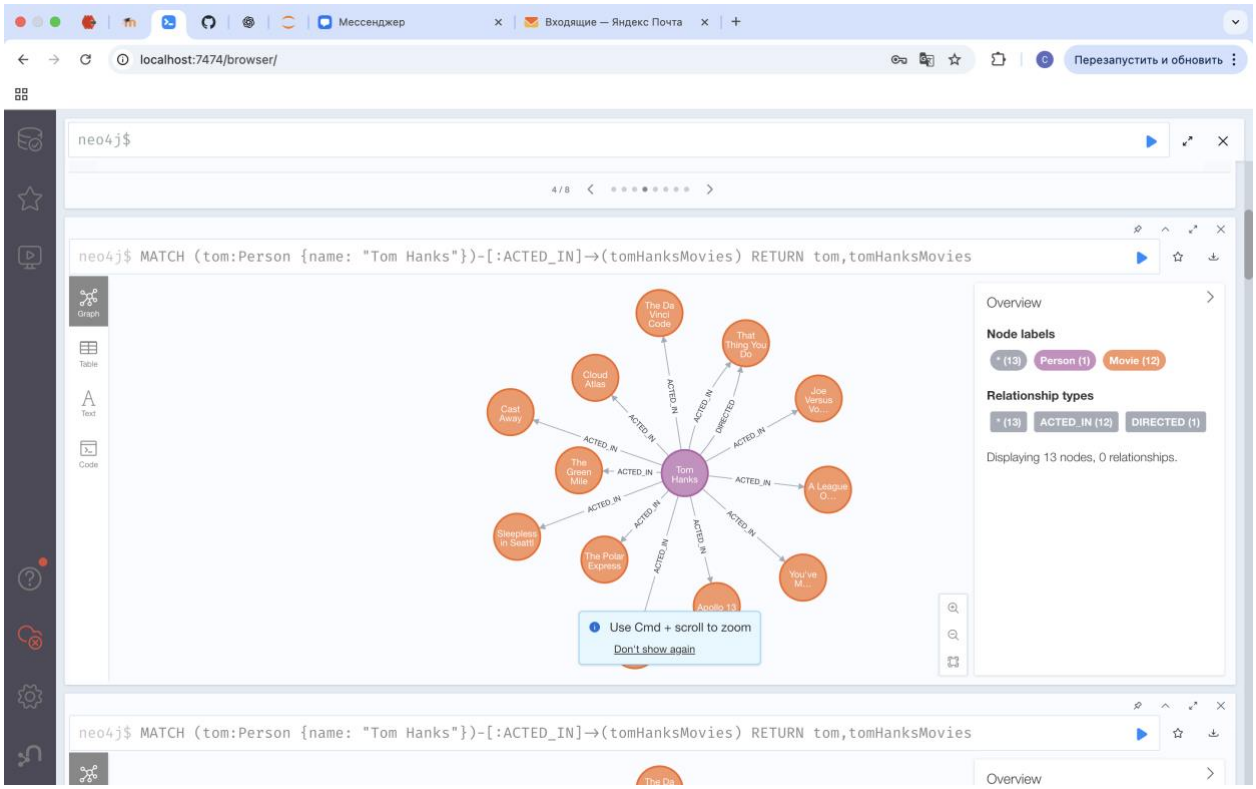
Мы видим, что все они относятся к типу Person (что показывается одинаковым цветом).

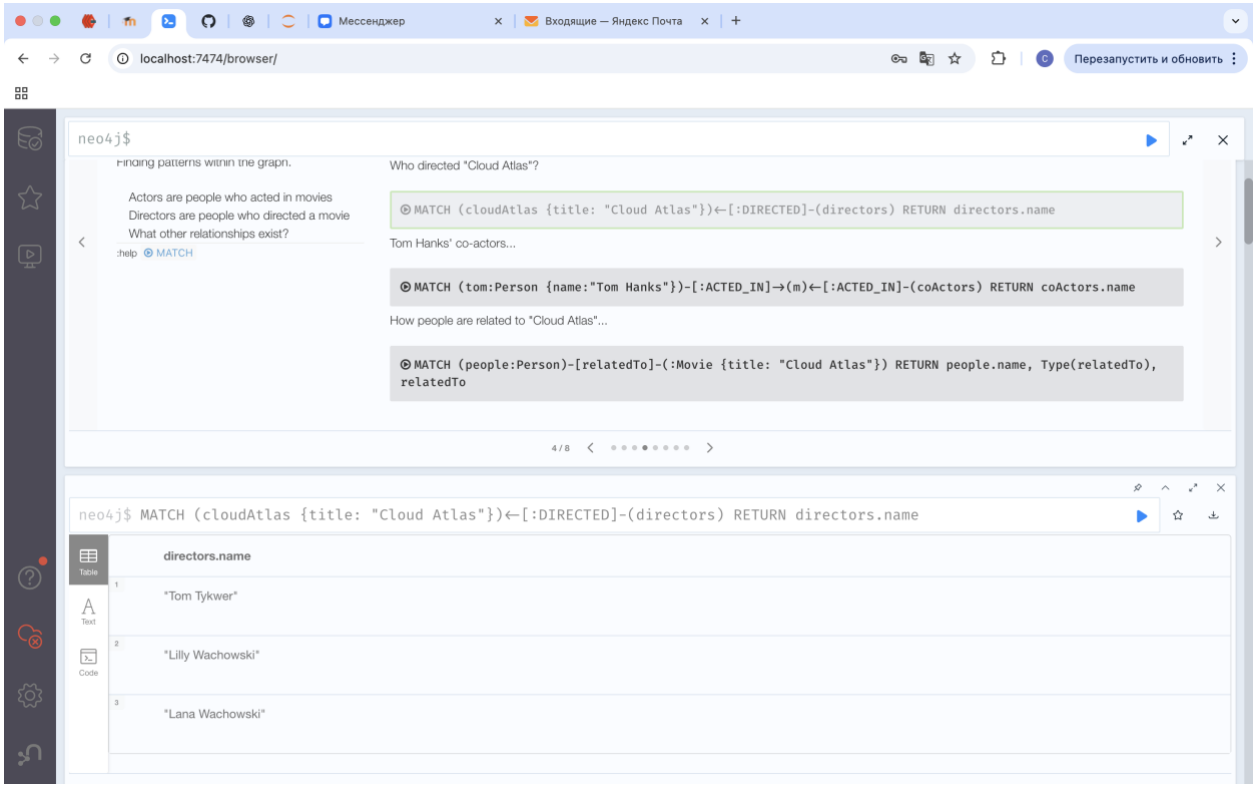
Выполнила все запросы на панели 3/8

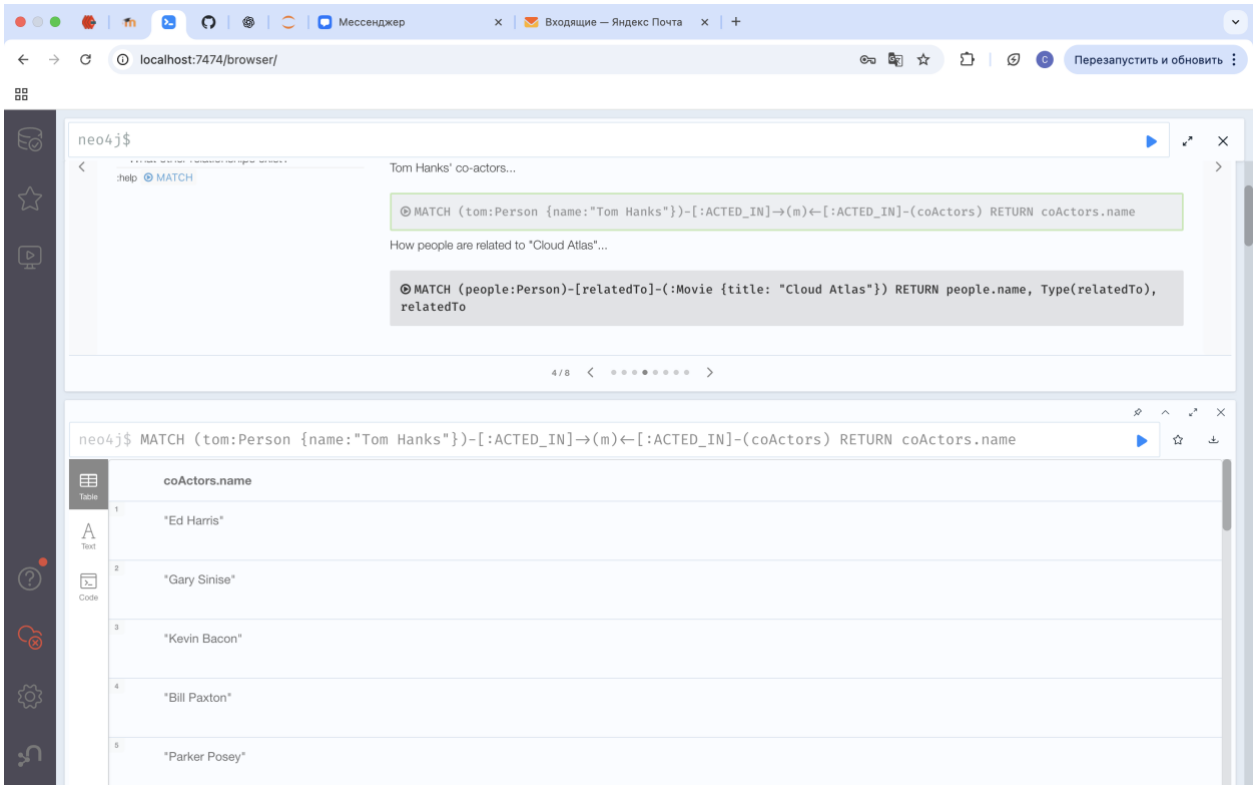


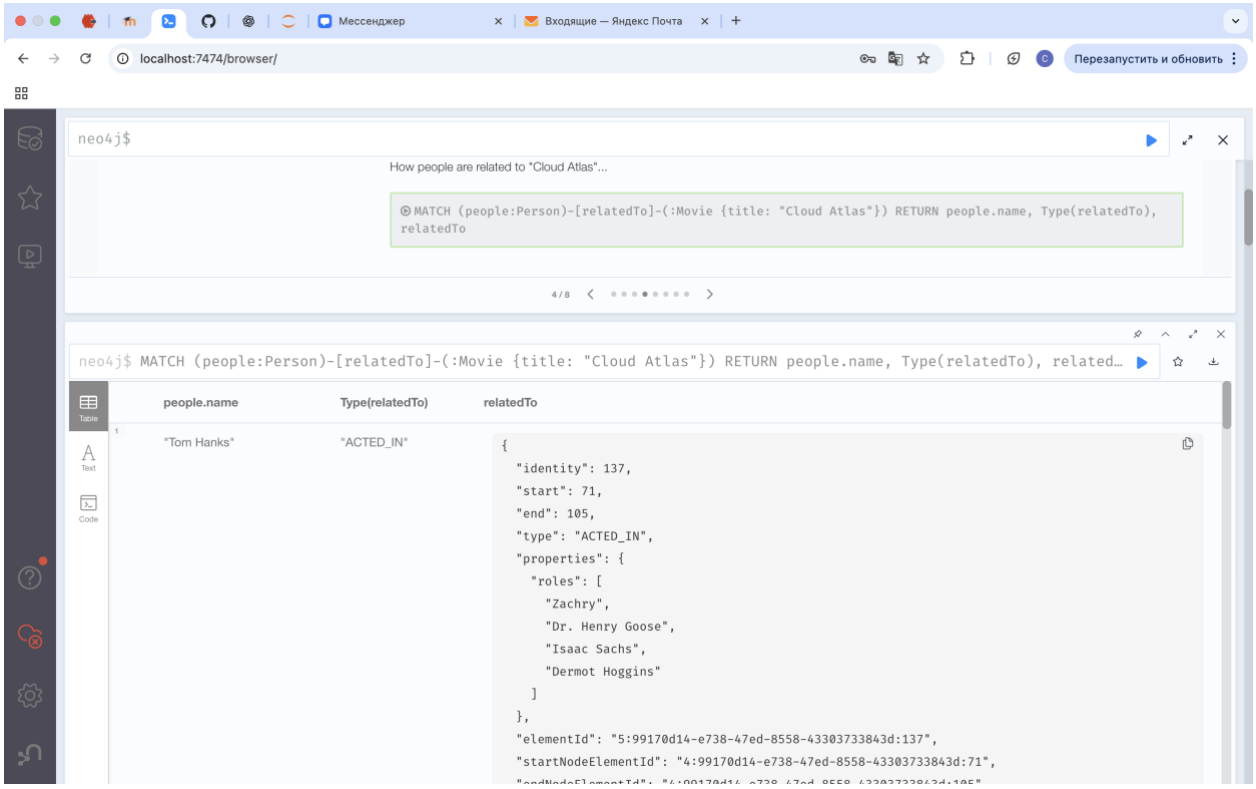


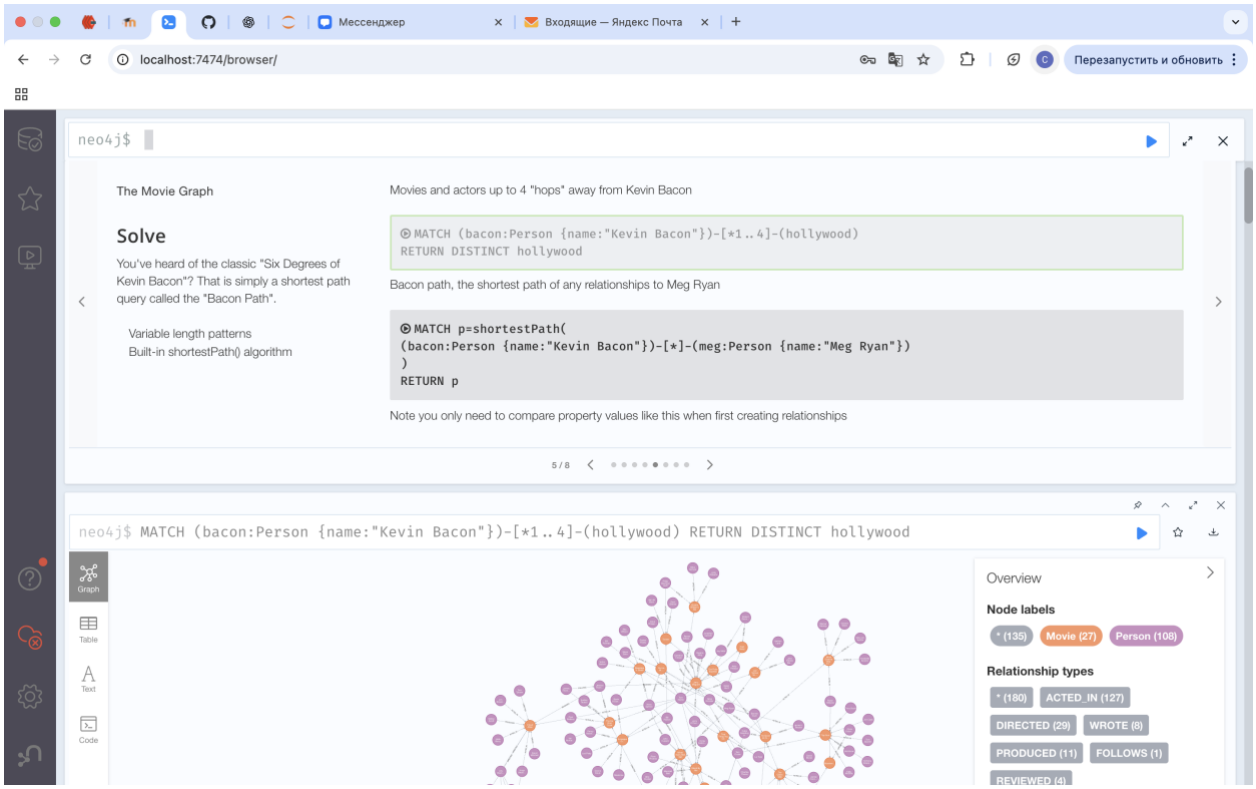

$:play movie graph
The Movie Graph
Query
Finding patterns within the graph.
Actors are people who acted in movies
Directors are people who directed a movie
What other relationships exist?
:help MATCH
List all Tom Hanks movies...
MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]→(tomHanksMovies) RETURN tom,tomHanksMovies
Who directed "Cloud Atlas"?
MATCH (cloudAtlas {title: "Cloud Atlas"})←[:DIRECTED]-(directors) RETURN directors.name
Tom Hanks' co-actors...
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors) RETURN coActors.name
How people are related to "Cloud Atlas"...
MATCH (people:Person)-[relatedTo]-(:Movie {title: "Cloud Atlas"}) RETURN people.name, Type(relatedTo), relatedTo
4 / 8
neo4j$ MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]→(tomHanksMovies) RETURN tom,tomHanksMovies
Graph
The Da Vinci Code
That Thing You
Overview
Node labels


Мессенджер
Входящие — Яндекс Почта
localhost:7474/browser/
Перезапустить и обновить
neo4j$
4 / 8
neo4j$ MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]→(tomHanksMovies) RETURN tom,tomHanksMovies
Graph
Table
Text
Code
The Da Vinci Code
That Thing You Do
Cloud Atlas
Cast Away
Joe Versus Vo...
The Green Mile
Tom Hanks
A League O...
Sleepless in Seattl
The Polar Express
You've M...
Apollo 13
ACTED_IN
DIRECTED
Use Cmd + scroll to zoom
Don't show again
Overview
Node labels
* (13)
Person (1)
Movie (12)
Relationship types
* (13)
ACTED_IN (12)
DIRECTED (1)
Displaying 13 nodes, 0 relationships.
neo4j$ MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]→(tomHanksMovies) RETURN tom,tomHanksMovies
Overview

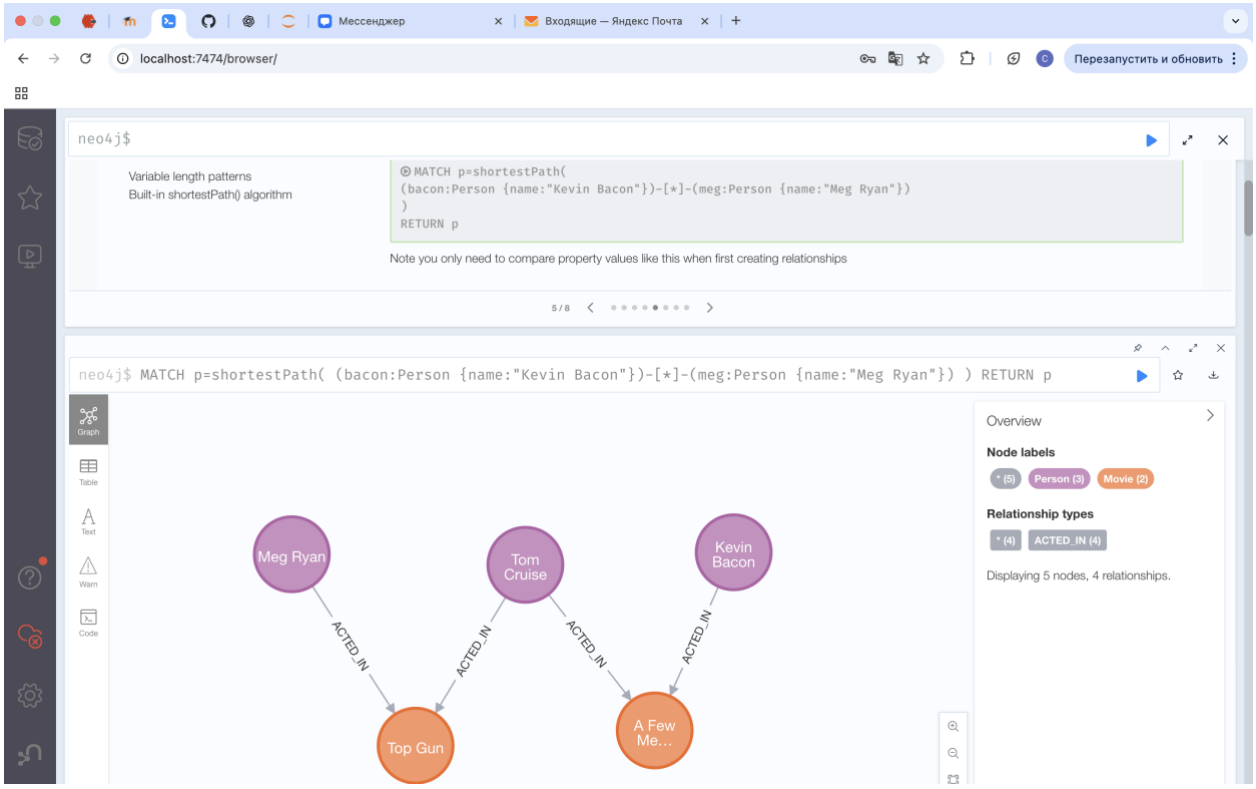

Мессенджер
Входящие — Яндекс Почта
localhost:7474/browser/
Перезапустить и обновить
neo4j$
Who directed "Cloud Atlas"?
MATCH (cloudAtlas {title: "Cloud Atlas"})←[:DIRECTED]-(directors) RETURN directors.name
Tom Hanks' co-actors...
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors) RETURN coActors.name
How people are related to "Cloud Atlas"...
MATCH (people:Person)-[relatedTo]-(:Movie {title: "Cloud Atlas"}) RETURN people.name, Type(relatedTo), relatedTo
4 / 8
neo4j$ MATCH (cloudAtlas {title: "Cloud Atlas"})←[:DIRECTED]-(directors) RETURN directors.name
Table
Text
Code
directors.name
"Tom Tykwer"
"Lilly Wachowski"
"Lana Wachowski"

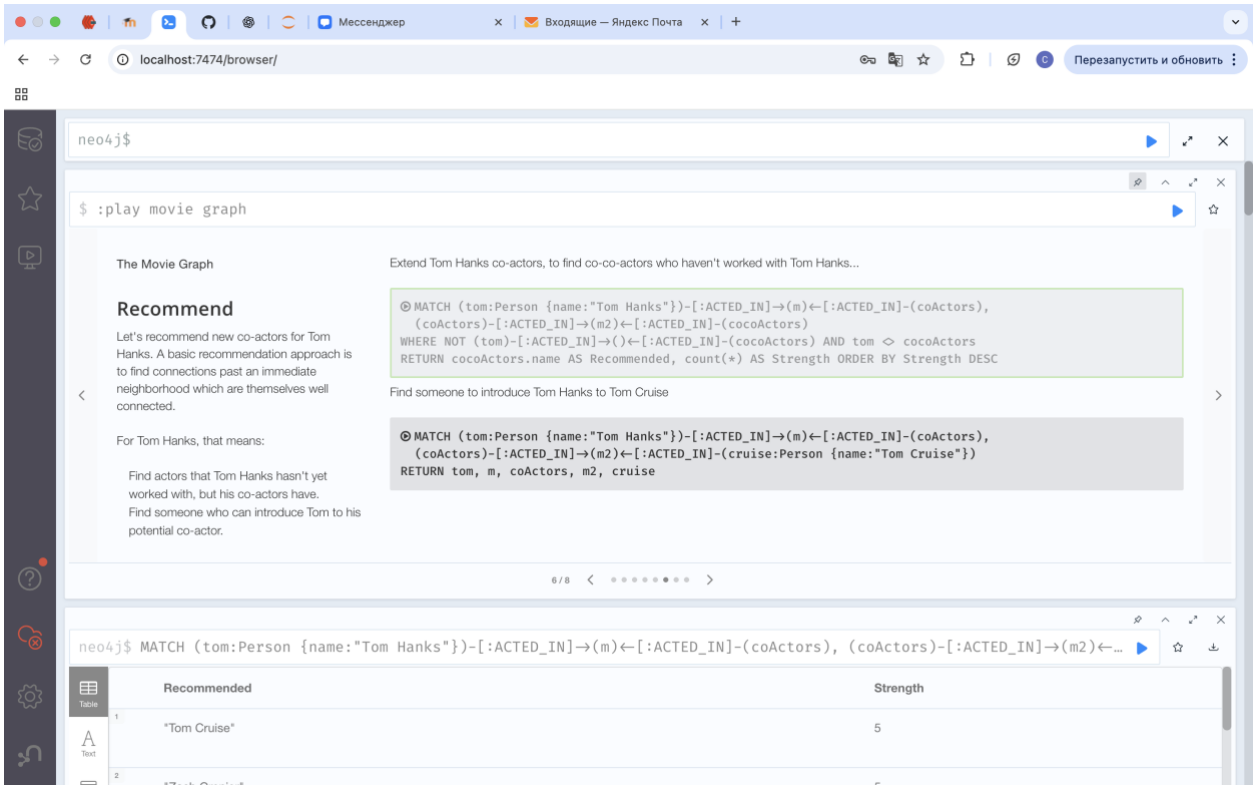

Мессенджер
Входящие — Яндекс Почта
localhost:7474/browser/
Перезапустить и обновить
neo4j$
Tom Hanks' co-actors...
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors) RETURN coActors.name
How people are related to "Cloud Atlas"...
MATCH (people:Person)-[relatedTo]-(:Movie {title: "Cloud Atlas"}) RETURN people.name, Type(relatedTo), relatedTo
4 / 8
neo4j$ MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors) RETURN coActors.name
Table
Text
Code
coActors.name
"Ed Harris"
"Gary Sinise"
"Kevin Bacon"
"Bill Paxton"
"Parker Posey"

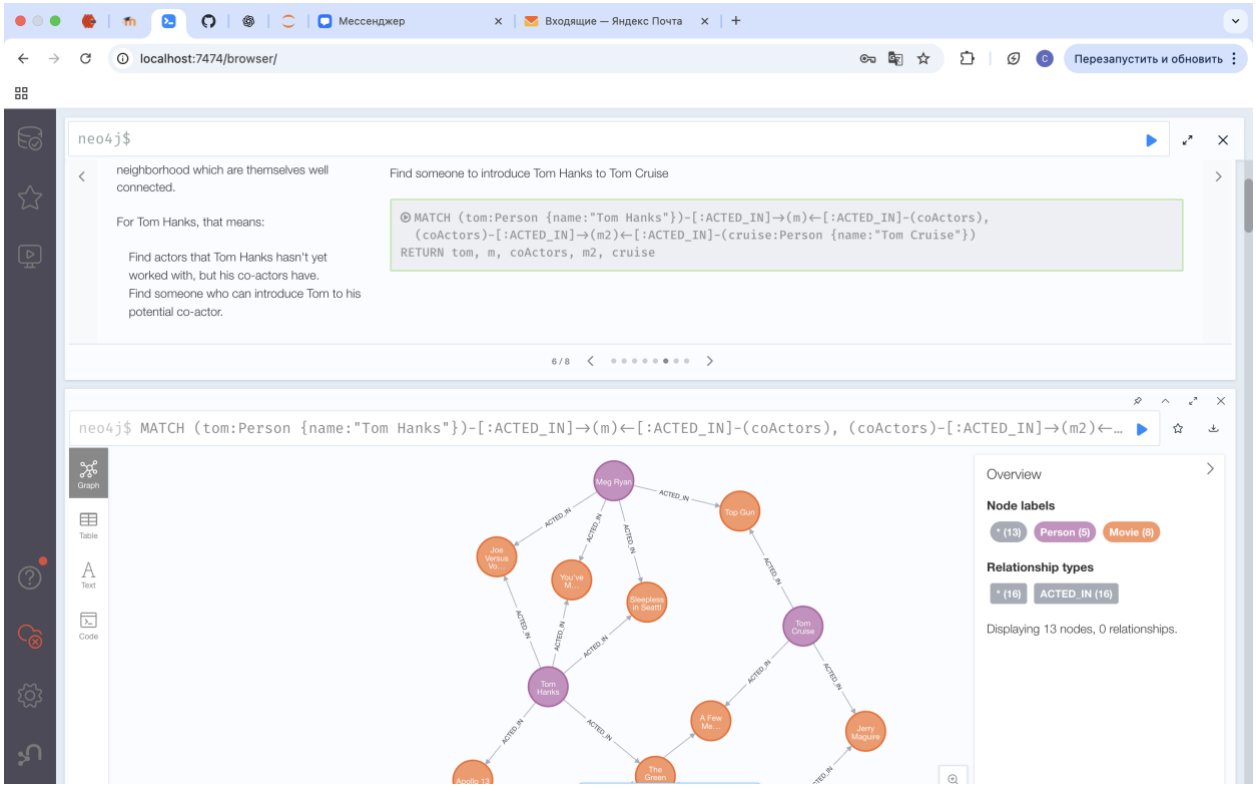


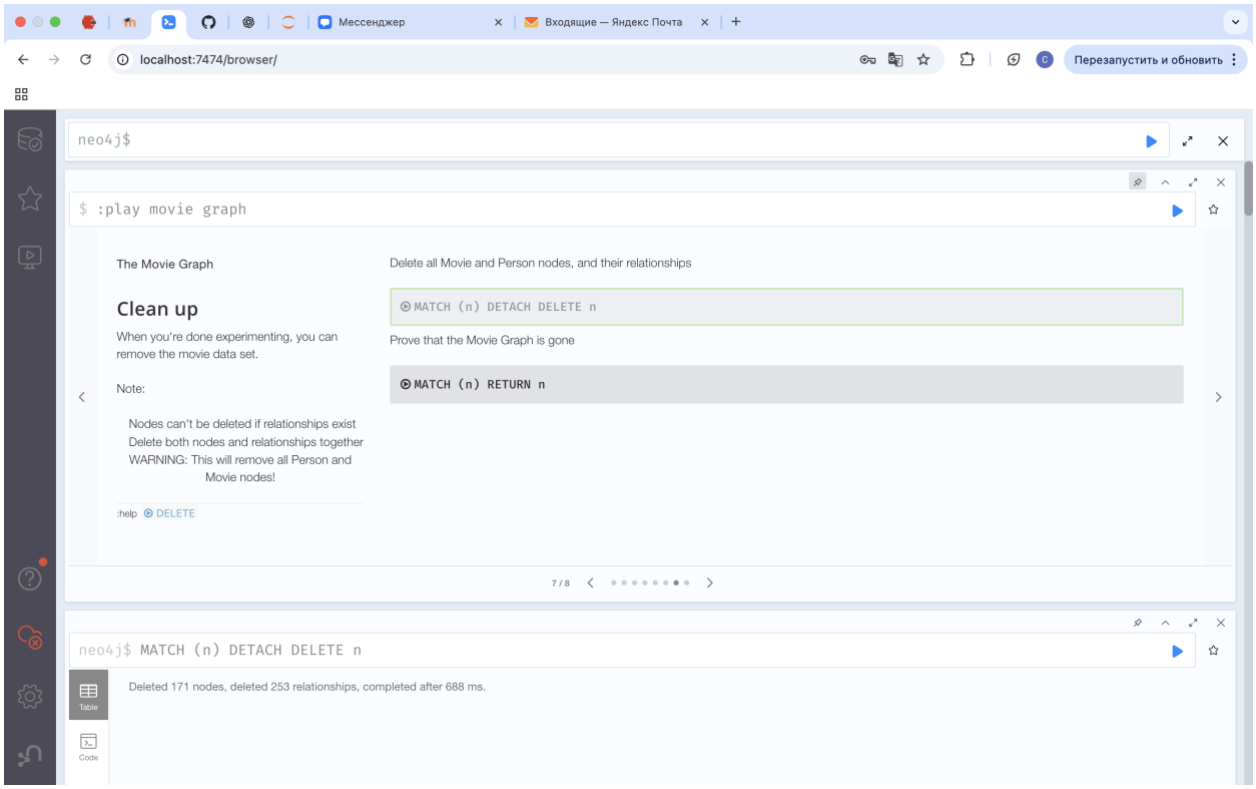


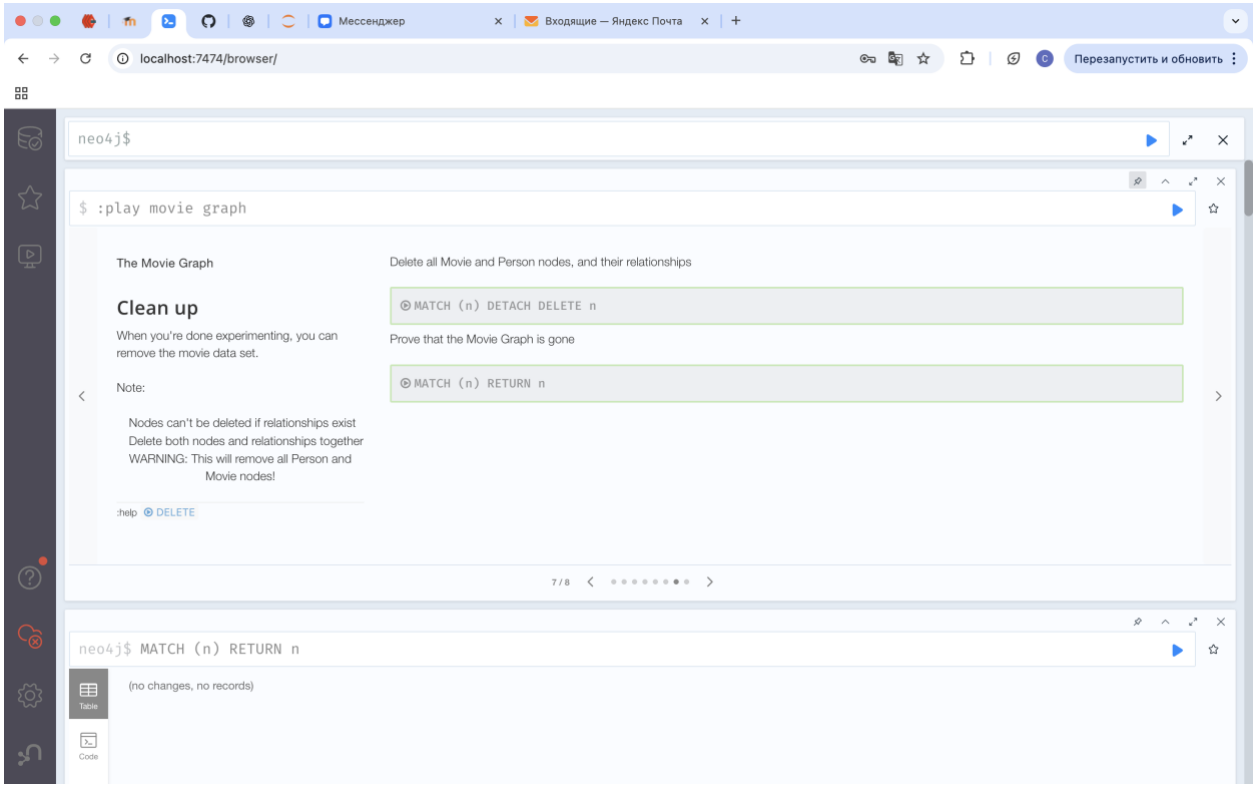
Найдите "Бейконовскую цепочку" — самый короткий путь через любые отношения к Meg Ryan.
Результат покажет самый короткий путь от Kevin Bacon до Meg Ryan.


Мессенджер
Входящие — Яндекс Почта
localhost:7474/browser/
Перезапустить и обновить
neo4j$
Variable length patterns
Built-in shortestPath() algorithm
MATCH p=shortestPath(
(bacon:Person {name:"Kevin Bacon"})-[*]-(meg:Person {name:"Meg Ryan"})
)
RETURN p
Note you only need to compare property values like this when first creating relationships
5 / 8
neo4j$ MATCH p=shortestPath((bacon:Person {name:"Kevin Bacon"})-[*]-(meg:Person {name:"Meg Ryan"})) RETURN p
Graph
Table
Text
Warn
Code
Meg Ryan
Tom Cruise
Kevin Bacon
Top Gun
A Few Me…
ACTED_IN
ACTED_IN
ACTED_IN
ACTED_IN
Overview
Node labels
* (5)
Person (3)
Movie (2)
Relationship types
* (4)
ACTED_IN (4)
Displaying 5 nodes, 4 relationships.


Мессенджер
Входящие — Яндекс Почта
localhost:7474/browser/
Перезапустить и обновить
neo4j$
$:play movie graph
The Movie Graph
Recommend
Let's recommend new co-actors for Tom Hanks. A basic recommendation approach is to find connections past an immediate neighborhood which are themselves well connected.
For Tom Hanks, that means:
Find actors that Tom Hanks hasn't yet worked with, but his co-actors have.
Find someone who can introduce Tom to his potential co-actor.
Extend Tom Hanks co-actors, to find co-co-actors who haven't worked with Tom Hanks...
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors),
(coActors)-[:ACTED_IN]→(m2)←[:ACTED_IN]-(cocoActors)
WHERE NOT (tom)-[:ACTED_IN]→()←[:ACTED_IN]-(cocoActors) AND tom <> cocoActors
RETURN cocoActors.name AS Recommended, count(*) AS Strength ORDER BY Strength DESC
Find someone to introduce Tom Hanks to Tom Cruise
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors),
(coActors)-[:ACTED_IN]→(m2)←[:ACTED_IN]-(cruise:Person {name:"Tom Cruise"})
RETURN tom, m, coActors, m2, cruise
6 / 8
neo4j$ MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors), (coActors)-[:ACTED_IN]→(m2)←…
Table
Text
Recommended
Strength
1
"Tom Cruise"
5


Мессенджер
Входящие — Яндекс Почта
localhost:7474/browser/
Перезапустить и обновить
neo4j$
neighborhood which are themselves well connected.
For Tom Hanks, that means:
Find actors that Tom Hanks hasn't yet worked with, but his co-actors have.
Find someone who can introduce Tom to his potential co-actor.
Find someone to introduce Tom Hanks to Tom Cruise
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors),
(coActors)-[:ACTED_IN]→(m2)←[:ACTED_IN]-(cruise:Person {name:"Tom Cruise"})
RETURN tom, m, coActors, m2, cruise
6 / 8
neo4j$ MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors), (coActors)-[:ACTED_IN]→(m2)←…
Graph
Table
Text
Code
Meg Ryan
Top Gun
Joe Versus Vo…
You've M…
Sleepless in Seattl…
Tom Cruise
Tom Hanks
A Few Me…
Jerry Maguire
The Green
Overview
Node labels
* (13)
Person (5)
Movie (8)
Relationship types
* (16)
ACTED_IN (16)
Displaying 13 nodes, 0 relationships.

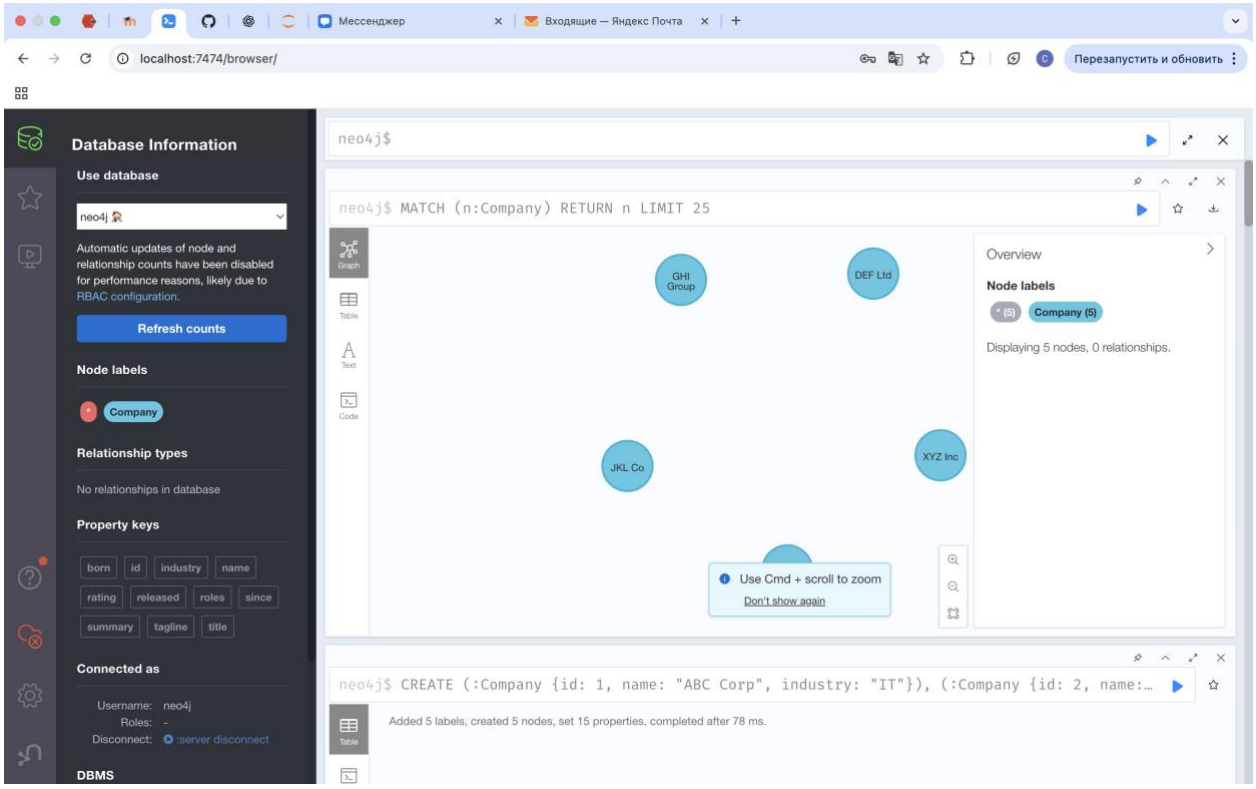

Мессенджер
Входящие — Яндекс Почта
localhost:7474/browser/
Перезапустить и обновить
neo4j$
$:play movie graph
The Movie Graph
Clean up
When you're done experimenting, you can remove the movie data set.
Note:
Nodes can't be deleted if relationships exist
Delete both nodes and relationships together
WARNING: This will remove all Person and Movie nodes!
:help DELETE
Delete all Movie and Person nodes, and their relationships
MATCH (n) DETACH DELETE n
Prove that the Movie Graph is gone
MATCH (n) RETURN n
7 / 8
neo4j$ MATCH (n) DETACH DELETE n
Table
Code
Deleted 171 nodes, deleted 253 relationships, completed after 688 ms.



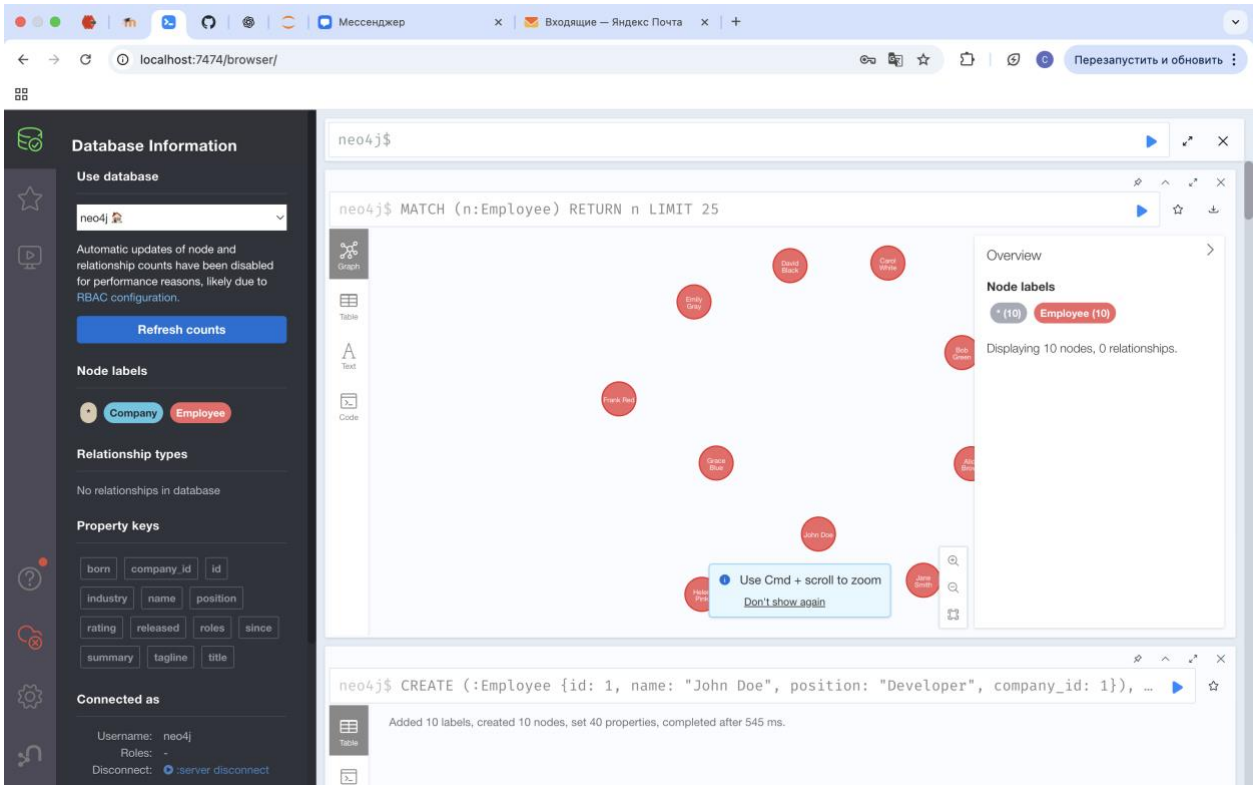
ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Загрузка данных

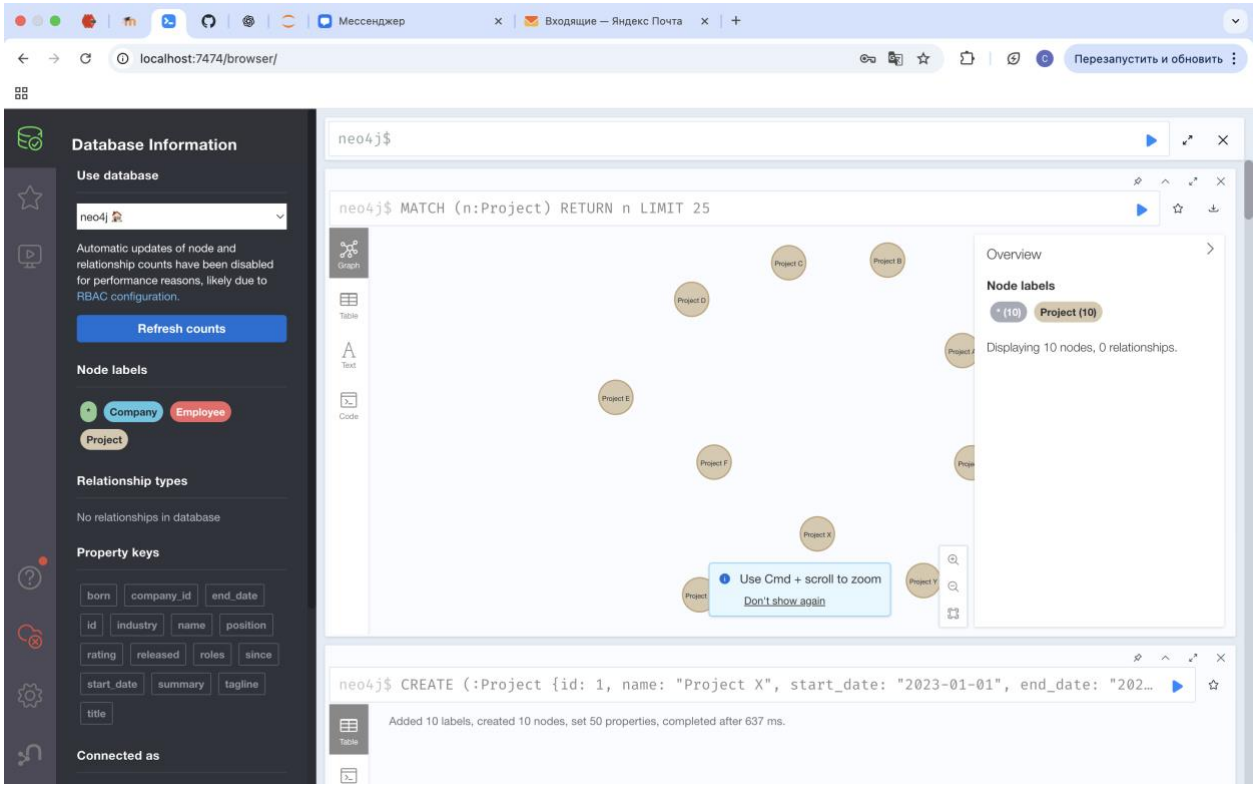
Создание компаний



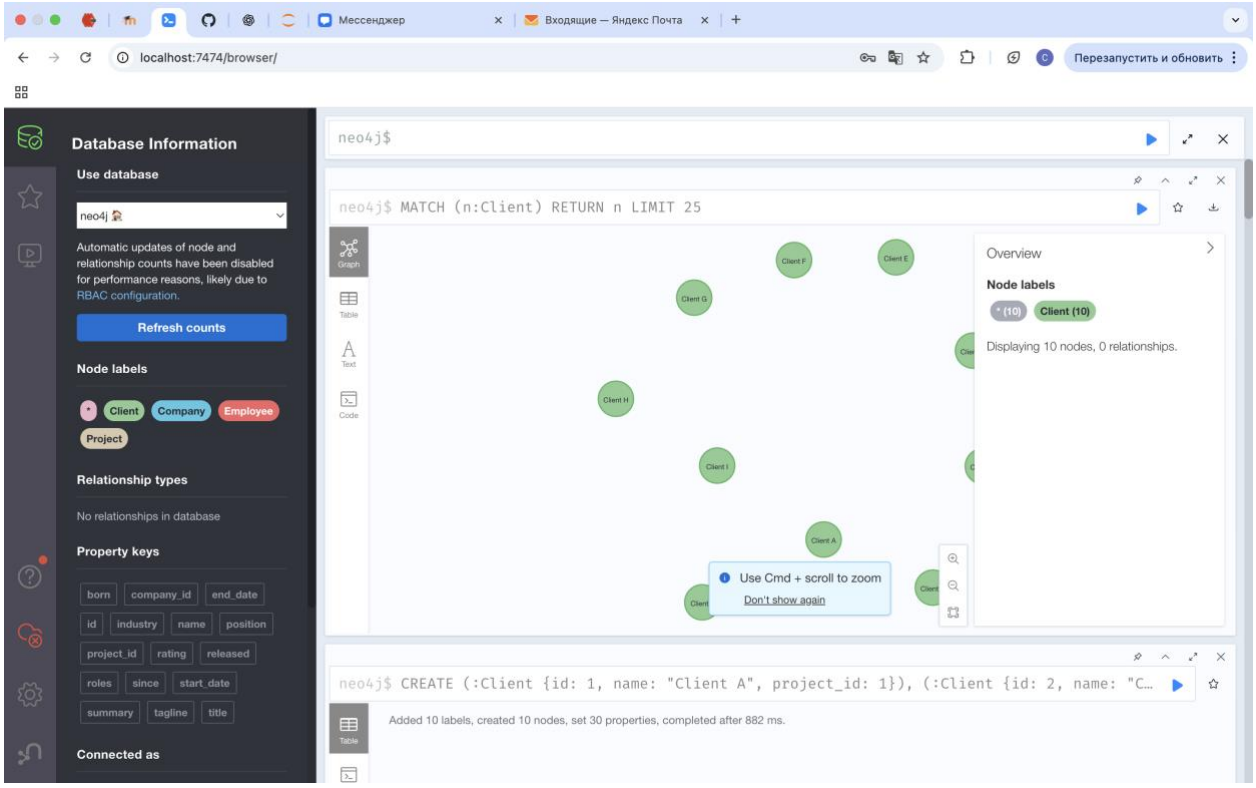
Создание сотрудников



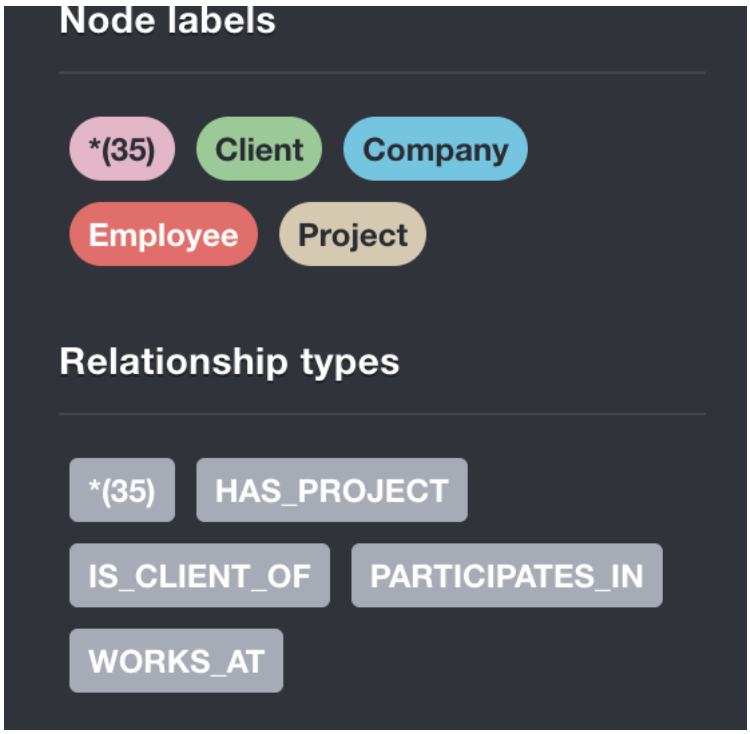
Создание проектов



Создание клиентов



Создание отношений



Задание 1 Найдите все компании, имеющие более 6 проектов.

Задание 2 Найдите всех сотрудников, занимающих должность "Designer".

```
MATCH (e:Employee)
WHERE e.position = "Designer"
RETURN e.id AS EmployeeID, e.name AS Name, e.company_id AS CompanyID
ORDER BY e.id;
```

	EmployeeID	Name	CompanyID
1	5	"Carol White"	5
2	10	"Helen Pink"	5

Started streaming 2 records after 984 ms and completed after 1017 ms.

Задание 3 Найдите все проекты, длительностью более 8 месяцев.

```
MATCH (p:Project)
WITH p, duration.between(date(p.start_date), date(p.end_date)) AS dur
WHERE dur.months + dur.years * 12 > 8
RETURN p.id AS ProjectID, p.name AS ProjectName, dur AS Duration
ORDER BY Duration DESC;
```

(no changes, no records)

Completed after 568 ms.

Задание 4 Найдите всех клиентов, связанных с проектами компании "GHI Group".

```
MATCH (company:Company {name: "GHI Group"})-[:HAS_PROJECT]→(p:Project)←
[:IS_CLIENT_OF]-(client:Client)
RETURN DISTINCT client.id AS ClientID, client.name AS ClientName, p.id AS
ProjectID, p.name AS ProjectName
ORDER BY client.id;
```

	ClientID	ClientName	ProjectID	ProjectName
1	4	"Client D"	4	"Project A"
2	9	"Client I"	9	"Project F"

Started streaming 2 records after 1519 ms and completed after 1664 ms.

Задание 5 Найдите все роли сотрудника "Carol White" в проектах.

```
MATCH (e:Employee {name: "Carol White"})-[r:PARTICIPATES_IN]→(p:Project)
RETURN p.id AS ProjectID, p.name AS ProjectName, r.role AS Role
ORDER BY p.id;
```

	ProjectID	ProjectName	Role
1	5	"Project B"	"UX Designer"

Started streaming 1 records after 1727 ms and completed after 1765 ms.

Выводы

В ходе выполнения лабораторной работы были изучены основные концепции и операции для трех типов NoSQL баз данных:

- MongoDB — создание коллекций и документов, выполнение запросов с фильтрацией, обновление и удаление данных, использование индексов для оптимизации.
- Redis — работа с хэшами, добавление и выборка данных по ключам, применение команд для управления значениями.
- Neo4j — работа с графовой моделью, создание узлов и связей, выполнение запросов на языке Cypher.

В результате были получены практические навыки по созданию, наполнению и запросу данных в различных NoSQL системах, а также понимание различий между моделями данных и областями их применения.