Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики управления и технологий

Мошенина Елена Дмитриевна БД-241м

Инструменты хранения и анализа больших данных

Практическая работа 2.1. Изучение методов хранения данных на основе NoSQL. Рекомендательные системы. GraphDB.

Направление подготовки/специальность
38.04.05 - Бизнес-информатика
Бизнес-аналитика и большие данные
(очная форма обучения)

Руководитель дисциплины:
Босенко Т.М., доцент департамента
информатики, управления и технологий,
доктор экономических наук

Москва
2025

Содержание

Введение **3**

Основная часть **5**

Заключение **25**

Введение

Цель занятия:

Цель данного занятия — освоение работы с базой данных GraphDB в виртуальной машине с использованием Docker. Студенты научатся загружать RDF-данные в базу, выполнять запросы с использованием SPARQL и анализировать результаты, используя функциональность GraphDB.

Задачи:

- Настроить и запустить контейнер с GraphDB с помощью Docker.
- Загрузить RDF-данные о фильмах в базу данных.
- Ознакомиться с основами SPARQL-запросов.
- Выполнить различные SPARQL-запросы для получения информации из базы данных.
- Проанализировать и интерпретировать результаты выполнения запросов.

Необходимое ПО:

Операционная система: Ubuntu 22.

СУБД: GraphDB.

Docker: для запуска контейнера с GraphDB.

Среда разработки: SPARQL редактор для выполнения запросов.

Процесс начала работы:

Подключитесь к виртуальной машине и войдите в нее.

Перейдите в каталог с проектом:

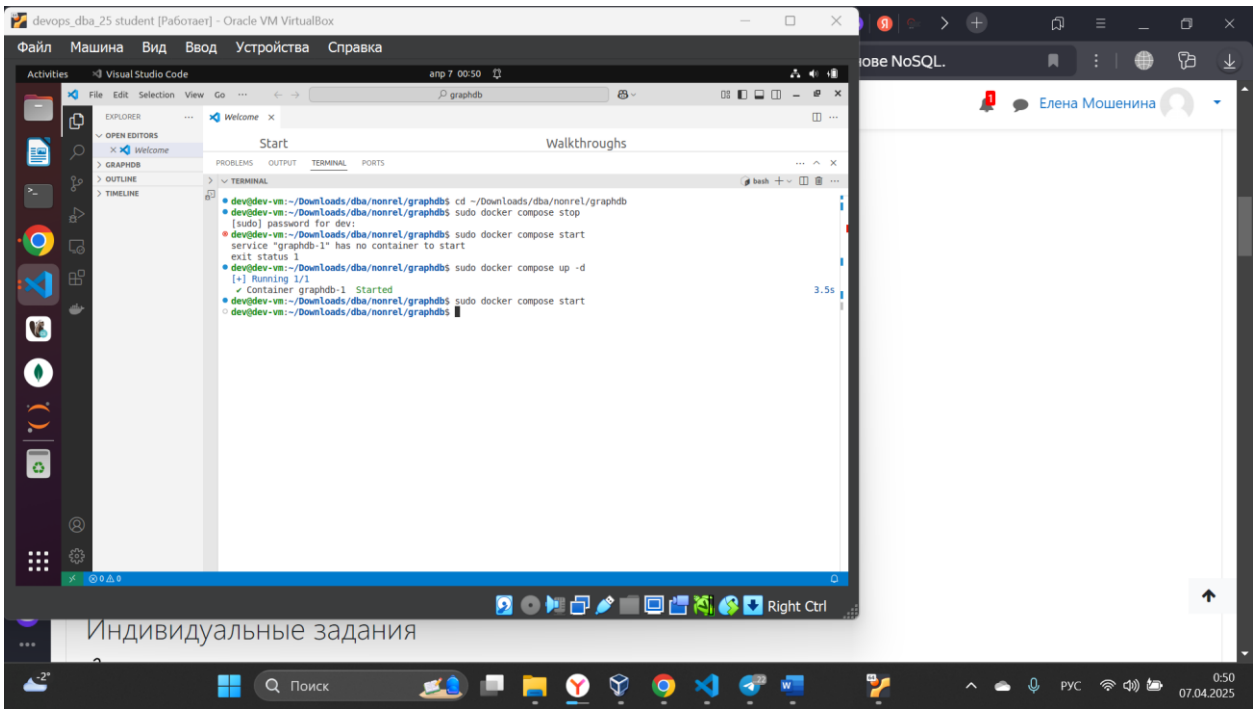
cd ~/Downloads/dba/nonrel/graphdb

Остановите контейнер, если он работает:

sudo docker compose stop

Запустите контейнер с базой данных GraphDB:

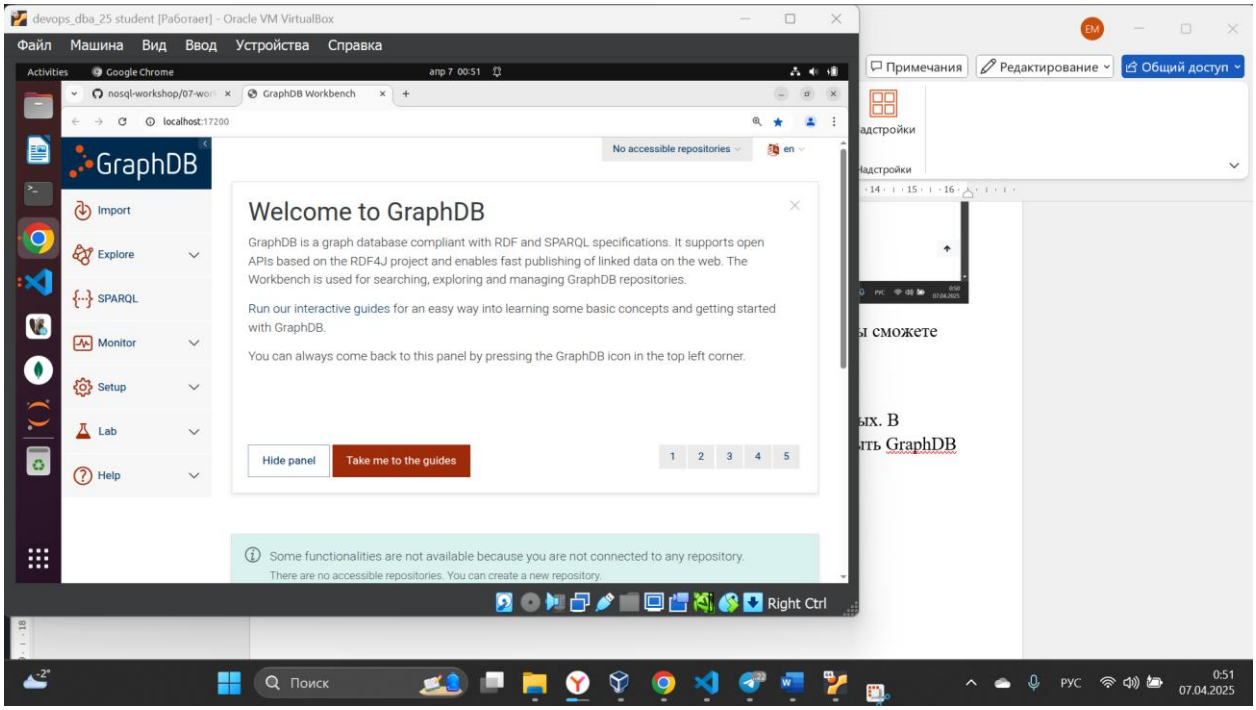
sudo docker compose start



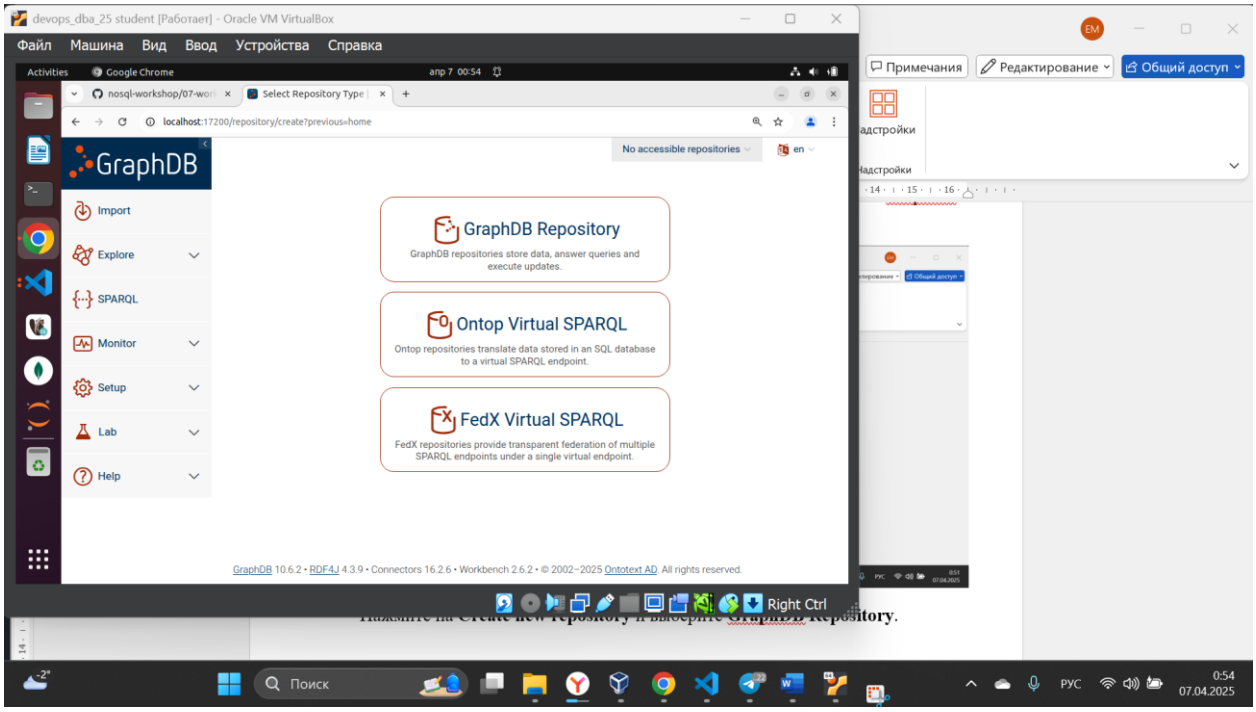
После этого база данных GraphDB будет готова для работы, и вы сможете приступить к выполнению заданий.

Основная часть

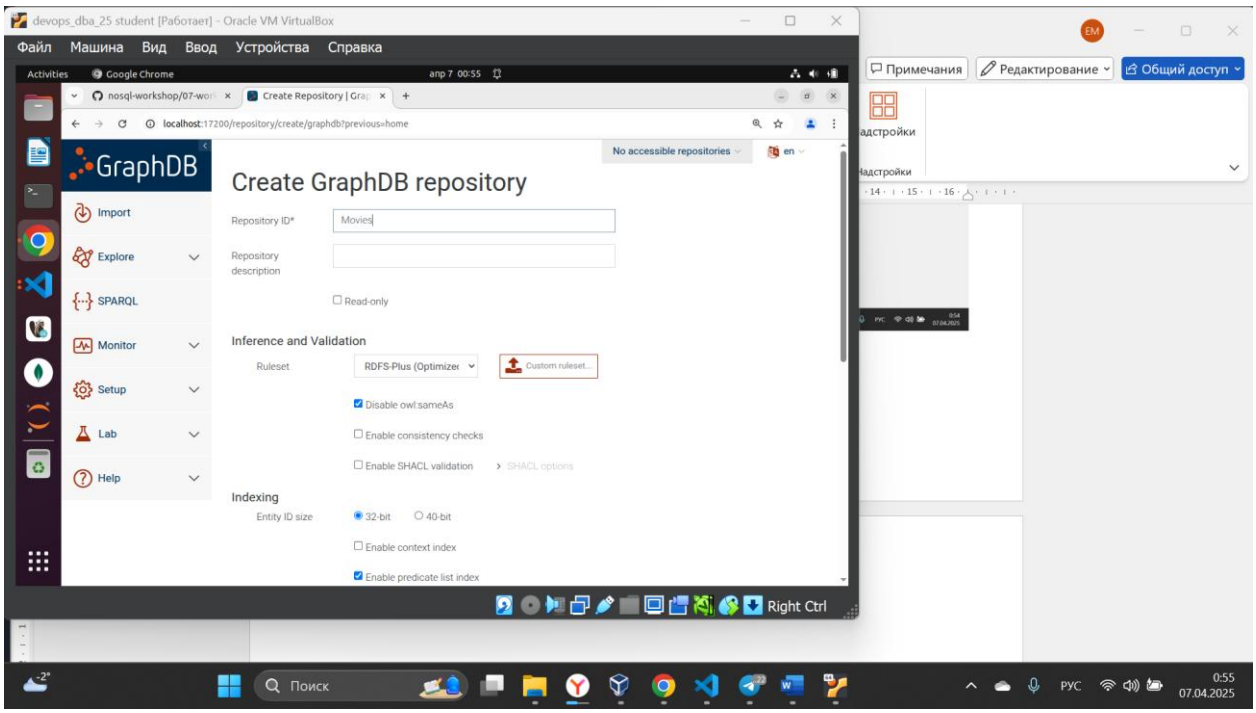
Мы будем использовать GraphDB Workbench для загрузки данных. В браузере перейдите по адресу http://localhost:17200, чтобы открыть GraphDB Workbench.



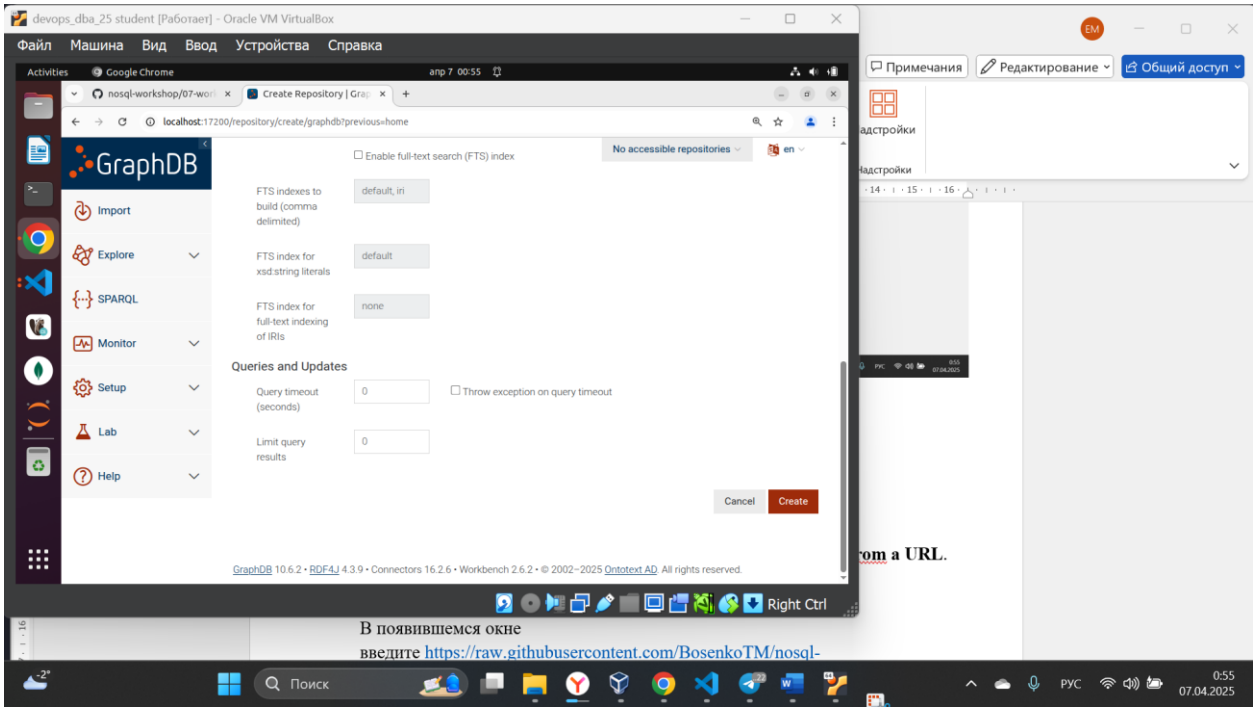
Нажмите на **Create new repository** и выберите **GraphDB Repository**.



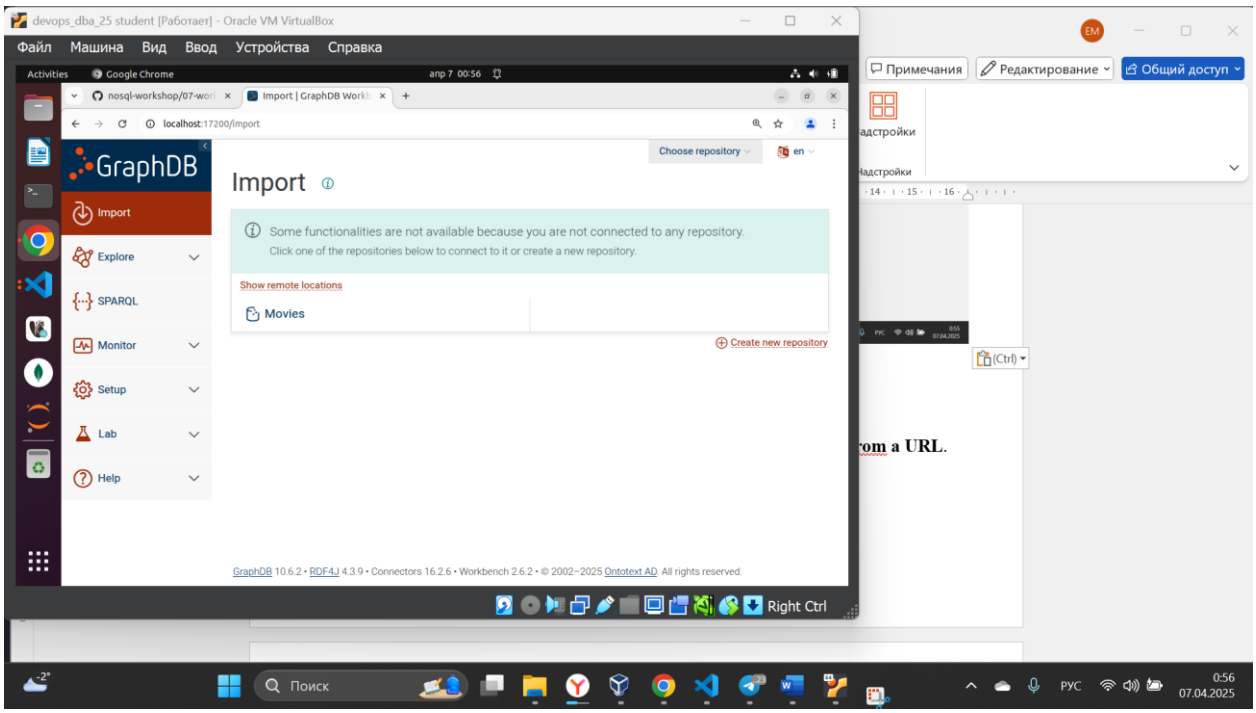
Введите Movies в поле **Repository**



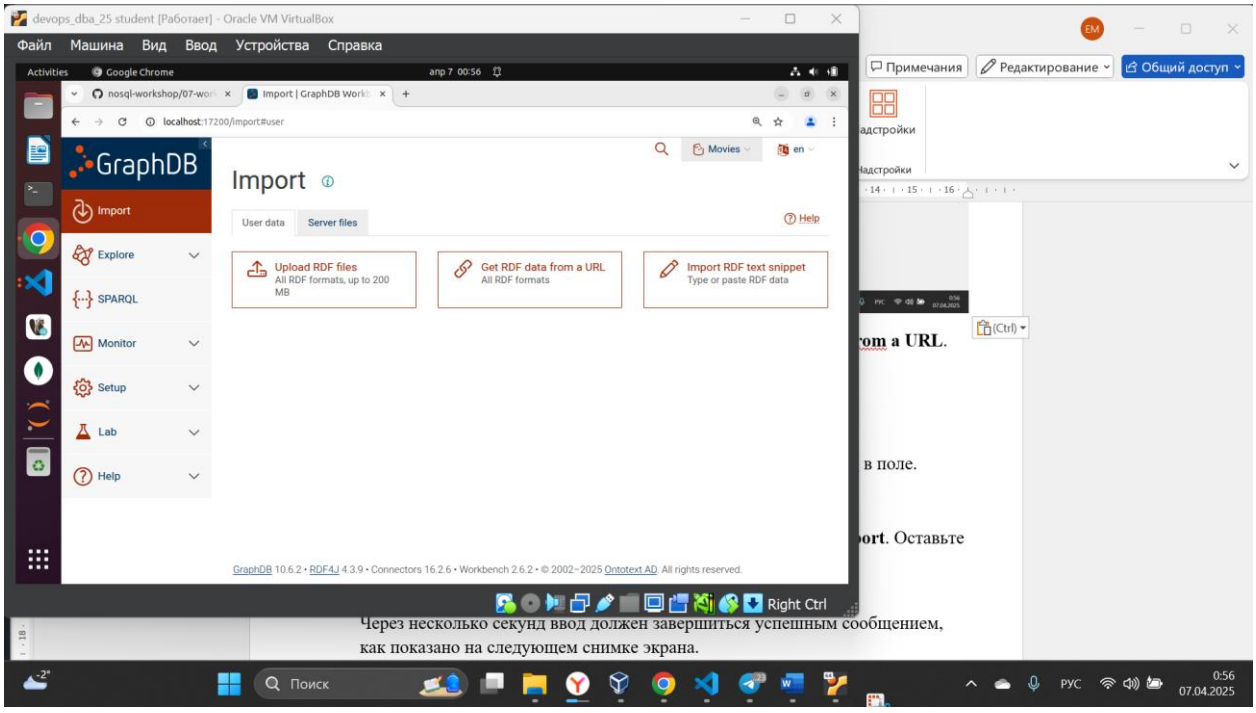
и нажмите **Create**.



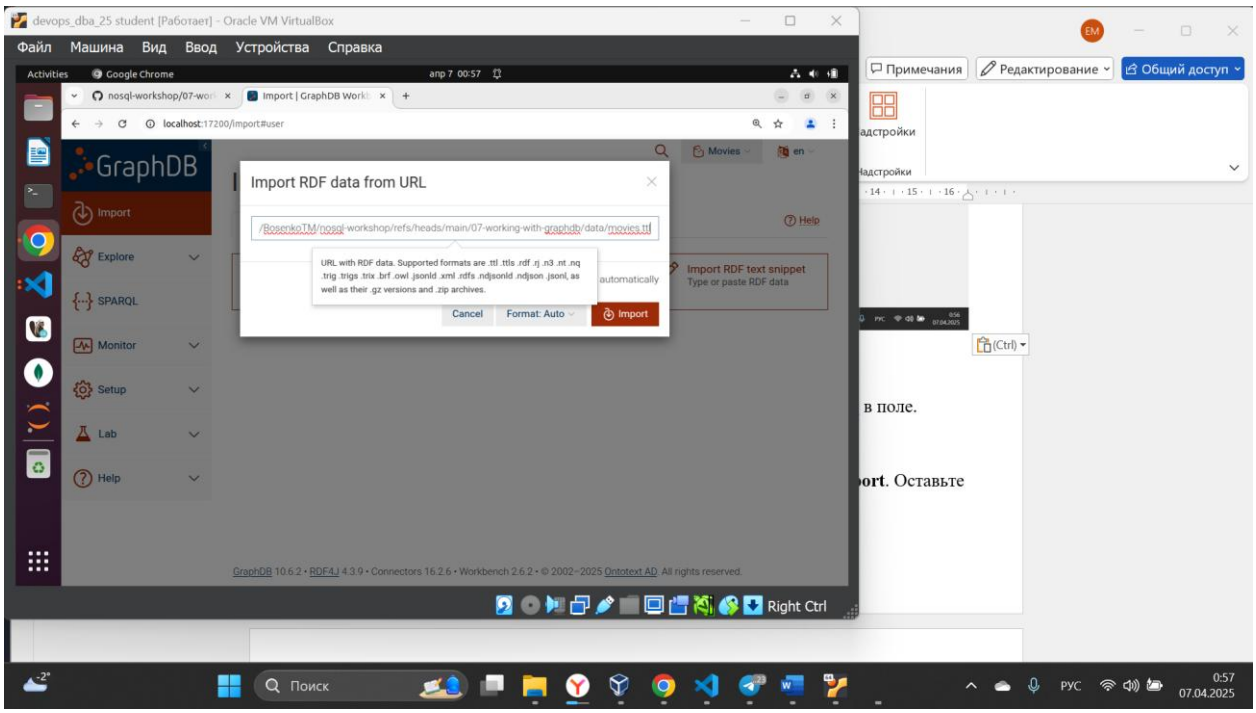
В меню слева нажмите на **Import**.



Нажмите на **Movies** и выберите второй вариант **Get RDF data from a URL**.

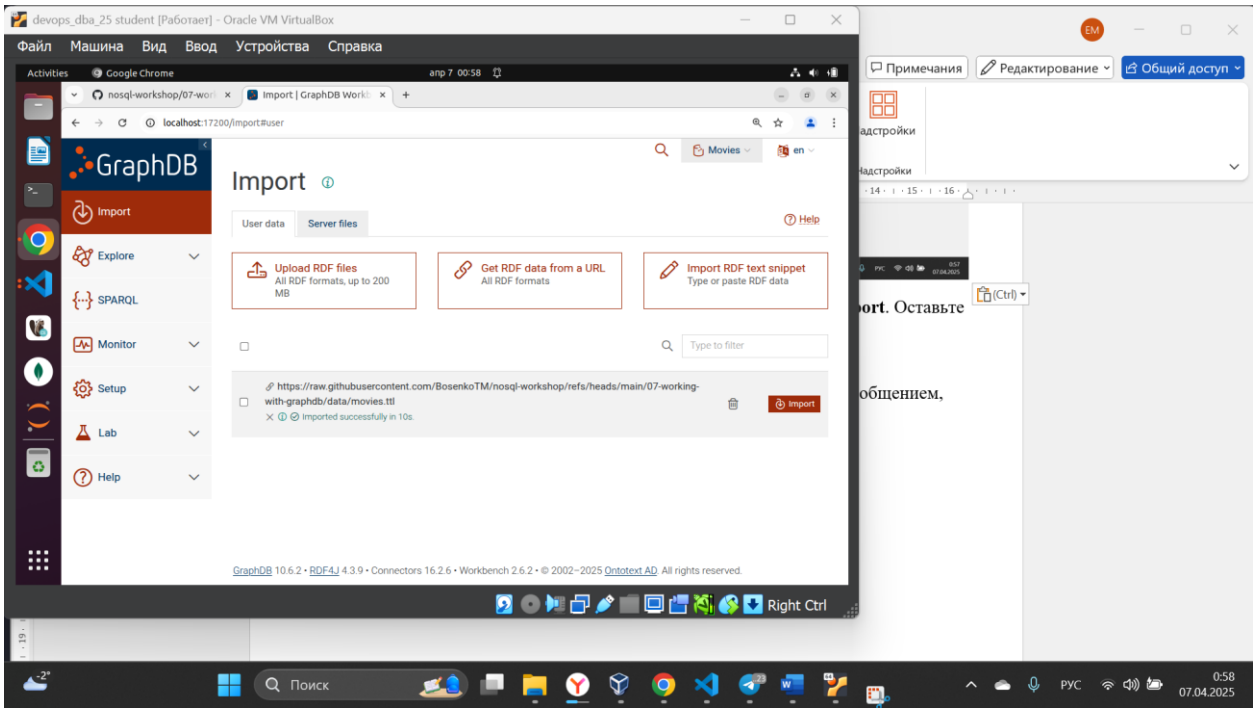


В появившемся окне
введите https://raw.githubusercontent.com/BosenkoTM/nosql-workshop/refs/heads/main/07-working-with-graphdb/data/movies.ttl в поле.



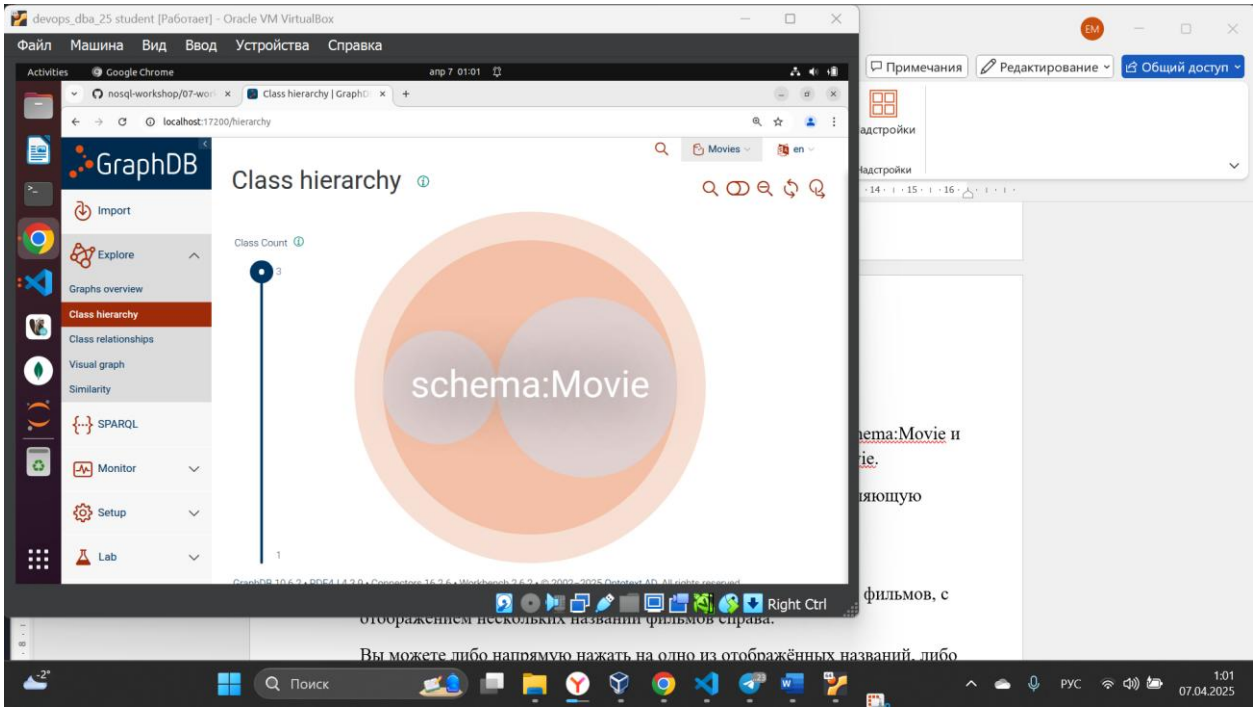
Оставьте выбранным **Start import automatically** и нажмите **Import**. Оставьте значения по умолчанию в другом всплывающем окне и снова нажмите **Import**.

Через несколько секунд ввод должен завершиться успешным сообщением, как показано на следующем снимке экрана.



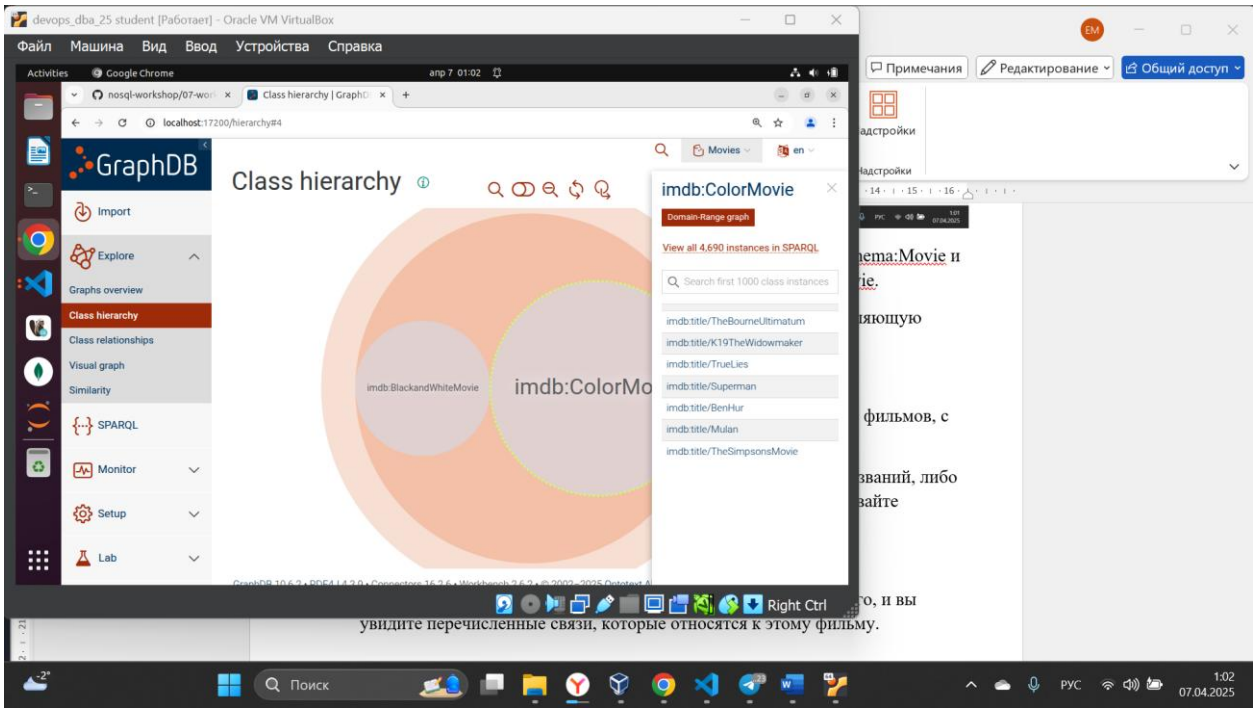
Теперь база данных готова к использованию.

Нажмите на **Explore** и **Class hierarchy**.



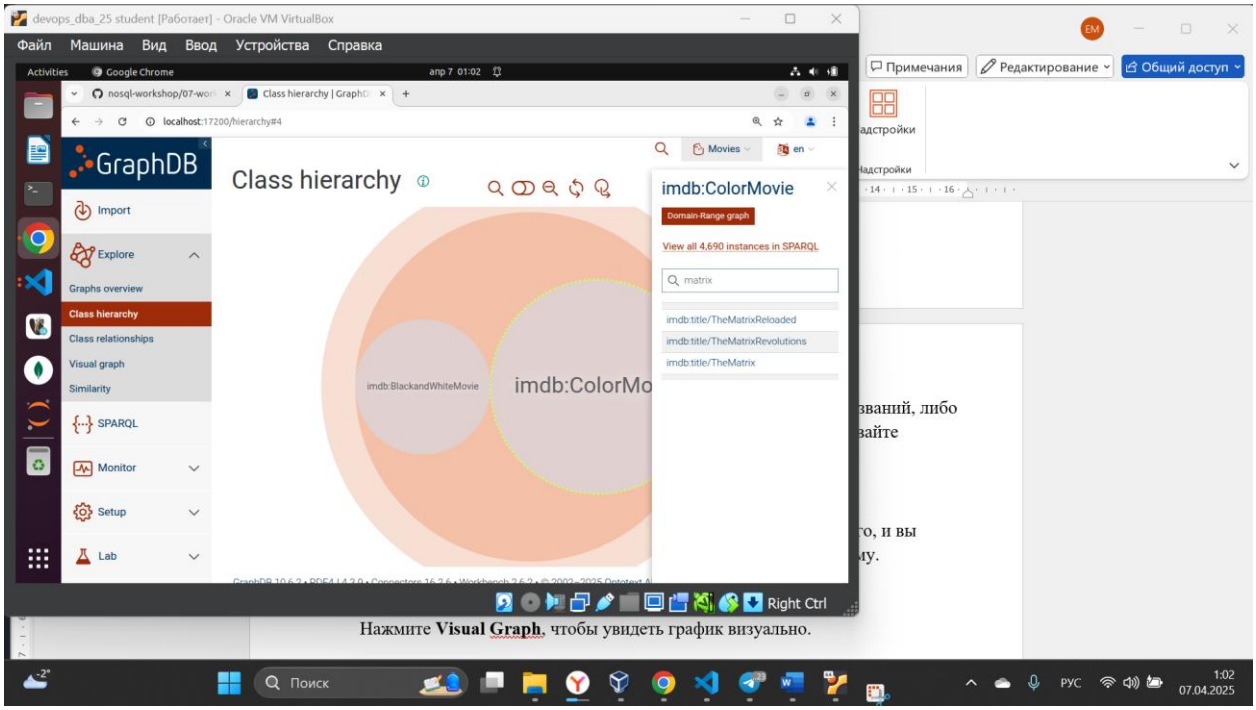
Мы можем увидеть схему графа фильмов с базовым классом schema:Movie и двумя подклассами imbd:BlackAndWhiteMovie и imdb:ColorMovie.

Нажмите на более крупную внутреннюю окружность, представляющую цветные фильмы.

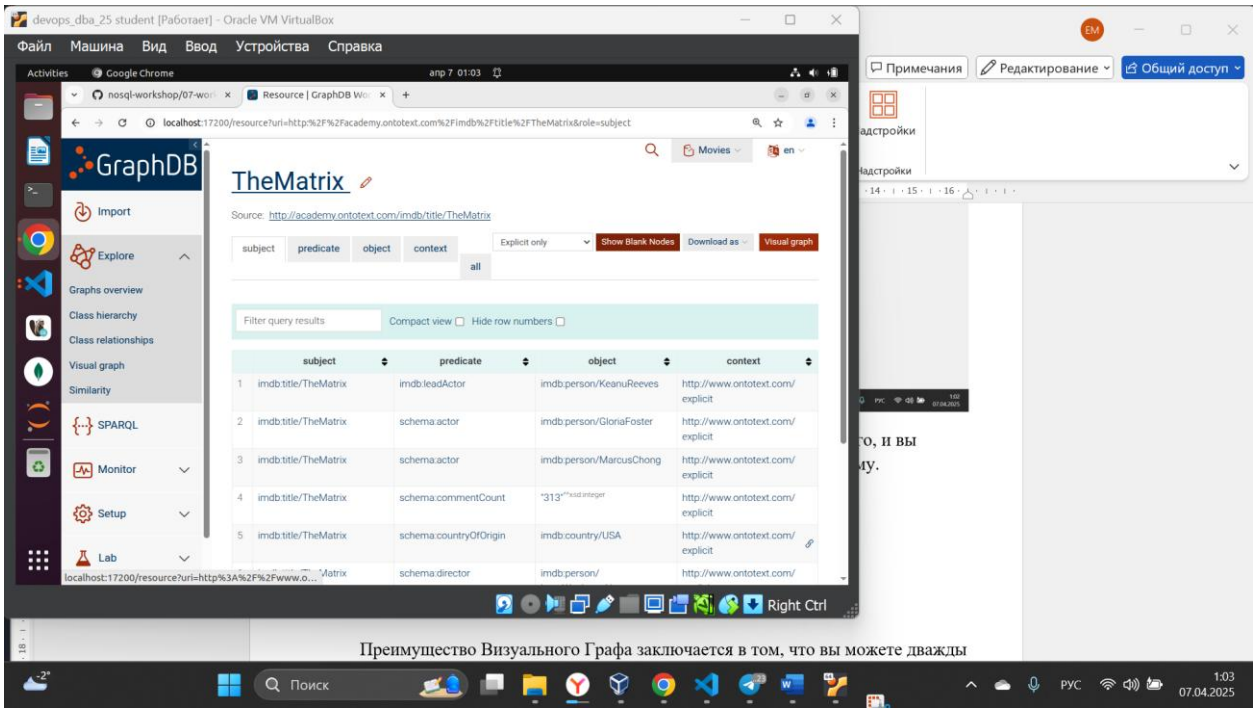


Мы можем увидеть, что в графе есть 4690 экземпляров цветных фильмов, с отображением нескольких названий фильмов справа.

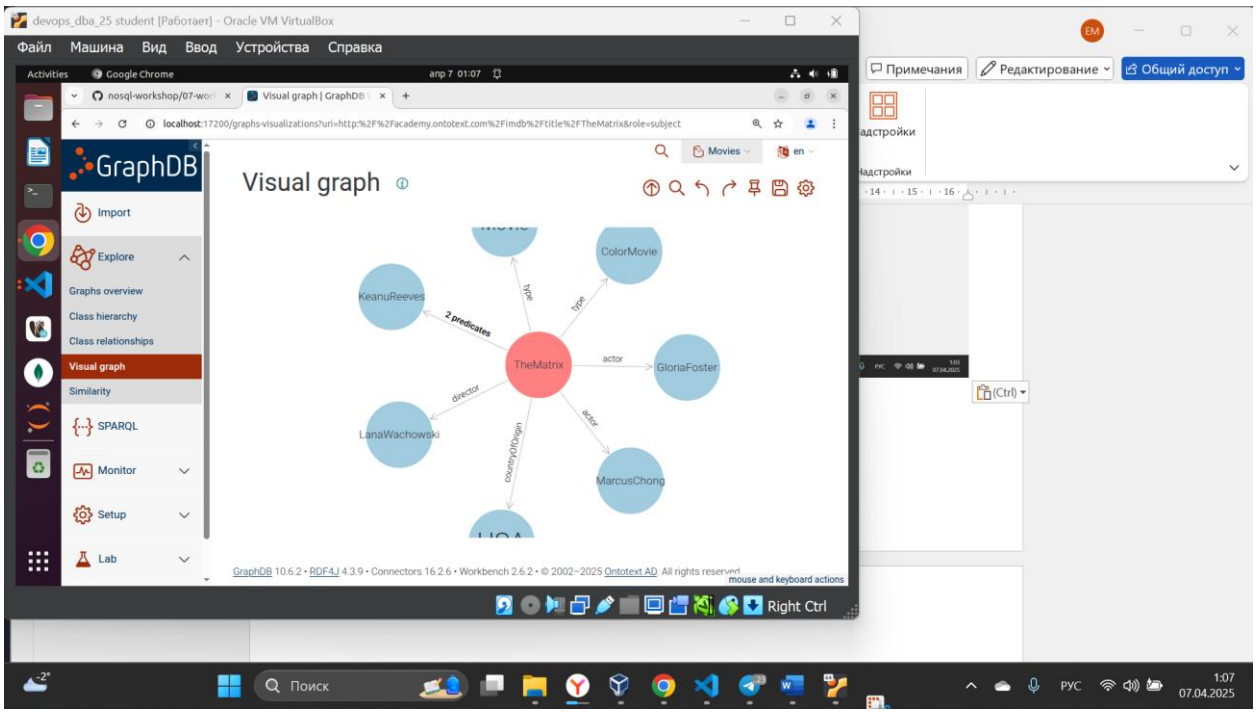
Вы можете либо напрямую нажать на одно из отображённых названий, либо использовать поиск для нахождения определённого фильма. Давайте введём matrix.



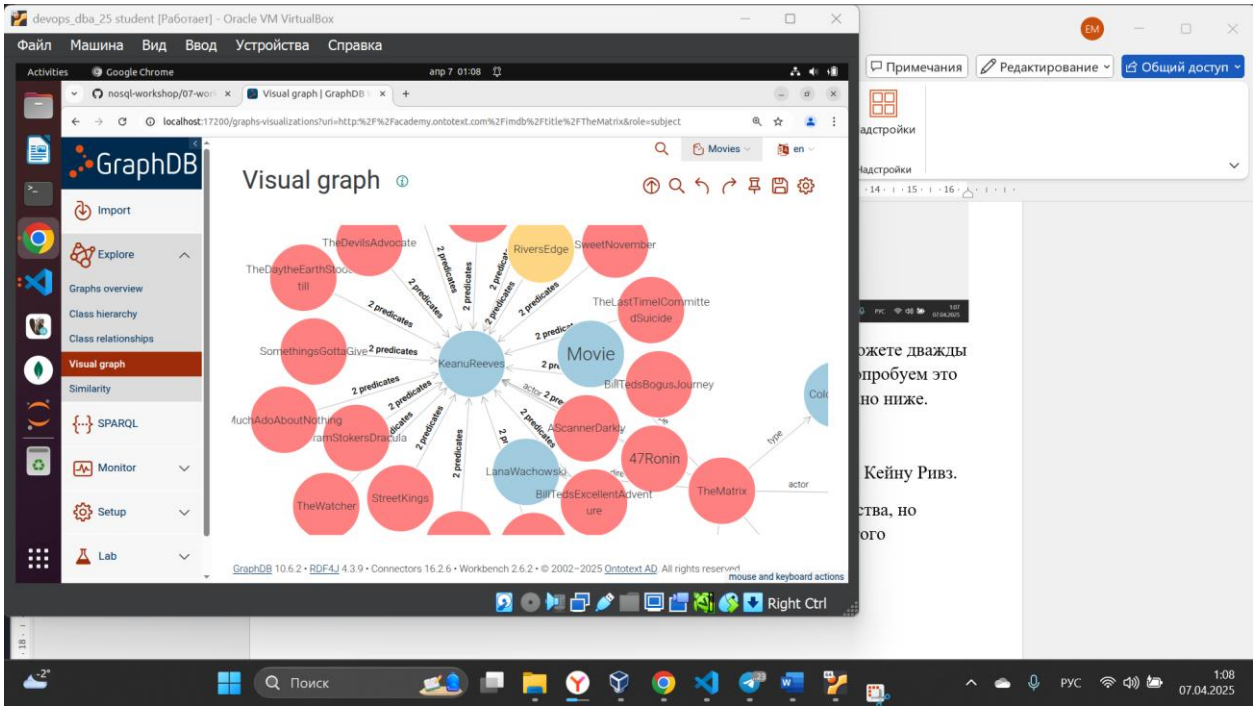
Чтобы найти фильм **The Matrix**, введите matrix. Нажмите на него, и вы увидите перечисленные связи, которые относятся к этому фильму.



Нажмите **Visual Graph**, чтобы увидеть график визуально.



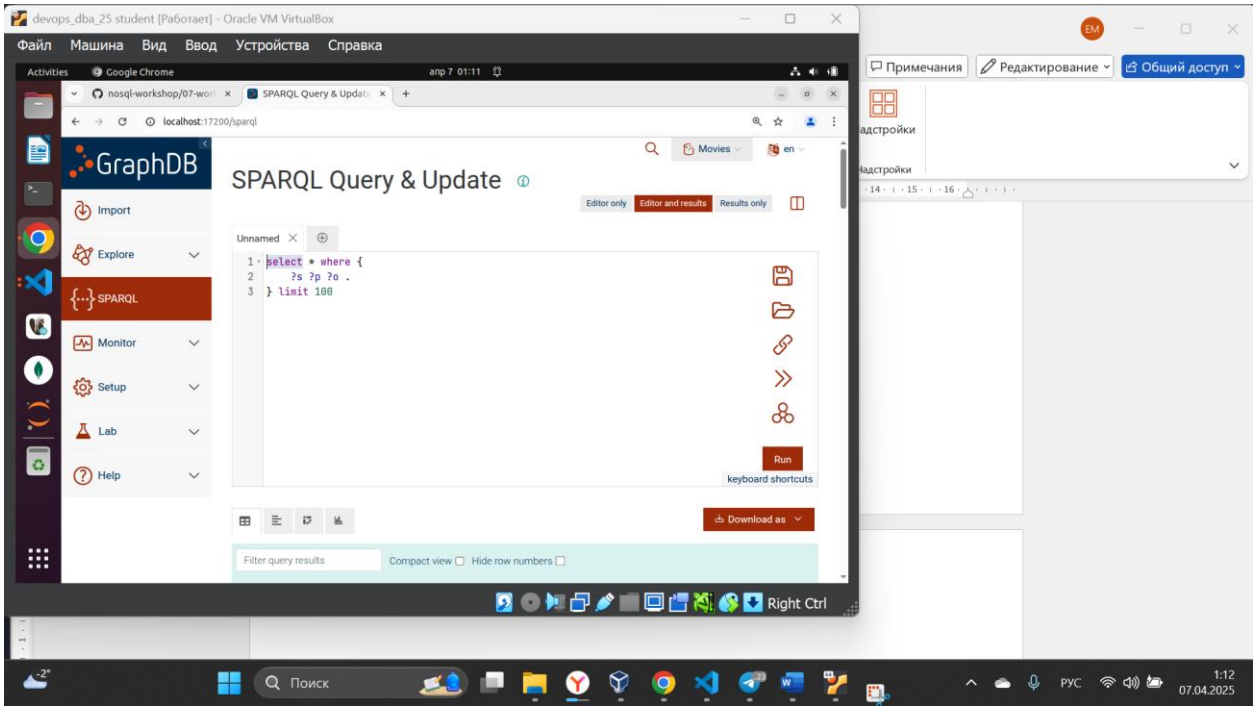
Преимущество Визуального Графа заключается в том, что вы можете дважды щёлкнуть на одном из узлов, чтобы расширить граф. Давайте попробуем это на актера **KeanuReeves**, и граф должен расшириться, как показано ниже.



Мы можем увидеть все другие фильмы, в которых также снялся Кейну Ривз.

Навигация по графу таким образом имеет некоторые преимущества, но сначала нам нужно найти стартовый узел в нашем графе. Для этого RDF/Triple-хранилище предлагает язык запросов SPARQL.

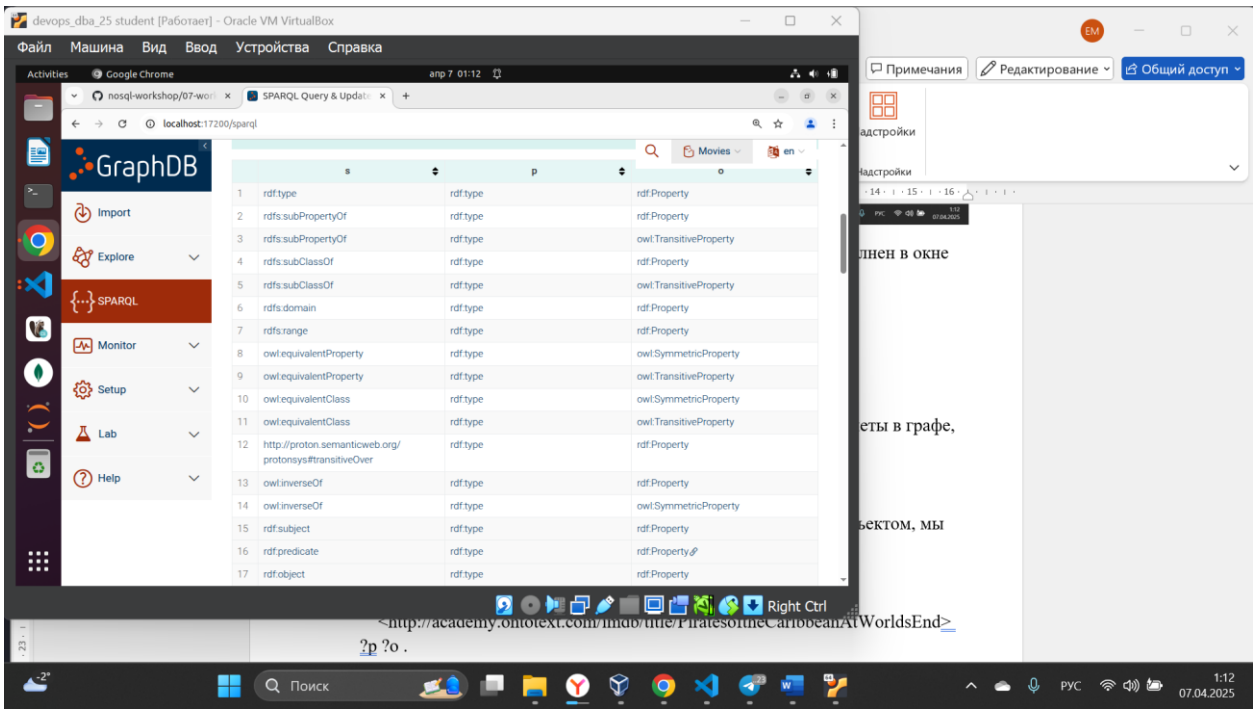
Нажмите на **SPARQL** в навигационном меню слева, и вы перейдёте в представление SPARQL, которое интегрирует редактор запросов YASGUI или редактор запросов YASGUI.



Самый простой оператор выбора SPARQL предварительно заполнен в окне запроса.

```
select * where {
    ?s ?p ?o .
} limit 100
```

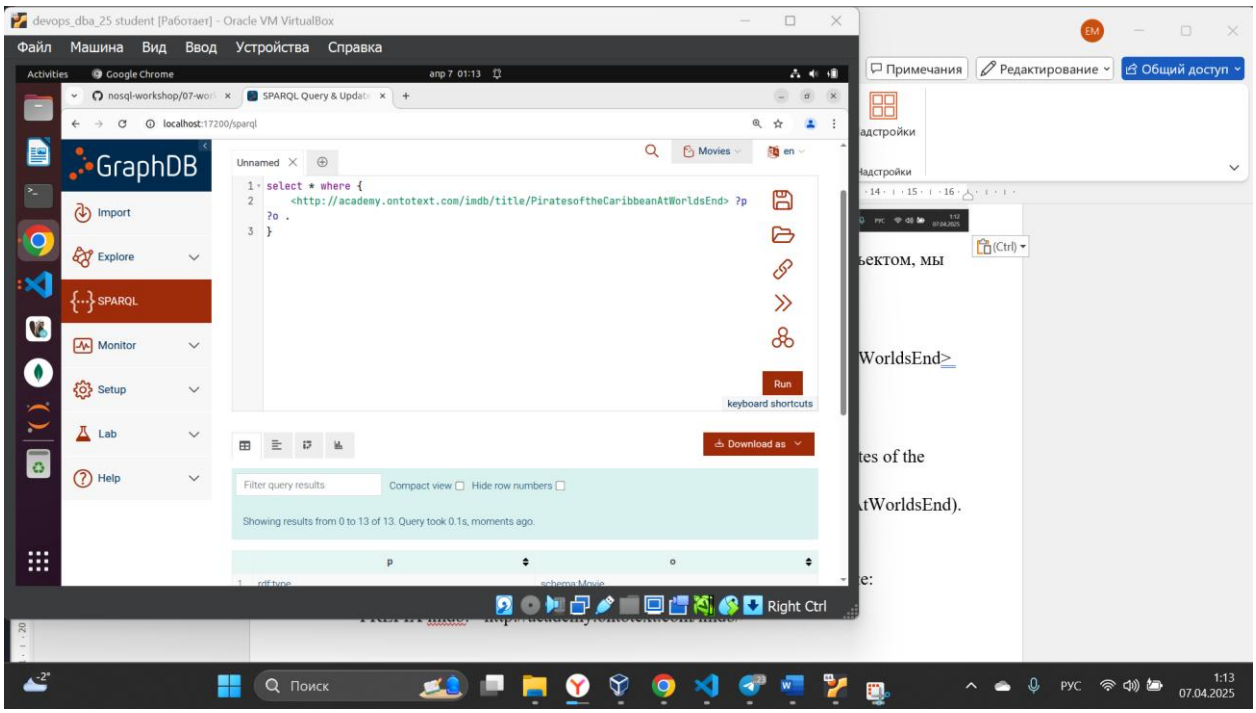
Нажмите **Run**, чтобы выполнить запрос. Он выбирает все триплеты в графе, но ограничивает результат до 100 записей.



Если мы хотим показывать только триплет с определённым субъектом, мы можем адаптировать запрос следующим образом:

```
select * where {
    <http://academy.ontotext.com/imdb/title/PiratesoftheCaribbeanAtWorldsEnd> ?p ?o .
}
```

Запрос выбирает RDF-утверждения, чей субъект — фильм "Pirates of the Caribbean: At World's End" (идентифицируемый IRI http://academy.ontotext.com/imdb/title/PiratesOfTheCaribbeanAtWorldsEnd).



Мы можем сократить IRI, установив префикс, как показано ниже:

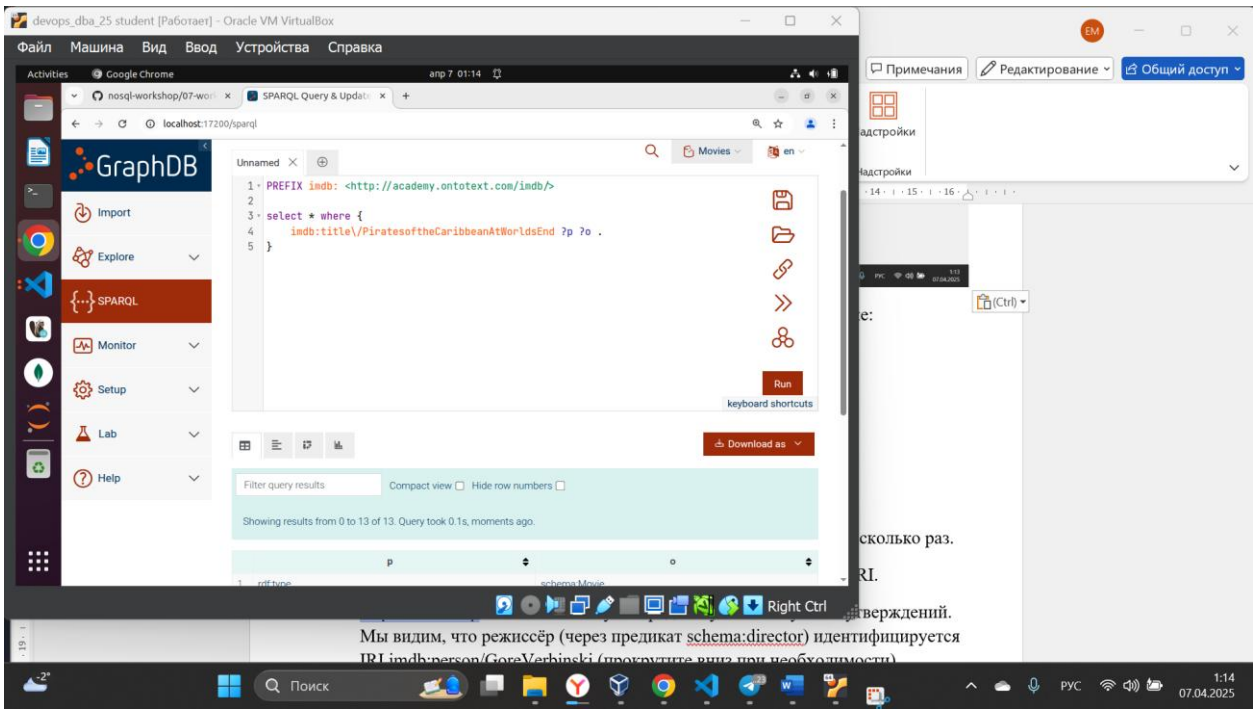
PREFIX imdb: <http://academy.ontotext.com/imdb/>

select * where {

imdb:title\/PiratesoftheCaribbeanAtWorldsEnd ?p ?o .

}

Это более полезно, если один и тот же префикс используется несколько раз.



Обратите внимание, что нужно экранировать / в сокращённом IRI.

Переменные ?p и ?o соответствуют предикату и объекту RDF-утверждений. Мы видим, что режиссёр (через предикат schema:director) идентифицируется IRI imdb:person/GoreVerbinski (прокрутите вниз при необходимости).

Следующий запрос выбирает все цветные фильмы по классу (a является сокращённой записью для rdf:type), а затем выполняет два соединения для получения названия фильма (через предикат schema:name) и количества комментариев к фильму (через предикат schema:commentCount). Наконец, результат должен быть отсортирован по количеству комментариев в порядке убывания.

PREFIX imdb: <http://academy.ontotext.com/imdb/>

PREFIX schema: <http://schema.org/>

SELECT ?movie ?name ?commentCount

WHERE {

  ?movie a imdb:ColorMovie ;

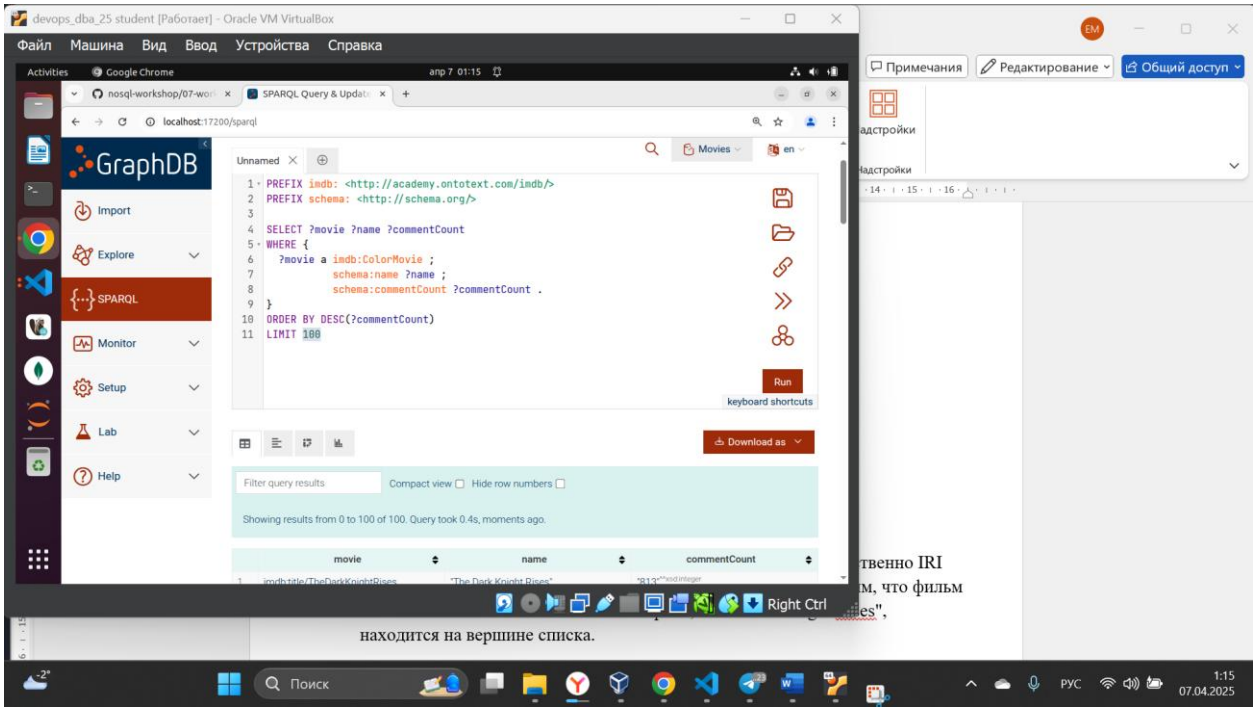
        schema:name ?name ;

        schema:commentCount ?commentCount .

}

ORDER BY DESC(?commentCount)

LIMIT 100



PREFIX imdb: <http://academy.ontotext.com/imdb/>

PREFIX schema: <http://schema.org/>

SELECT * {

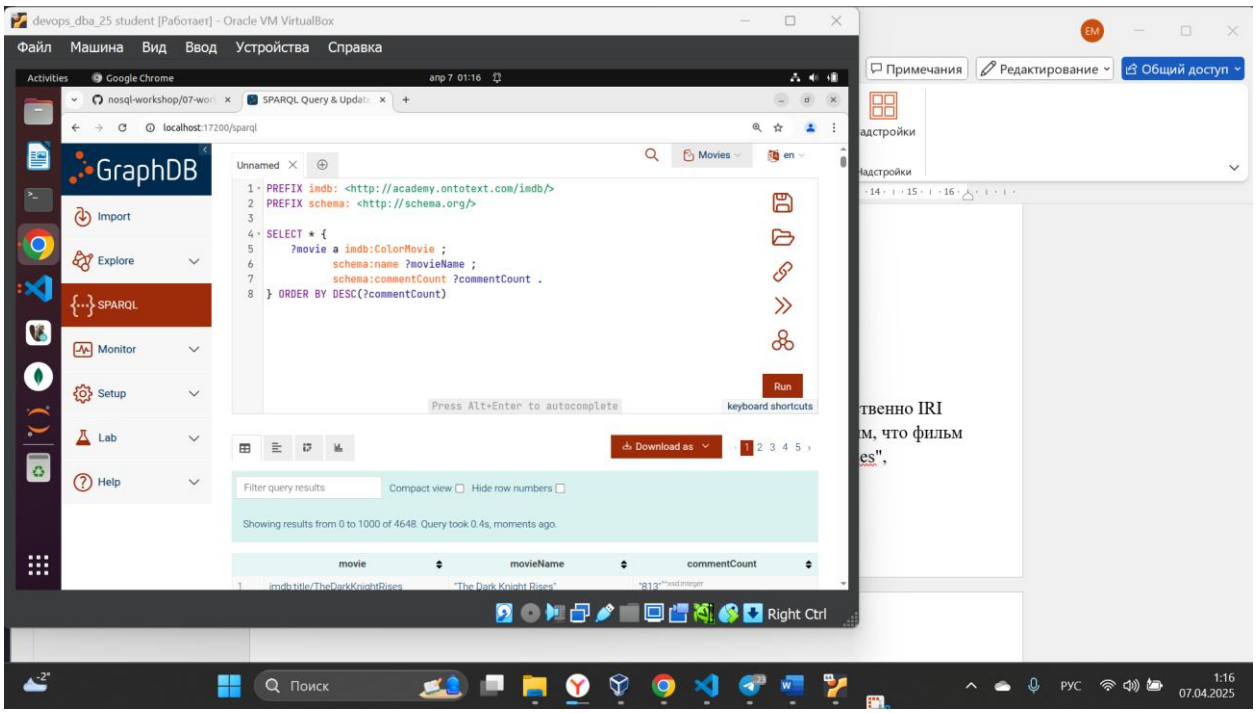
?movie a imdb:ColorMovie ;

schema:name ?movieName ;

schema:commentCount ?commentCount .

} ORDER BY DESC(?commentCount)

Таблица показывает результаты выполнения запроса.



Переменные ?movie, ?name и ?commentCount содержат соответственно IRI фильма, название фильма и количество комментариев. Мы видим, что фильм с наибольшим количеством комментариев, "The Dark Knight Rises", находится на вершине списка.

Следующий запрос выбирает RDF-утверждения, которые имеют одинаковый субъект (?movie) и одинаковый объект (?person). Для любого заданного фильма и человека должны существовать RDF-утверждения, связывающие фильм и человека с использованием
предикатов schema:director и imdb:leadActor.

PREFIX schema: <http://schema.org/>

PREFIX imdb: <http://academy.ontotext.com/imdb/>

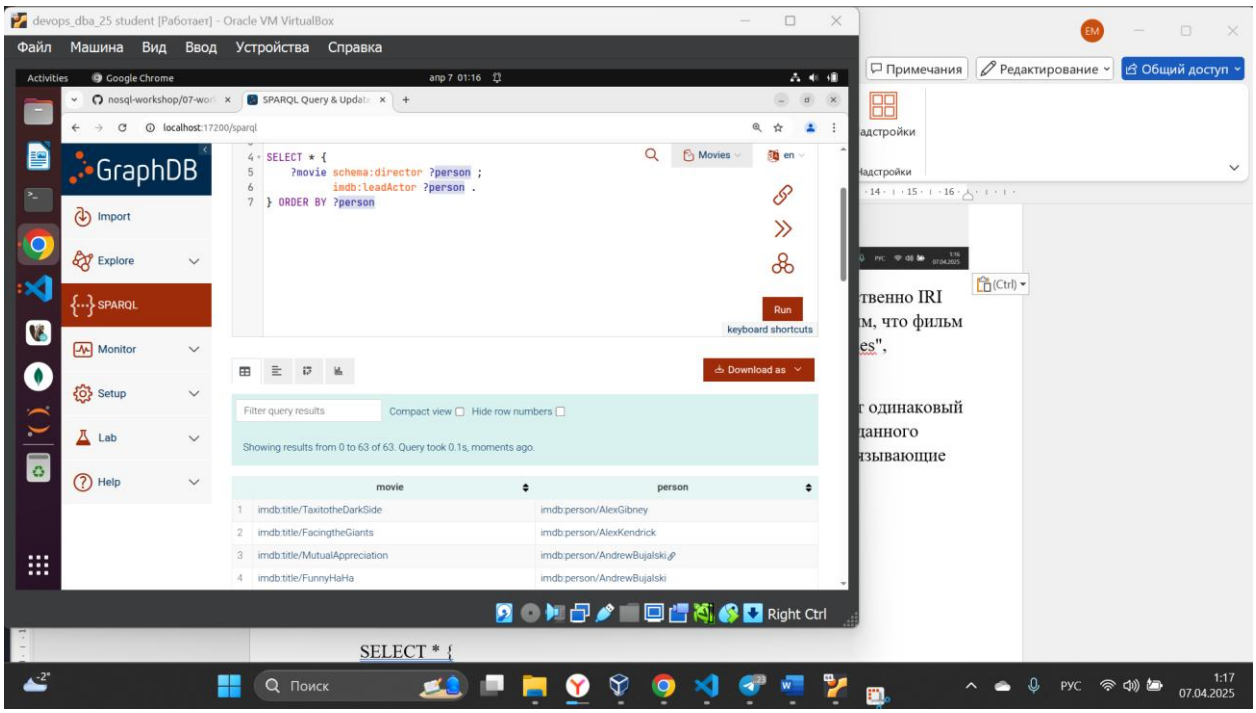
SELECT * {

?movie schema:director ?person ;

imdb:leadActor ?person .

} ORDER BY ?person

Таблица показывает результаты выполнения запроса.



Как и в предыдущем запросе, в следующем запросе мы выбираем фильмы и людей, которые являются как главными актёрами, так и режиссёрами. В этом запросе мы также используем GROUP BY ?person для группировки результатов по человеку и COUNT(?movie) для подсчёта количества фильмов, удовлетворяющих критериям для каждого человека. Подсчитанное количество возвращается в переменной ?numMovies.

PREFIX schema: <http://schema.org/>

PREFIX imdb: <http://academy.ontotext.com/imdb/>

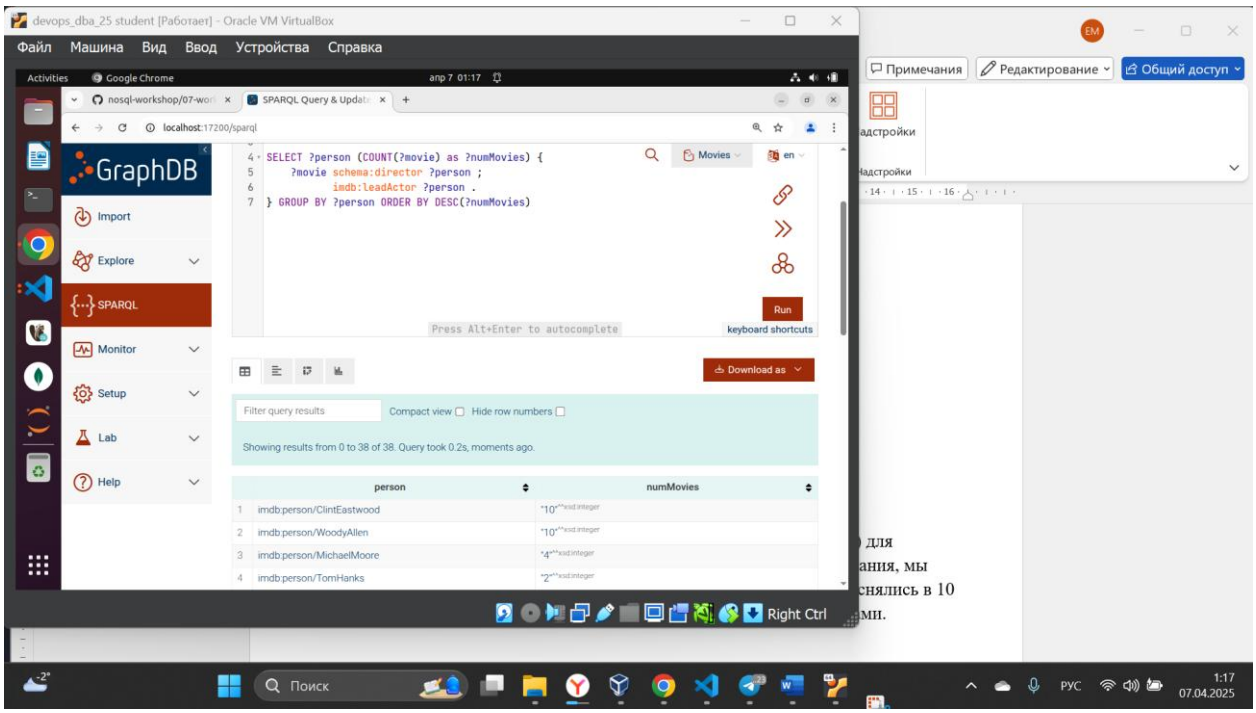
SELECT ?person (COUNT(?movie) as ?numMovies) {

?movie schema:director ?person ;

imdb:leadActor ?person .

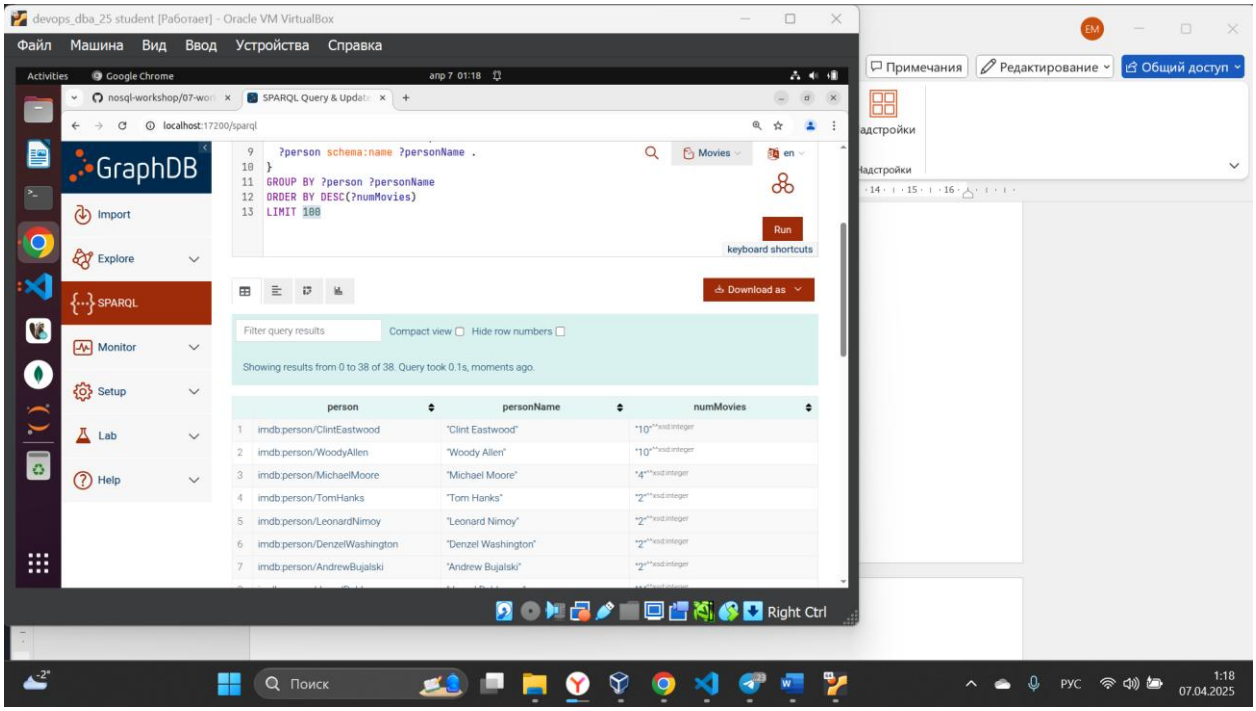
} GROUP BY ?person ORDER BY DESC(?numMovies)

Таблица показывает результаты выполнения запроса.



Так как мы также использовали ORDER BY DESC(?numMovies) для сортировки результатов по количеству фильмов в порядке убывания, мы легко можем увидеть, что как Клинт Иствуд, так и Уоди Аллен снялись в 10 фильмах, где они были как главными актёрами, так и режиссёрами.

```
PREFIX imdb: <http://academy.ontotext.com/imdb/>

PREFIX schema: <http://schema.org/>


SELECT ?person ?personName (COUNT(?movie) AS ?numMovies)

WHERE {

  ?movie schema:name ?name ;

         schema:director ?person ;

         imdb:leadActor ?person .

  ?person schema:name ?personName .

}

GROUP BY ?person ?personName

ORDER BY DESC(?numMovies)

LIMIT 100
```



Индивидуальные задания

Выполнить задание согласно своего варианта

15	Напишите запрос для получения всех фильмов, которые сняты в 1990-е годы.	Найдите фильмы, в которых главными актерами являются "Tom Cruise" и "Nicole Kidman".	Создайте запрос для нахождения всех фильмов, в которых снимались "Brad Pitt" и "Angelina Jolie".	Найдите все фильмы, у которых количество комментариев больше 100, но меньше 500.	Напишите запрос, который вернет только те фильмы, которые вышли после 2010 года.

PREFIX imdb: <http://academy.ontotext.com/imdb/>

PREFIX schema: <http://schema.org/>

SELECT ?movie ?title ?year WHERE {

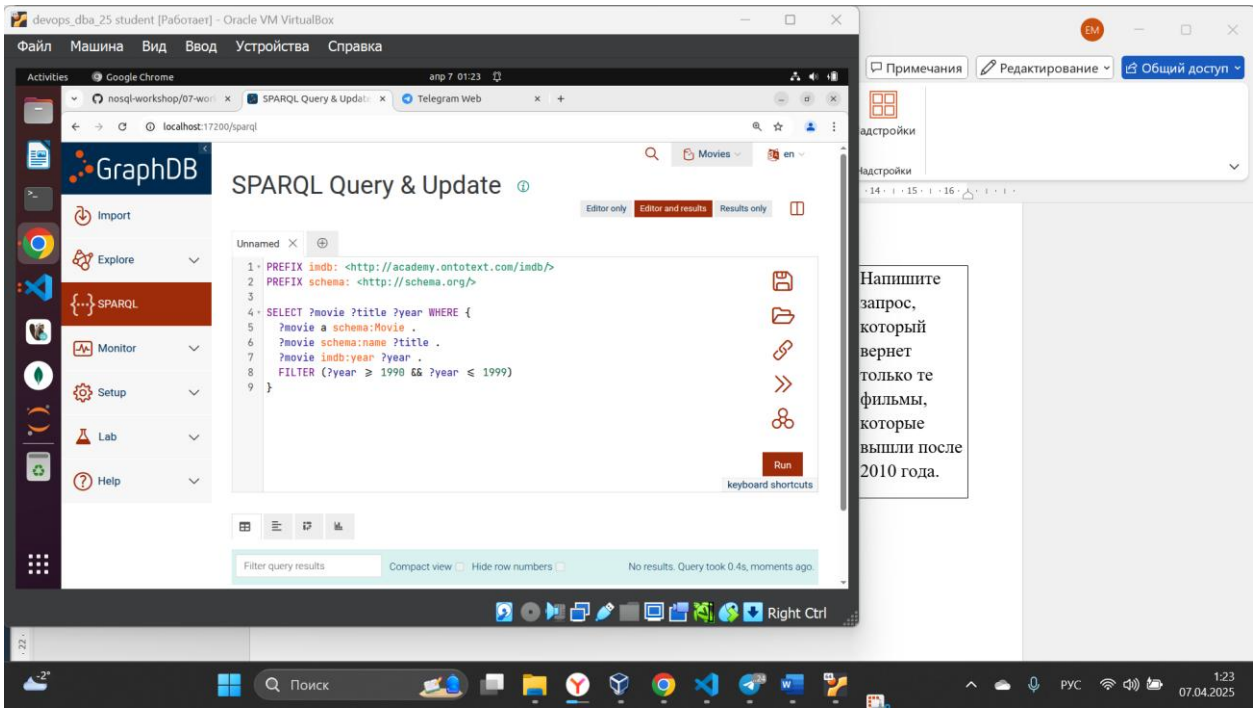
  ?movie a schema:Movie .

  ?movie schema:name ?title .

  ?movie imdb:year ?year .

FILTER (?year >= 1990 && ?year <= 1999)

}



SELECT DISTINCT ?movieTitle WHERE {

?person1 schema:director "Tom Cruise".

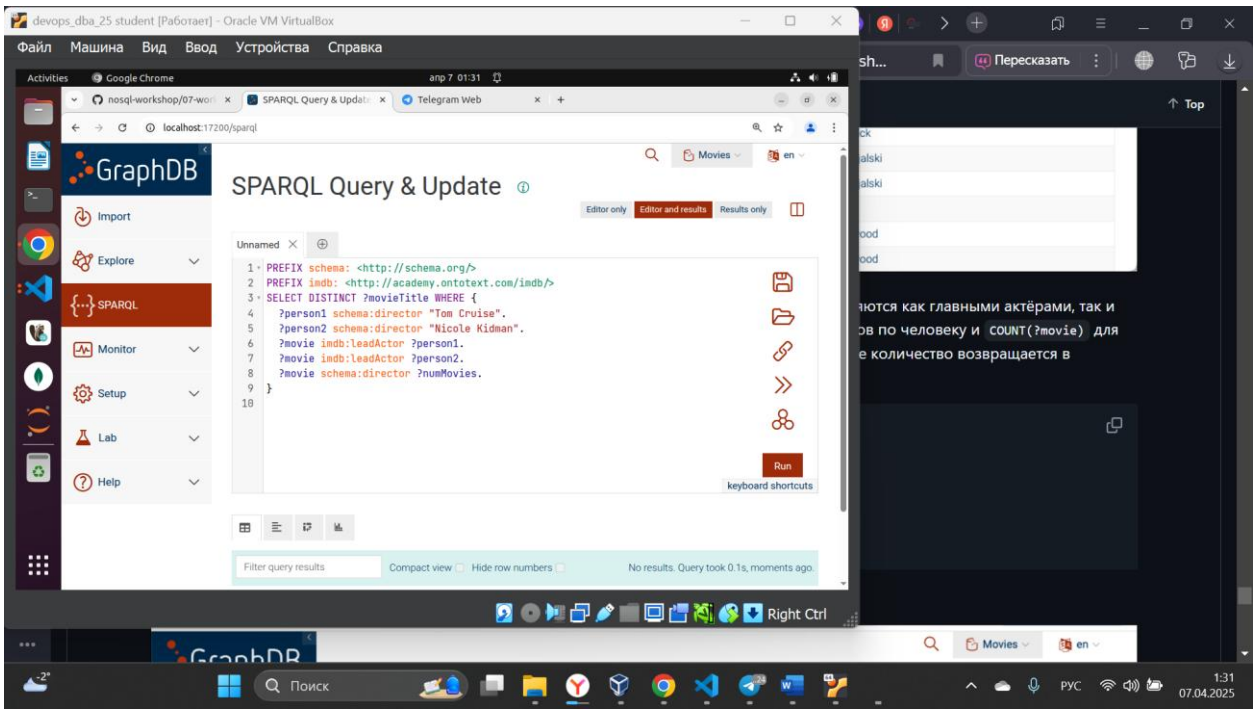
?person2 schema: director "Nicole Kidman".

?movie imdb:leaderActor ?person1.

?movie imdb: leaderActor ?person2.

?movie schema: director ?movieTitle.

}



PREFIX imdb: <http://www.example.com/imdb/>

PREFIX schema: <http://schema.org/>

SELECT DISTINCT ?movieTitle

WHERE {

?person1 schema:actor "Brad Pitt".

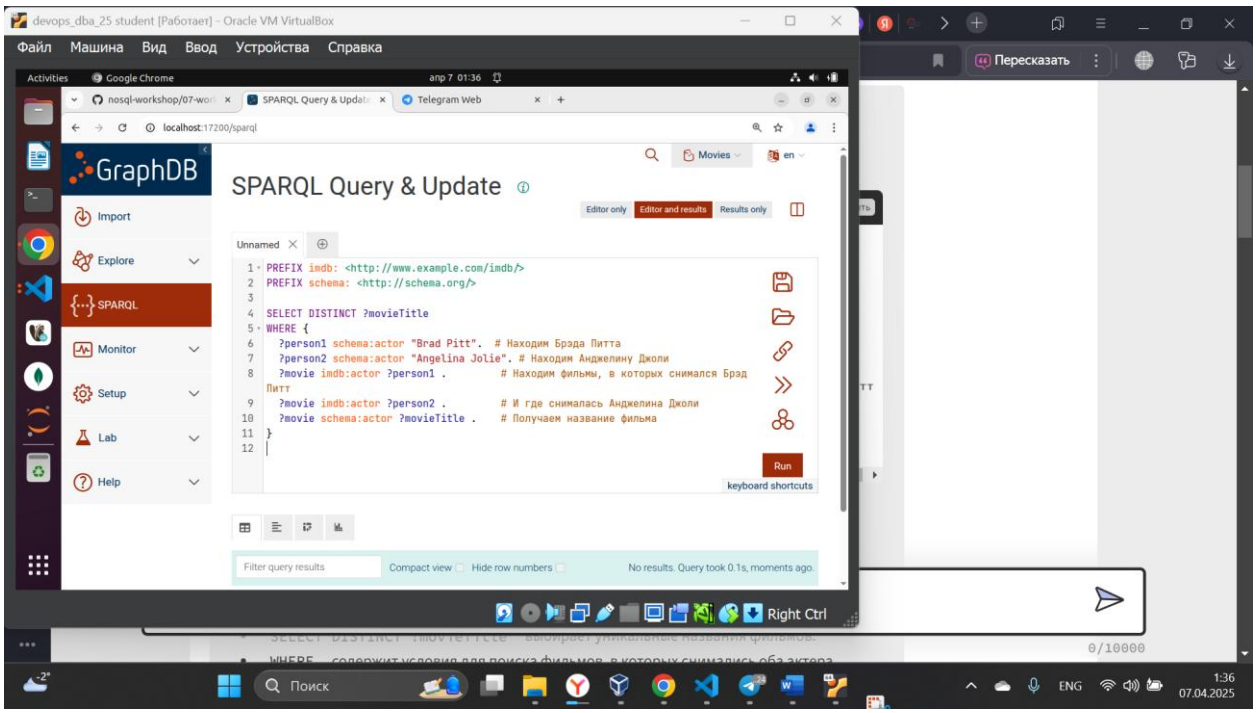
?person2 schema:actor "Angelina Jolie".

?movie imdb:actor ?person1 .

?movie imdb:actor ?person2 .

?movie schema:actor ?movieTitle .

}



PREFIX imdb: <http://academy.ontotext.com/imdb/>

PREFIX schema: <http://schema.org/>

SELECT ?movie ?title ?commentCount

WHERE {

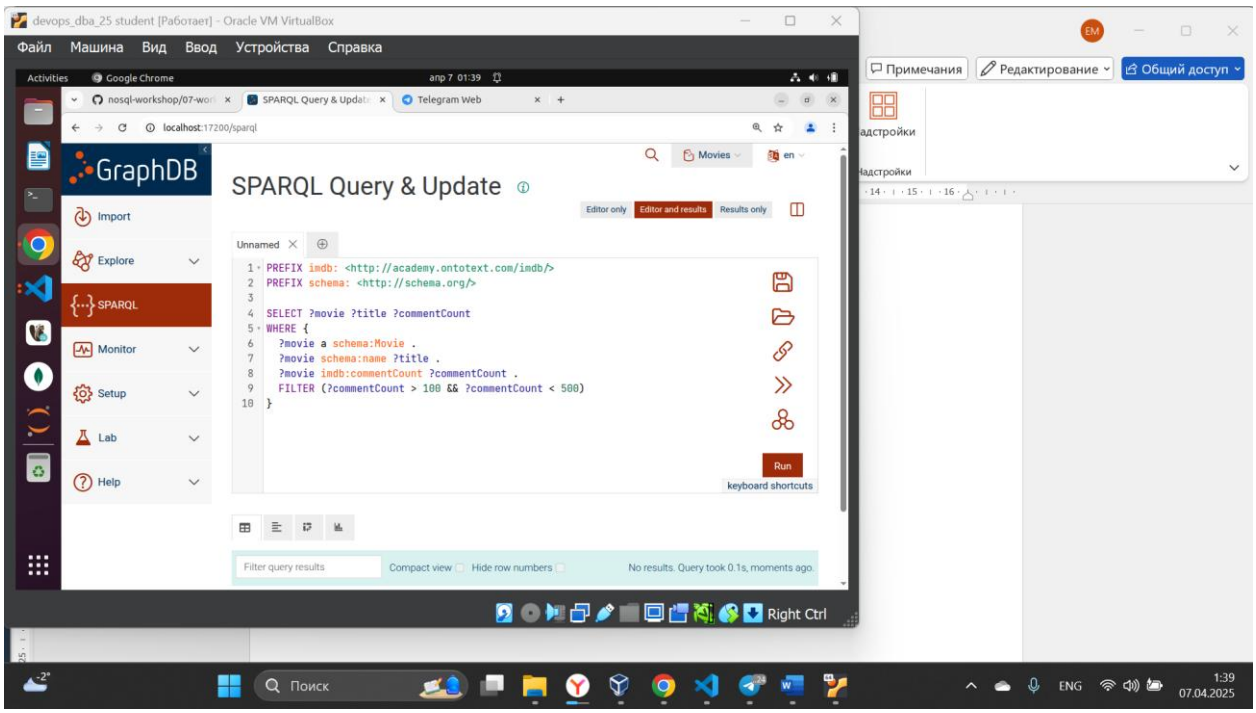
?movie a schema:Movie .

?movie schema:name ?title .

?movie imdb:commentCount ?commentCount .

FILTER (?commentCount > 100 && ?commentCount < 500)

}



PREFIX imdb: <http://academy.ontotext.com/imdb/>

PREFIX schema: <http://schema.org/>

SELECT ?movie ?title ?year

WHERE {

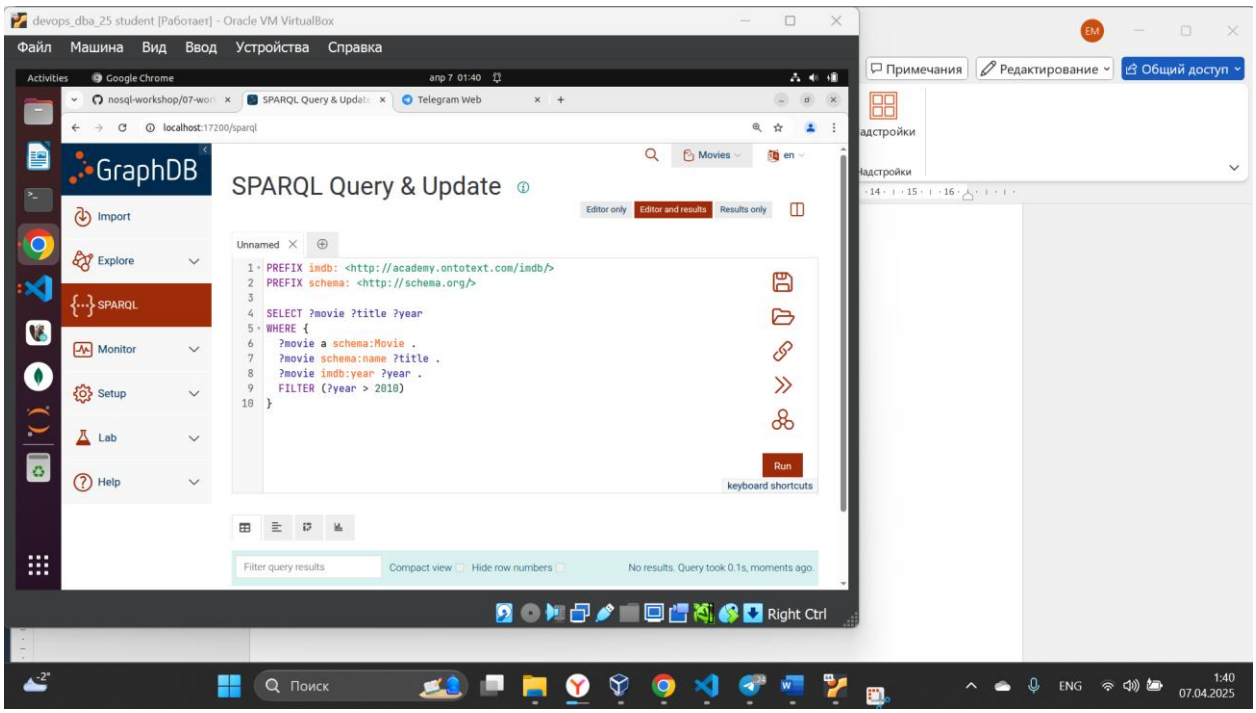
  ?movie a schema:Movie .

  ?movie schema:name ?title .

  ?movie imdb:year ?year .

  FILTER (?year > 2010)

}



Заключение

В ходе занятия мы успешно освоили работу с базой данных GraphDB в виртуальной машине с использованием Docker. Мы научились загружать RDF-данные, выполнять запросы с помощью SPARQL и анализировать результаты благодаря функциональным возможностям GraphDB. Это занятие позволило нам лучше понять принципы работы с графовыми базами данных и применить полученные знания на практике.