### Лабораторная работа № 1 Введение в Neo4j

В соответствии с рейтингами последних лет к лидирующим моделям баз данных относятся реляционные СУБД (relational DBMS): Oracle Database, MySQL, Microsoft SQL Server, PostgreSQL, документоориентированная СУБД (document-oriented database) MongoDB, графовая система управления базами данных Neo4j.

В зависимости от модели базы данных к основным СУБД NoSQL относятся:

* БД на основе пар «ключ - значение» (DynamoDB, Redis, Aerospike и др.)
* Колоночно-ориентированные СУБД или column-oriented DBMS (Cassandra, Hypertable, ConcourseDB и др)
* Документоориентированные СУБД (MongoDB, Azure DocumentDB, CouchDB и др.)
* Графовые БД (Neo4j, InfoGrid, Bigdata и др.)
* Базы данных XML (EMC Documentum xDB, eXist, Sedna и др.)

Neo4j - графическая база данных с открытым исходным кодом, которая разработана с использованием технологии Java. Он легко масштабируется и не содержит схем (NoSQL).  Другими базами данных Graph являются Oracle NoSQL Database, OrientDB, HypherGraphDB, GraphBase, InfiniteGraph и AllegroGraph.

**Преимущества Neo4j**

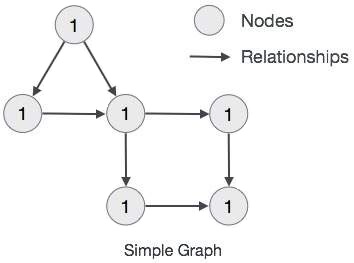
Ниже приведены преимущества Neo4j.

* Гибкая модель данных — Neo4j предоставляет гибкую простую и вместе с тем мощную модель данных, которую можно легко изменять в зависимости от приложений и отраслей.
* Анализ в реальном времени — Neo4j предоставляет результаты на основе данных в реальном времени.
* Высокая доступность — Neo4j отлично доступен для крупных корпоративных приложений реального времени с транзакционными гарантиями.
* Связанные и полуструктурированные данные. Используя Neo4j, вы можете легко представлять связанные и полуструктурированные данные.
* Простой поиск — Используя Neo4j, вы можете не только представлять, но и легко извлекать (перемещаться / перемещаться) связанные данные быстрее по сравнению с другими базами данных.
* Язык запросов Cypher — Neo4j предоставляет декларативный язык запросов для визуального представления графика с использованием синтаксиса ascii-art. Команды этого языка в удобочитаемом формате и очень просты в освоении.
* Нет объединений — используя Neo4j, он НЕ требует сложных объединений для извлечения связанных / связанных данных, так как очень легко получить сведения о соседнем узле или взаимосвязи без объединений или индексов.
* Драйверы — Neo4j может работать с —
  + REST API для работы с такими языками программирования, как Java, Spring, Scala и т. Д.
  + Java Script для работы с пользовательскими интерфейсами MVC, такими как Node JS.
  + Он поддерживает два вида Java API: Cypher API и Native Java API для разработки приложений Java. В дополнение к этому вы также можете работать с другими базами данных, такими как MongoDB, Cassandra и т. Д.
* Индексирование — Neo4j поддерживает Индексы с помощью Apache Lucence.

Основными строительными блоками модели данных Graph DB являются —

* Вершины
* Отношения
* свойства

Ниже приведен простой пример графа свойств.

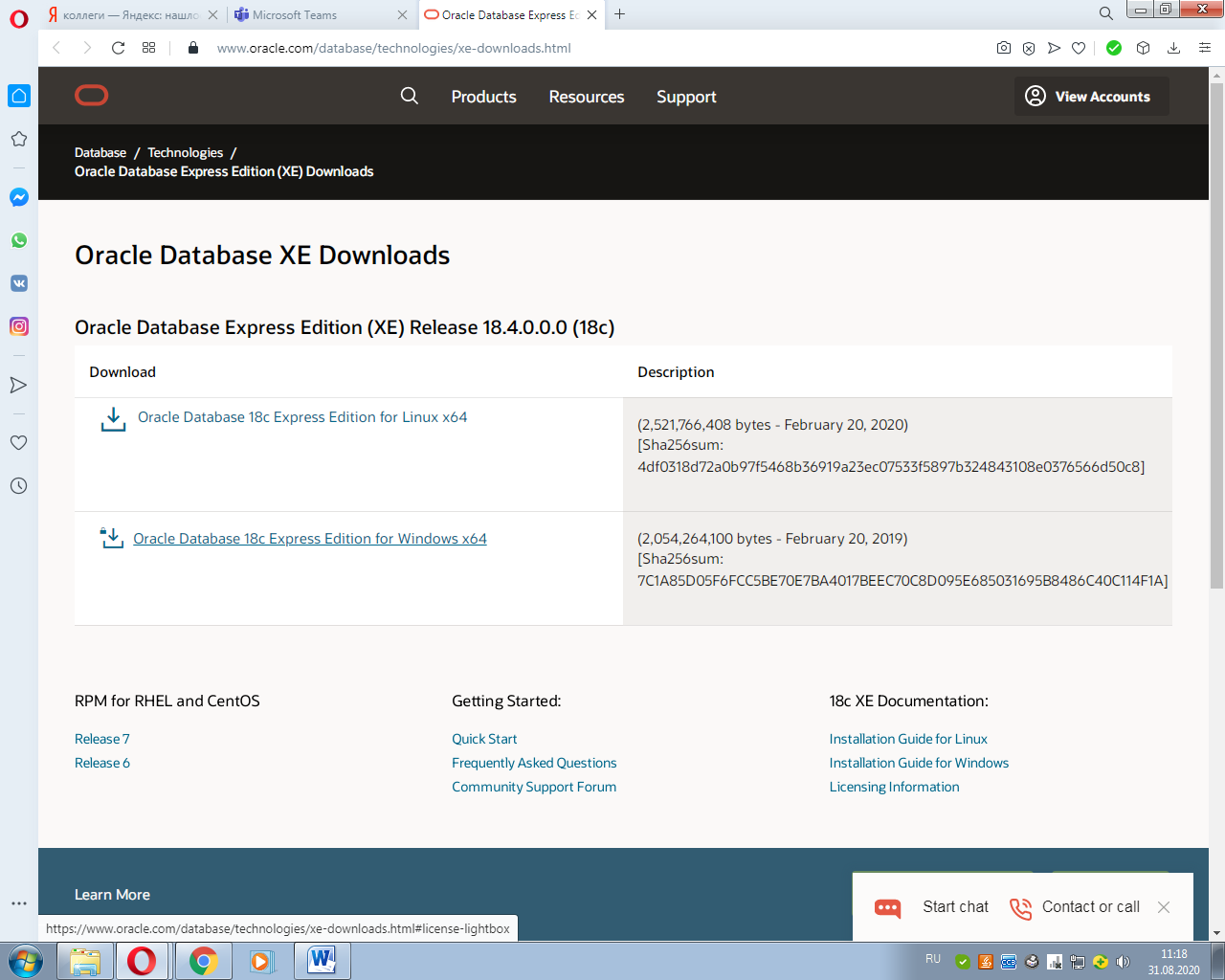


Здесь мы представили узлы, используя круги. Отношения представлены стрелками. Отношения являются направленными. Мы можем представить данные узла в терминах свойств (пары ключ-значение). В этом примере мы представили свойство Id каждого узла в круге узла.

**Ниже приведена таблица, в которой сравниваются реляционные базы данных и базы данных Graph.**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **RDBMS** | **База данных графиков** |
| 1 | таблицы | диаграммы |
| 2 | Ряды | Вершины |
| 3 | Столбцы и данные | Свойства и его значения |
| 4 | Ограничения | Отношения |
| 5 | присоединяется | пересечение |

- ЗАДАНИЕ №1. - Установите Oracle Database Express Edition (XE) Release 18.4.0.0.0 (18c) - https://www.oracle.com/database/technologies/express-edition-downloads.html

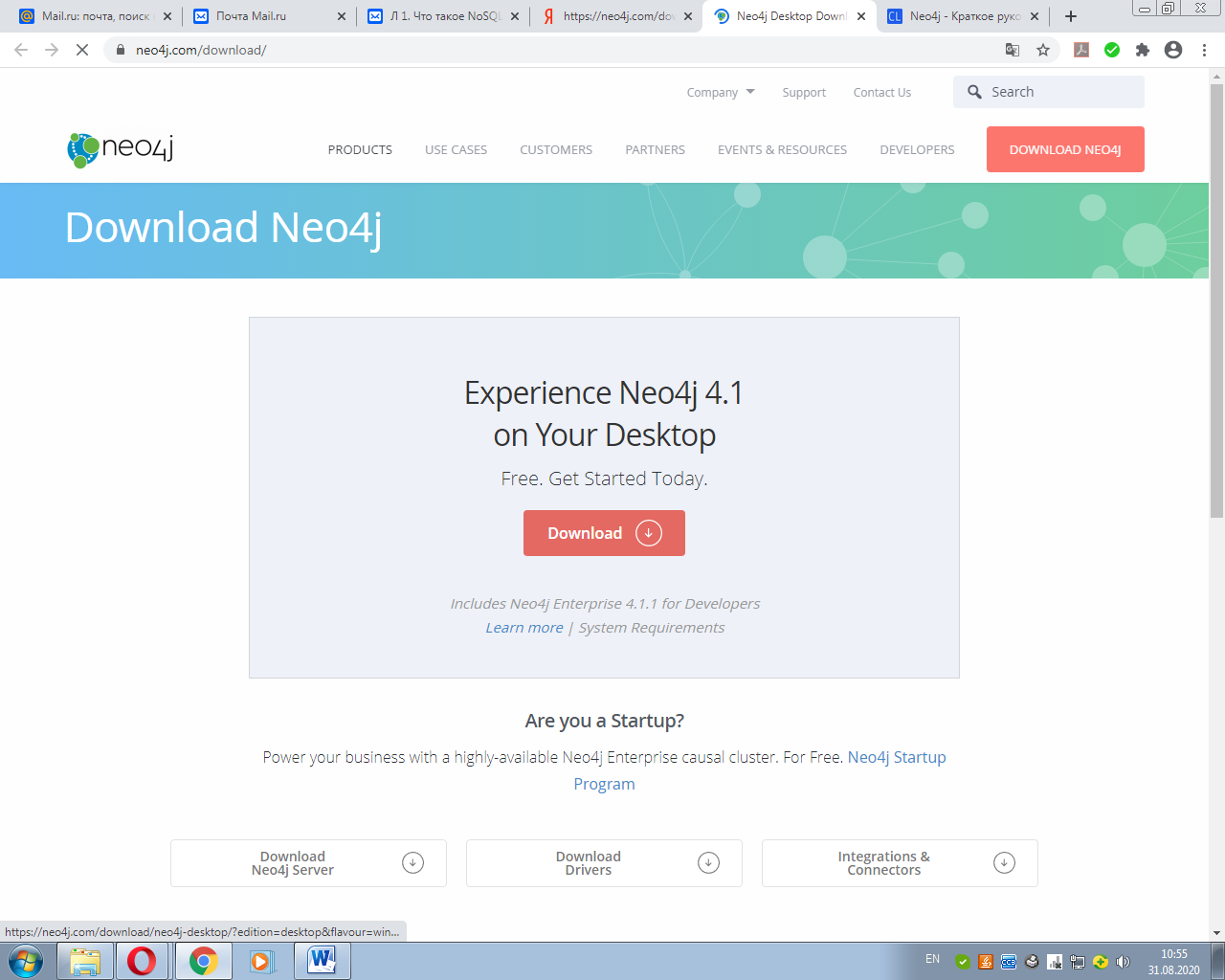


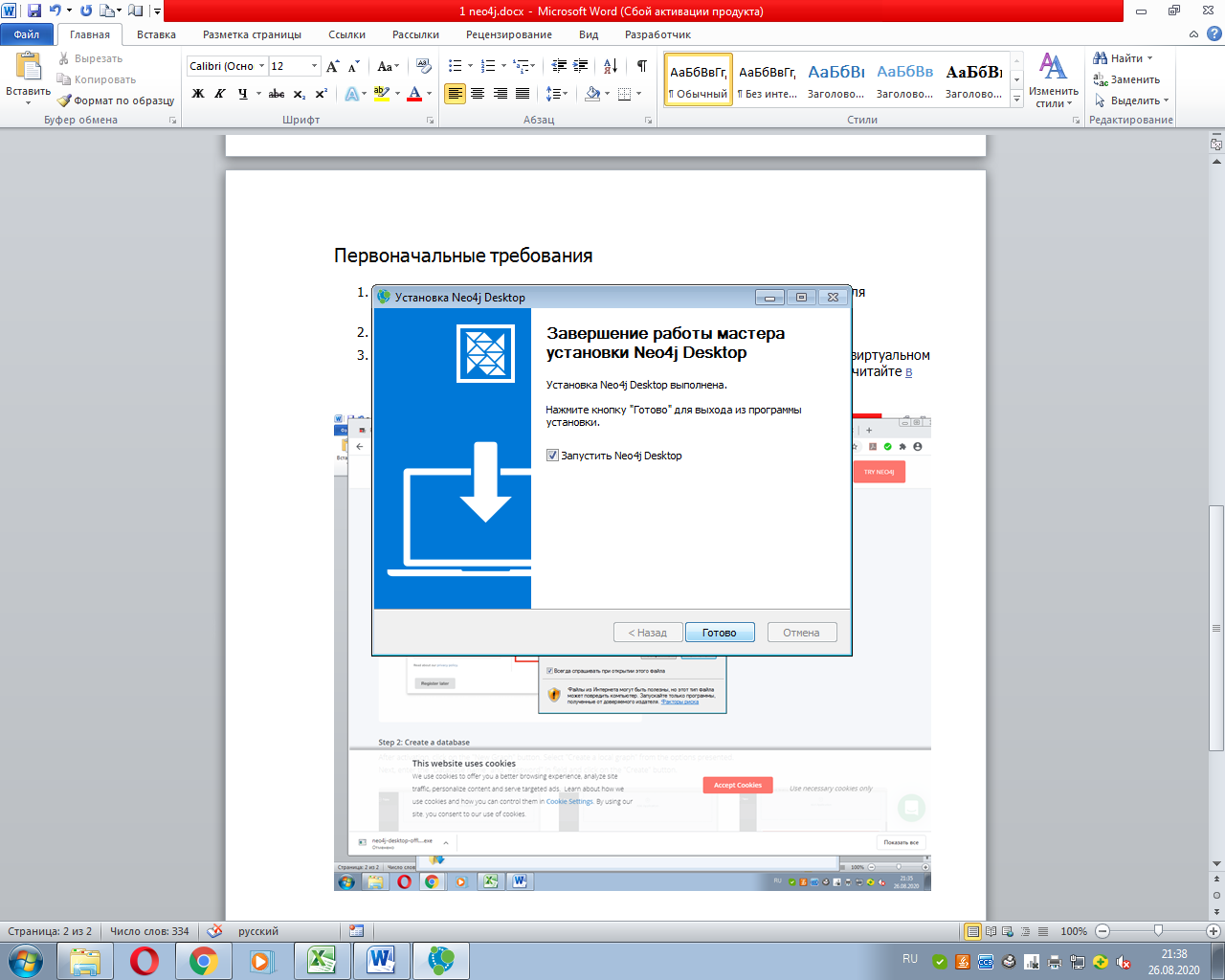
<https://rustamkhodjaev.com/2020/04/28/install_database_express_edition_18c_on_windows/> (инструкция по установке)

Oracle Database 11g Express Edition https://www.oracle.com/database/technologies/xe-prior-release-downloads.html

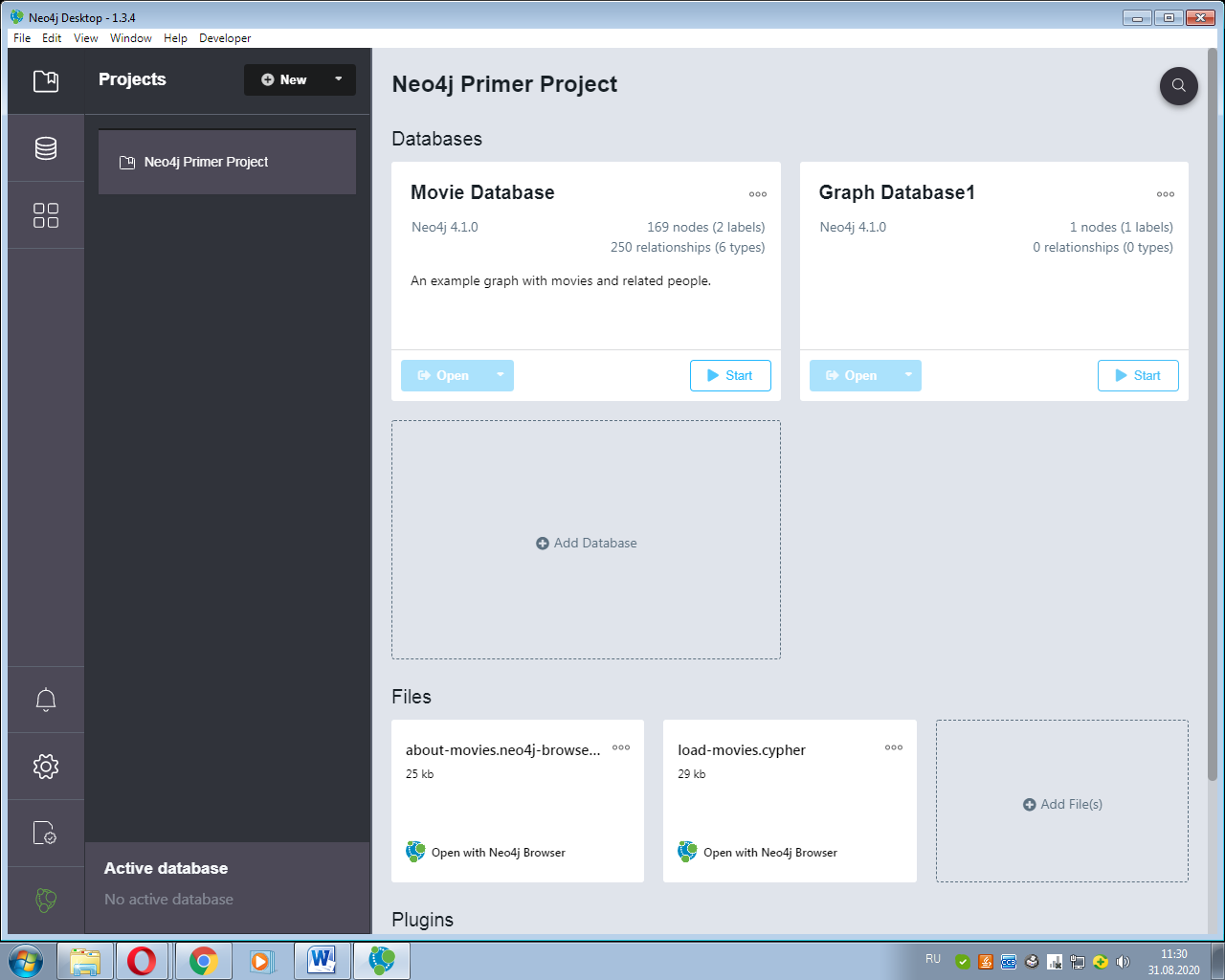
- ЗАДАНИЕ № 2 - Установите Neo4j ( для работы СУБД neo4j необходимо минимум 2Gb оперативной памяти, а для стабильной работы рекомендуется 16Gb.)

#### <https://neo4j.com/download/> Free [Download](https://neo4j.com/download/neo4j-desktop/?edition=desktop&flavour=winstall64&release=1.3.4&offline=true)





#### - ЗАДАНИЕ № 3 – Запустите встроенный в систему шаблон БД Movie Database (командой – «start» затем «open»)



### 

### - ЗАДАНИЕ № 4 - просмотрите руководство («Getting started with Neo4j Browser)

**Ниже дан перевод** Intro Guide**(вводное руководство12345…8)**

### 1Навигация по браузеру Neo4j Браузер Neo4j - это управляемый командами клиент в виде веб-оболочки. Он идеально подходит для выполнения специальных графических запросов, имея достаточно возможностей для создания прототипа приложения на основе Neo4j. Ориентированный на разработчика. Используется для написания и выполнения запросов Cypher graph. Экспортируемые табличные результаты любого результата запроса. Графическая визуализация результатов запроса, содержащих узлы и взаимосвязи. Удобное изучение HTTP API Neo4j (REST).

### 2 Редактор браузера (он же Редактор) Редактирование и выполнение инструкций Cypher и команд браузера. Редактор является основным интерфейсом для ввода и выполнения инструкций Cypher и команд браузера. Команды браузера начинаются с :, например :help. Выполнить текущую команду <Ctrl-Return> Предыдущая команда в истории <Ctrl-Стрелка вверх> Следующая команда в истории <Ctrl-Стрелка вниз> Вы можете просмотреть список привязок клавиш в любое время, выполнив команду :help keys или нажав клавишу F1 в редакторе, чтобы просмотреть все привязки клавиш для конкретного редактора.

### 3 Рамка результата Последняя выполненная команда или шифровальный запрос Фрейм результата создается для каждого выполнения и добавляется в начало потока, чтобы создать прокручиваемую коллекцию в обратном хронологическом порядке. Закрепленная рамка всегда остается в одном и том же положении. Вы можете очистить поток результирующих кадров, выполнив команду :clear. Максимальное количество результирующих кадров, отображаемых в потоке, равно 30. Вы можете изменить этот номер в настройках браузера. Вы можете просмотреть историю выполненных команд и запросов, выполнив команду : история.

### 4Многоразовая рама Исправьте свой запрос на месте Вы также можете выполнять итерации в одном и том же кадре вместо создания прокручиваемого потока кадров. Каждый повторно используемый фрейм поддерживает свою собственную локальную историю команд и обновляет основную в редакторе, если вам нужно создать новый результирующий фрейм оттуда.

### 5Боковая панель: База данных Информация о базе данных Когда Neo4j установлен, он инициируется с помощью двух баз данных – системной базы данных и базы данных neo4j по умолчанию. Запуск браузера Neo4j автоматически указывает вам на базу данных neo4j, отображаемую подсказкой neo4j$ в редакторе.

### 6Боковая панель: Избранное Быстрый способ сохранить ваши запросы Избранные запросы или команды можно сохранить в локальном хранилище и отобразить на боковой панели. Избранное является глобальным и не зависит от проекта или базы данных, что означает, что вы можете получить доступ к избранному из браузера Neo4j с различными базами данных, платформами размещения и наборами данных.

### 7 Боковая панель: Файлы проекта Сохраняйте зашифрованные файлы, чтобы поделиться ими со своими коллегами Файлы проекта позволяют сохранять запросы, руководства и другие сценарии в виде файлов Cypher. В отличие от избранного, которое сохраняется в локальном хранилище вашего браузера, файлы проекта зависят от конкретного проекта и являются фактическими файлами, хранящимися в каталоге проекта на вашем жестком диске. Все сохраненные файлы перечислены в разделе проекта, на который они ссылаются. Ящик для файлов проекта предназначен для рабочего стола Neo4j.

### 8Следующие шаги Руководство по концепциям - Узнайте о графиках свойств Neo4j Руководство по шифрованию - Изучите основы шифрования Рекомендации Команды справки - Полезные команды браузера Neo4j Клавиши справки - Сочетания клавиш

### - ЗАДАНИЕ № 5 - просмотрите учебный шаблон Movie (“Try Neo4j with live data”)

*Семантика: The Movie Graph* это база данных, содержащая актеров, режиссеров, связанных с фильмами, над которыми они сотрудничали. Это руководство покажет вам, как:

Создавать: вставить данные фильма в граф

Искать: извлечь отдельные фильмы и актеров

Выполнять запросы: найти связанных актеров и режиссеров

ЗАПУСК УЧЕБНОГО ШАБЛОНА :play movie-graph

**ПОСЛЕДОВАТЕЛЬНО ПРОРАБОТАЙТЕ СТРАНИЦЫ 1 / 8 :**

Например: 2 СОЗДАНИЕ

Нажмите на блок кода (обратите внимание, что он копируется в редактор справа )

Нажмите кнопку воспроизведения редактора, чтобы выполнить команду

Дождитесь завершения запроса

Разберитесь с графической и текстовой интерпретацией результата выполнения команд

CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome to the Real World'})

CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})

CREATE (Carrie:Person {name:'Carrie-Anne Moss', born:1967})

CREATE (Laurence:Person {name:'Laurence Fishburne', born:1961})

CREATE (Hugo:Person {name:'Hugo Weaving', born:1960})

CREATE (LillyW:Person {name:'Lilly Wachowski', born:1967})

CREATE (LanaW:Person {name:'Lana Wachowski', born:1965})

CREATE (JoelS:Person {name:'Joel Silver', born:1952})

CREATE

(Keanu)-[:ACTED\_IN {roles:['Neo']}]->(TheMatrix),

(Carrie)-[:ACTED\_IN {roles:['Trinity']}]->(TheMatrix),

(Laurence)-[:ACTED\_IN {roles:['Morpheus']}]->(TheMatrix),

(Hugo)-[:ACTED\_IN {roles:['Agent Smith']}]->(TheMatrix),

(LillyW)-[:DIRECTED]->(TheMatrix),

(LanaW)-[:DIRECTED]->(TheMatrix),

(JoelS)-[:PRODUCED]->(TheMatrix)

CREATE (Emil:Person {name:"Emil Eifrem", born:1978})

CREATE (Emil)-[:ACTED\_IN {roles:["Emil"]}]->(TheMatrix)

CREATE (TheMatrixReloaded:Movie {title:'The Matrix Reloaded', released:2003, tagline:'Free your mind'})

CREATE

(Keanu)-[:ACTED\_IN {roles:['Neo']}]->(TheMatrixReloaded),

(Carrie)-[:ACTED\_IN {roles:['Trinity']}]->(TheMatrixReloaded),

(Laurence)-[:ACTED\_IN {roles:['Morpheus']}]->(TheMatrixReloaded),

(Hugo)-[:ACTED\_IN {roles:['Agent Smith']}]->(TheMatrixReloaded),

(LillyW)-[:DIRECTED]->(TheMatrixReloaded),

(LanaW)-[:DIRECTED]->(TheMatrixReloaded),

(JoelS)-[:PRODUCED]->(TheMatrixReloaded)

CREATE (TheMatrixRevolutions:Movie {title:'The Matrix Revolutions', released:2003, tagline:'Everything that has a beginning has an end'})

CREATE

(Keanu)-[:ACTED\_IN {roles:['Neo']}]->(TheMatrixRevolutions),

(Carrie)-[:ACTED\_IN {roles:['Trinity']}]->(TheMatrixRevolutions),

(Laurence)-[:ACTED\_IN {roles:['Morpheus']}]->(TheMatrixRevolutions),

(Hugo)-[:ACTED\_IN {roles:['Agent Smith']}]->(TheMatrixRevolutions),

(LillyW)-[:DIRECTED]->(TheMatrixRevolutions),

(LanaW)-[:DIRECTED]->(TheMatrixRevolutions),

(JoelS)-[:PRODUCED]->(TheMatrixRevolutions)

CREATE (TheDevilsAdvocate:Movie {title:"The Devil's Advocate", released:1997, tagline:'Evil has its winning ways'})

CREATE (Charlize:Person {name:'Charlize Theron', born:1975})

CREATE (Al:Person {name:'Al Pacino', born:1940})

CREATE (Taylor:Person {name:'Taylor Hackford', born:1944})

CREATE

(Keanu)-[:ACTED\_IN {roles:['Kevin Lomax']}]->(TheDevilsAdvocate),

(Charlize)-[:ACTED\_IN {roles:['Mary Ann Lomax']}]->(TheDevilsAdvocate),

(Al)-[:ACTED\_IN {roles:['John Milton']}]->(TheDevilsAdvocate),

(Taylor)-[:DIRECTED]->(TheDevilsAdvocate)

CREATE (AFewGoodMen:Movie {title:"A Few Good Men", released:1992, tagline:"In the heart of the nation's capital, in a courthouse of the U.S. government, one man will stop at nothing to keep his honor, and one will stop at nothing to find the truth."})

CREATE (TomC:Person {name:'Tom Cruise', born:1962})

CREATE (JackN:Person {name:'Jack Nicholson', born:1937})

CREATE (DemiM:Person {name:'Demi Moore', born:1962})

CREATE (KevinB:Person {name:'Kevin Bacon', born:1958})

CREATE (KieferS:Person {name:'Kiefer Sutherland', born:1966})

CREATE (NoahW:Person {name:'Noah Wyle', born:1971})

CREATE (CubaG:Person {name:'Cuba Gooding Jr.', born:1968})

CREATE (KevinP:Person {name:'Kevin Pollak', born:1957})

CREATE (JTW:Person {name:'J.T. Walsh', born:1943})

CREATE (JamesM:Person {name:'James Marshall', born:1967})

CREATE (ChristopherG:Person {name:'Christopher Guest', born:1948})

CREATE (RobR:Person {name:'Rob Reiner', born:1947})

CREATE (AaronS:Person {name:'Aaron Sorkin', born:1961})

CREATE

(TomC)-[:ACTED\_IN {roles:['Lt. Daniel Kaffee']}]->(AFewGoodMen),

(JackN)-[:ACTED\_IN {roles:['Col. Nathan R. Jessup']}]->(AFewGoodMen),

(DemiM)-[:ACTED\_IN {roles:['Lt. Cdr. JoAnne Galloway']}]->(AFewGoodMen),

(KevinB)-[:ACTED\_IN {roles:['Capt. Jack Ross']}]->(AFewGoodMen),

(KieferS)-[:ACTED\_IN {roles:['Lt. Jonathan Kendrick']}]->(AFewGoodMen),

(NoahW)-[:ACTED\_IN {roles:['Cpl. Jeffrey Barnes']}]->(AFewGoodMen),

(CubaG)-[:ACTED\_IN {roles:['Cpl. Carl Hammaker']}]->(AFewGoodMen),

(KevinP)-[:ACTED\_IN {roles:['Lt. Sam Weinberg']}]->(AFewGoodMen),

(JTW)-[:ACTED\_IN {roles:['Lt. Col. Matthew Andrew Markinson']}]->(AFewGoodMen),

(JamesM)-[:ACTED\_IN {roles:['Pfc. Louden Downey']}]->(AFewGoodMen),

(ChristopherG)-[:ACTED\_IN {roles:['Dr. Stone']}]->(AFewGoodMen),

(AaronS)-[:ACTED\_IN {roles:['Man in Bar']}]->(AFewGoodMen),

(RobR)-[:DIRECTED]->(AFewGoodMen),

(AaronS)-[:WROTE]->(AFewGoodMen)

CREATE (TopGun:Movie {title:"Top Gun", released:1986, tagline:'I feel the need, the need for speed.'})

CREATE (KellyM:Person {name:'Kelly McGillis', born:1957})

CREATE (ValK:Person {name:'Val Kilmer', born:1959})

CREATE (AnthonyE:Person {name:'Anthony Edwards', born:1962})

CREATE (TomS:Person {name:'Tom Skerritt', born:1933})

CREATE (MegR:Person {name:'Meg Ryan', born:1961})

CREATE (TonyS:Person {name:'Tony Scott', born:1944})

CREATE (JimC:Person {name:'Jim Cash', born:1941})

CREATE

(TomC)-[:ACTED\_IN {roles:['Maverick']}]->(TopGun),

(KellyM)-[:ACTED\_IN {roles:['Charlie']}]->(TopGun),

(ValK)-[:ACTED\_IN {roles:['Iceman']}]->(TopGun),

(AnthonyE)-[:ACTED\_IN {roles:['Goose']}]->(TopGun),

(TomS)-[:ACTED\_IN {roles:['Viper']}]->(TopGun),

(MegR)-[:ACTED\_IN {roles:['Carole']}]->(TopGun),

(TonyS)-[:DIRECTED]->(TopGun),

(JimC)-[:WROTE]->(TopGun)

CREATE (JerryMaguire:Movie {title:'Jerry Maguire', released:2000, tagline:'The rest of his life begins now.'})

CREATE (ReneeZ:Person {name:'Renee Zellweger', born:1969})

CREATE (KellyP:Person {name:'Kelly Preston', born:1962})

CREATE (JerryO:Person {name:"Jerry O'Connell", born:1974})

CREATE (JayM:Person {name:'Jay Mohr', born:1970})

CREATE (BonnieH:Person {name:'Bonnie Hunt', born:1961})

CREATE (ReginaK:Person {name:'Regina King', born:1971})

CREATE (JonathanL:Person {name:'Jonathan Lipnicki', born:1996})

CREATE (CameronC:Person {name:'Cameron Crowe', born:1957})

CREATE

(TomC)-[:ACTED\_IN {roles:['Jerry Maguire']}]->(JerryMaguire),

(CubaG)-[:ACTED\_IN {roles:['Rod Tidwell']}]->(JerryMaguire),

(ReneeZ)-[:ACTED\_IN {roles:['Dorothy Boyd']}]->(JerryMaguire),

(KellyP)-[:ACTED\_IN {roles:['Avery Bishop']}]->(JerryMaguire),

(JerryO)-[:ACTED\_IN {roles:['Frank Cushman']}]->(JerryMaguire),

(JayM)-[:ACTED\_IN {roles:['Bob Sugar']}]->(JerryMaguire),

(BonnieH)-[:ACTED\_IN {roles:['Laurel Boyd']}]->(JerryMaguire),

(ReginaK)-[:ACTED\_IN {roles:['Marcee Tidwell']}]->(JerryMaguire),

(JonathanL)-[:ACTED\_IN {roles:['Ray Boyd']}]->(JerryMaguire),

(CameronC)-[:DIRECTED]->(JerryMaguire),

(CameronC)-[:PRODUCED]->(JerryMaguire),

(CameronC)-[:WROTE]->(JerryMaguire)

CREATE (StandByMe:Movie {title:"Stand By Me", released:1986, tagline:"For some, it's the last real taste of innocence, and the first real taste of life. But for everyone, it's the time that memories are made of."})

CREATE (RiverP:Person {name:'River Phoenix', born:1970})

CREATE (CoreyF:Person {name:'Corey Feldman', born:1971})

CREATE (WilW:Person {name:'Wil Wheaton', born:1972})

CREATE (JohnC:Person {name:'John Cusack', born:1966})

CREATE (MarshallB:Person {name:'Marshall Bell', born:1942})

CREATE

(WilW)-[:ACTED\_IN {roles:['Gordie Lachance']}]->(StandByMe),

(RiverP)-[:ACTED\_IN {roles:['Chris Chambers']}]->(StandByMe),

(JerryO)-[:ACTED\_IN {roles:['Vern Tessio']}]->(StandByMe),

(CoreyF)-[:ACTED\_IN {roles:['Teddy Duchamp']}]->(StandByMe),

(JohnC)-[:ACTED\_IN {roles:['Denny Lachance']}]->(StandByMe),

(KieferS)-[:ACTED\_IN {roles:['Ace Merrill']}]->(StandByMe),

(MarshallB)-[:ACTED\_IN {roles:['Mr. Lachance']}]->(StandByMe),

(RobR)-[:DIRECTED]->(StandByMe)

CREATE (AsGoodAsItGets:Movie {title:'As Good as It Gets', released:1997, tagline:'A comedy from the heart that goes for the throat.'})

CREATE (HelenH:Person {name:'Helen Hunt', born:1963})

CREATE (GregK:Person {name:'Greg Kinnear', born:1963})

CREATE (JamesB:Person {name:'James L. Brooks', born:1940})

CREATE

(JackN)-[:ACTED\_IN {roles:['Melvin Udall']}]->(AsGoodAsItGets),

(HelenH)-[:ACTED\_IN {roles:['Carol Connelly']}]->(AsGoodAsItGets),

(GregK)-[:ACTED\_IN {roles:['Simon Bishop']}]->(AsGoodAsItGets),

(CubaG)-[:ACTED\_IN {roles:['Frank Sachs']}]->(AsGoodAsItGets),

(JamesB)-[:DIRECTED]->(AsGoodAsItGets)

CREATE (WhatDreamsMayCome:Movie {title:'What Dreams May Come', released:1998, tagline:'After life there is more. The end is just the beginning.'})

CREATE (AnnabellaS:Person {name:'Annabella Sciorra', born:1960})

CREATE (MaxS:Person {name:'Max von Sydow', born:1929})

CREATE (WernerH:Person {name:'Werner Herzog', born:1942})

CREATE (Robin:Person {name:'Robin Williams', born:1951})

CREATE (VincentW:Person {name:'Vincent Ward', born:1956})

CREATE

(Robin)-[:ACTED\_IN {roles:['Chris Nielsen']}]->(WhatDreamsMayCome),

(CubaG)-[:ACTED\_IN {roles:['Albert Lewis']}]->(WhatDreamsMayCome),

(AnnabellaS)-[:ACTED\_IN {roles:['Annie Collins-Nielsen']}]->(WhatDreamsMayCome),

(MaxS)-[:ACTED\_IN {roles:['The Tracker']}]->(WhatDreamsMayCome),

(WernerH)-[:ACTED\_IN {roles:['The Face']}]->(WhatDreamsMayCome),

(VincentW)-[:DIRECTED]->(WhatDreamsMayCome)

CREATE (SnowFallingonCedars:Movie {title:'Snow Falling on Cedars', released:1999, tagline:'First loves last. Forever.'})

CREATE (EthanH:Person {name:'Ethan Hawke', born:1970})

CREATE (RickY:Person {name:'Rick Yune', born:1971})

CREATE (JamesC:Person {name:'James Cromwell', born:1940})

CREATE (ScottH:Person {name:'Scott Hicks', born:1953})

CREATE

(EthanH)-[:ACTED\_IN {roles:['Ishmael Chambers']}]->(SnowFallingonCedars),

(RickY)-[:ACTED\_IN {roles:['Kazuo Miyamoto']}]->(SnowFallingonCedars),

(MaxS)-[:ACTED\_IN {roles:['Nels Gudmundsson']}]->(SnowFallingonCedars),

(JamesC)-[:ACTED\_IN {roles:['Judge Fielding']}]->(SnowFallingonCedars),

(ScottH)-[:DIRECTED]->(SnowFallingonCedars)

CREATE (YouveGotMail:Movie {title:"You've Got Mail", released:1998, tagline:'At odds in life... in love on-line.'})

CREATE (ParkerP:Person {name:'Parker Posey', born:1968})

CREATE (DaveC:Person {name:'Dave Chappelle', born:1973})

CREATE (SteveZ:Person {name:'Steve Zahn', born:1967})

CREATE (TomH:Person {name:'Tom Hanks', born:1956})

CREATE (NoraE:Person {name:'Nora Ephron', born:1941})

CREATE

(TomH)-[:ACTED\_IN {roles:['Joe Fox']}]->(YouveGotMail),

(MegR)-[:ACTED\_IN {roles:['Kathleen Kelly']}]->(YouveGotMail),

(GregK)-[:ACTED\_IN {roles:['Frank Navasky']}]->(YouveGotMail),

(ParkerP)-[:ACTED\_IN {roles:['Patricia Eden']}]->(YouveGotMail),

(DaveC)-[:ACTED\_IN {roles:['Kevin Jackson']}]->(YouveGotMail),

(SteveZ)-[:ACTED\_IN {roles:['George Pappas']}]->(YouveGotMail),

(NoraE)-[:DIRECTED]->(YouveGotMail)

CREATE (SleeplessInSeattle:Movie {title:'Sleepless in Seattle', released:1993, tagline:'What if someone you never met, someone you never saw, someone you never knew was the only someone for you?'})

CREATE (RitaW:Person {name:'Rita Wilson', born:1956})

CREATE (BillPull:Person {name:'Bill Pullman', born:1953})

CREATE (VictorG:Person {name:'Victor Garber', born:1949})

CREATE (RosieO:Person {name:"Rosie O'Donnell", born:1962})

CREATE

(TomH)-[:ACTED\_IN {roles:['Sam Baldwin']}]->(SleeplessInSeattle),

(MegR)-[:ACTED\_IN {roles:['Annie Reed']}]->(SleeplessInSeattle),

(RitaW)-[:ACTED\_IN {roles:['Suzy']}]->(SleeplessInSeattle),

(BillPull)-[:ACTED\_IN {roles:['Walter']}]->(SleeplessInSeattle),

(VictorG)-[:ACTED\_IN {roles:['Greg']}]->(SleeplessInSeattle),

(RosieO)-[:ACTED\_IN {roles:['Becky']}]->(SleeplessInSeattle),

(NoraE)-[:DIRECTED]->(SleeplessInSeattle)

CREATE (JoeVersustheVolcano:Movie {title:'Joe Versus the Volcano', released:1990, tagline:'A story of love, lava and burning desire.'})

CREATE (JohnS:Person {name:'John Patrick Stanley', born:1950})

CREATE (Nathan:Person {name:'Nathan Lane', born:1956})

CREATE

(TomH)-[:ACTED\_IN {roles:['Joe Banks']}]->(JoeVersustheVolcano),

(MegR)-[:ACTED\_IN {roles:['DeDe', 'Angelica Graynamore', 'Patricia Graynamore']}]->(JoeVersustheVolcano),

(Nathan)-[:ACTED\_IN {roles:['Baw']}]->(JoeVersustheVolcano),

(JohnS)-[:DIRECTED]->(JoeVersustheVolcano)

CREATE (WhenHarryMetSally:Movie {title:'When Harry Met Sally', released:1998, tagline:'Can two friends sleep together and still love each other in the morning?'})

CREATE (BillyC:Person {name:'Billy Crystal', born:1948})

CREATE (CarrieF:Person {name:'Carrie Fisher', born:1956})

CREATE (BrunoK:Person {name:'Bruno Kirby', born:1949})

CREATE

(BillyC)-[:ACTED\_IN {roles:['Harry Burns']}]->(WhenHarryMetSally),

(MegR)-[:ACTED\_IN {roles:['Sally Albright']}]->(WhenHarryMetSally),

(CarrieF)-[:ACTED\_IN {roles:['Marie']}]->(WhenHarryMetSally),

(BrunoK)-[:ACTED\_IN {roles:['Jess']}]->(WhenHarryMetSally),

(RobR)-[:DIRECTED]->(WhenHarryMetSally),

(RobR)-[:PRODUCED]->(WhenHarryMetSally),

(NoraE)-[:PRODUCED]->(WhenHarryMetSally),

(NoraE)-[:WROTE]->(WhenHarryMetSally)

CREATE (ThatThingYouDo:Movie {title:'That Thing You Do', released:1996, tagline:'In every life there comes a time when that thing you dream becomes that thing you do'})

CREATE (LivT:Person {name:'Liv Tyler', born:1977})

CREATE

(TomH)-[:ACTED\_IN {roles:['Mr. White']}]->(ThatThingYouDo),

(LivT)-[:ACTED\_IN {roles:['Faye Dolan']}]->(ThatThingYouDo),

(Charlize)-[:ACTED\_IN {roles:['Tina']}]->(ThatThingYouDo),

(TomH)-[:DIRECTED]->(ThatThingYouDo)

CREATE (TheReplacements:Movie {title:'The Replacements', released:2000, tagline:'Pain heals, Chicks dig scars... Glory lasts forever'})

CREATE (Brooke:Person {name:'Brooke Langton', born:1970})

CREATE (Gene:Person {name:'Gene Hackman', born:1930})

CREATE (Orlando:Person {name:'Orlando Jones', born:1968})

CREATE (Howard:Person {name:'Howard Deutch', born:1950})

CREATE

(Keanu)-[:ACTED\_IN {roles:['Shane Falco']}]->(TheReplacements),

(Brooke)-[:ACTED\_IN {roles:['Annabelle Farrell']}]->(TheReplacements),

(Gene)-[:ACTED\_IN {roles:['Jimmy McGinty']}]->(TheReplacements),

(Orlando)-[:ACTED\_IN {roles:['Clifford Franklin']}]->(TheReplacements),

(Howard)-[:DIRECTED]->(TheReplacements)

CREATE (RescueDawn:Movie {title:'RescueDawn', released:2006, tagline:"Based on the extraordinary true story of one man's fight for freedom"})

CREATE (ChristianB:Person {name:'Christian Bale', born:1974})

CREATE (ZachG:Person {name:'Zach Grenier', born:1954})

CREATE

(MarshallB)-[:ACTED\_IN {roles:['Admiral']}]->(RescueDawn),

(ChristianB)-[:ACTED\_IN {roles:['Dieter Dengler']}]->(RescueDawn),

(ZachG)-[:ACTED\_IN {roles:['Squad Leader']}]->(RescueDawn),

(SteveZ)-[:ACTED\_IN {roles:['Duane']}]->(RescueDawn),

(WernerH)-[:DIRECTED]->(RescueDawn)

CREATE (TheBirdcage:Movie {title:'The Birdcage', released:1996, tagline:'Come as you are'})

CREATE (MikeN:Person {name:'Mike Nichols', born:1931})

CREATE

(Robin)-[:ACTED\_IN {roles:['Armand Goldman']}]->(TheBirdcage),

(Nathan)-[:ACTED\_IN {roles:['Albert Goldman']}]->(TheBirdcage),

(Gene)-[:ACTED\_IN {roles:['Sen. Kevin Keeley']}]->(TheBirdcage),

(MikeN)-[:DIRECTED]->(TheBirdcage)

CREATE (Unforgiven:Movie {title:'Unforgiven', released:1992, tagline:"It's a hell of a thing, killing a man"})

CREATE (RichardH:Person {name:'Richard Harris', born:1930})

CREATE (ClintE:Person {name:'Clint Eastwood', born:1930})

CREATE

(RichardH)-[:ACTED\_IN {roles:['English Bob']}]->(Unforgiven),

(ClintE)-[:ACTED\_IN {roles:['Bill Munny']}]->(Unforgiven),

(Gene)-[:ACTED\_IN {roles:['Little Bill Daggett']}]->(Unforgiven),

(ClintE)-[:DIRECTED]->(Unforgiven)

CREATE (JohnnyMnemonic:Movie {title:'Johnny Mnemonic', released:1995, tagline:'The hottest data on earth. In the coolest head in town'})

CREATE (Takeshi:Person {name:'Takeshi Kitano', born:1947})

CREATE (Dina:Person {name:'Dina Meyer', born:1968})

CREATE (IceT:Person {name:'Ice-T', born:1958})

CREATE (RobertL:Person {name:'Robert Longo', born:1953})

CREATE

(Keanu)-[:ACTED\_IN {roles:['Johnny Mnemonic']}]->(JohnnyMnemonic),

(Takeshi)-[:ACTED\_IN {roles:['Takahashi']}]->(JohnnyMnemonic),

(Dina)-[:ACTED\_IN {roles:['Jane']}]->(JohnnyMnemonic),

(IceT)-[:ACTED\_IN {roles:['J-Bone']}]->(JohnnyMnemonic),

(RobertL)-[:DIRECTED]->(JohnnyMnemonic)

CREATE (CloudAtlas:Movie {title:'Cloud Atlas', released:2012, tagline:'Everything is connected'})

CREATE (HalleB:Person {name:'Halle Berry', born:1966})

CREATE (JimB:Person {name:'Jim Broadbent', born:1949})

CREATE (TomT:Person {name:'Tom Tykwer', born:1965})

CREATE (DavidMitchell:Person {name:'David Mitchell', born:1969})

CREATE (StefanArndt:Person {name:'Stefan Arndt', born:1961})

CREATE

(TomH)-[:ACTED\_IN {roles:['Zachry', 'Dr. Henry Goose', 'Isaac Sachs', 'Dermot Hoggins']}]->(CloudAtlas),

(Hugo)-[:ACTED\_IN {roles:['Bill Smoke', 'Haskell Moore', 'Tadeusz Kesselring', 'Nurse Noakes', 'Boardman Mephi', 'Old Georgie']}]->(CloudAtlas),

(HalleB)-[:ACTED\_IN {roles:['Luisa Rey', 'Jocasta Ayrs', 'Ovid', 'Meronym']}]->(CloudAtlas),

(JimB)-[:ACTED\_IN {roles:['Vyvyan Ayrs', 'Captain Molyneux', 'Timothy Cavendish']}]->(CloudAtlas),

(TomT)-[:DIRECTED]->(CloudAtlas),

(LillyW)-[:DIRECTED]->(CloudAtlas),

(LanaW)-[:DIRECTED]->(CloudAtlas),

(DavidMitchell)-[:WROTE]->(CloudAtlas),

(StefanArndt)-[:PRODUCED]->(CloudAtlas)

CREATE (TheDaVinciCode:Movie {title:'The Da Vinci Code', released:2006, tagline:'Break The Codes'})

CREATE (IanM:Person {name:'Ian McKellen', born:1939})

CREATE (AudreyT:Person {name:'Audrey Tautou', born:1976})

CREATE (PaulB:Person {name:'Paul Bettany', born:1971})

CREATE (RonH:Person {name:'Ron Howard', born:1954})

CREATE

(TomH)-[:ACTED\_IN {roles:['Dr. Robert Langdon']}]->(TheDaVinciCode),

(IanM)-[:ACTED\_IN {roles:['Sir Leight Teabing']}]->(TheDaVinciCode),

(AudreyT)-[:ACTED\_IN {roles:['Sophie Neveu']}]->(TheDaVinciCode),

(PaulB)-[:ACTED\_IN {roles:['Silas']}]->(TheDaVinciCode),

(RonH)-[:DIRECTED]->(TheDaVinciCode)

CREATE (VforVendetta:Movie {title:'V for Vendetta', released:2006, tagline:'Freedom! Forever!'})

CREATE (NatalieP:Person {name:'Natalie Portman', born:1981})

CREATE (StephenR:Person {name:'Stephen Rea', born:1946})

CREATE (JohnH:Person {name:'John Hurt', born:1940})

CREATE (BenM:Person {name: 'Ben Miles', born:1967})

CREATE

(Hugo)-[:ACTED\_IN {roles:['V']}]->(VforVendetta),

(NatalieP)-[:ACTED\_IN {roles:['Evey Hammond']}]->(VforVendetta),

(StephenR)-[:ACTED\_IN {roles:['Eric Finch']}]->(VforVendetta),

(JohnH)-[:ACTED\_IN {roles:['High Chancellor Adam Sutler']}]->(VforVendetta),

(BenM)-[:ACTED\_IN {roles:['Dascomb']}]->(VforVendetta),

(JamesM)-[:DIRECTED]->(VforVendetta),

(LillyW)-[:PRODUCED]->(VforVendetta),

(LanaW)-[:PRODUCED]->(VforVendetta),

(JoelS)-[:PRODUCED]->(VforVendetta),

(LillyW)-[:WROTE]->(VforVendetta),

(LanaW)-[:WROTE]->(VforVendetta)

CREATE (SpeedRacer:Movie {title:'Speed Racer', released:2008, tagline:'Speed has no limits'})

CREATE (EmileH:Person {name:'Emile Hirsch', born:1985})

CREATE (JohnG:Person {name:'John Goodman', born:1960})

CREATE (SusanS:Person {name:'Susan Sarandon', born:1946})

CREATE (MatthewF:Person {name:'Matthew Fox', born:1966})

CREATE (ChristinaR:Person {name:'Christina Ricci', born:1980})

CREATE (Rain:Person {name:'Rain', born:1982})

CREATE

(EmileH)-[:ACTED\_IN {roles:['Speed Racer']}]->(SpeedRacer),

(JohnG)-[:ACTED\_IN {roles:['Pops']}]->(SpeedRacer),

(SusanS)-[:ACTED\_IN {roles:['Mom']}]->(SpeedRacer),

(MatthewF)-[:ACTED\_IN {roles:['Racer X']}]->(SpeedRacer),

(ChristinaR)-[:ACTED\_IN {roles:['Trixie']}]->(SpeedRacer),

(Rain)-[:ACTED\_IN {roles:['Taejo Togokahn']}]->(SpeedRacer),

(BenM)-[:ACTED\_IN {roles:['Cass Jones']}]->(SpeedRacer),

(LillyW)-[:DIRECTED]->(SpeedRacer),

(LanaW)-[:DIRECTED]->(SpeedRacer),

(LillyW)-[:WROTE]->(SpeedRacer),

(LanaW)-[:WROTE]->(SpeedRacer),

(JoelS)-[:PRODUCED]->(SpeedRacer)

CREATE (NinjaAssassin:Movie {title:'Ninja Assassin', released:2009, tagline:'Prepare to enter a secret world of assassins'})

CREATE (NaomieH:Person {name:'Naomie Harris'})

CREATE

(Rain)-[:ACTED\_IN {roles:['Raizo']}]->(NinjaAssassin),

(NaomieH)-[:ACTED\_IN {roles:['Mika Coretti']}]->(NinjaAssassin),

(RickY)-[:ACTED\_IN {roles:['Takeshi']}]->(NinjaAssassin),

(BenM)-[:ACTED\_IN {roles:['Ryan Maslow']}]->(NinjaAssassin),

(JamesM)-[:DIRECTED]->(NinjaAssassin),

(LillyW)-[:PRODUCED]->(NinjaAssassin),

(LanaW)-[:PRODUCED]->(NinjaAssassin),

(JoelS)-[:PRODUCED]->(NinjaAssassin)

CREATE (TheGreenMile:Movie {title:'The Green Mile', released:1999, tagline:"Walk a mile you'll never forget."})

CREATE (MichaelD:Person {name:'Michael Clarke Duncan', born:1957})

CREATE (DavidM:Person {name:'David Morse', born:1953})

CREATE (SamR:Person {name:'Sam Rockwell', born:1968})

CREATE (GaryS:Person {name:'Gary Sinise', born:1955})

CREATE (PatriciaC:Person {name:'Patricia Clarkson', born:1959})

CREATE (FrankD:Person {name:'Frank Darabont', born:1959})

CREATE

(TomH)-[:ACTED\_IN {roles:['Paul Edgecomb']}]->(TheGreenMile),

(MichaelD)-[:ACTED\_IN {roles:['John Coffey']}]->(TheGreenMile),

(DavidM)-[:ACTED\_IN {roles:['Brutus "Brutal" Howell']}]->(TheGreenMile),

(BonnieH)-[:ACTED\_IN {roles:['Jan Edgecomb']}]->(TheGreenMile),

(JamesC)-[:ACTED\_IN {roles:['Warden Hal Moores']}]->(TheGreenMile),

(SamR)-[:ACTED\_IN {roles:['"Wild Bill" Wharton']}]->(TheGreenMile),

(GaryS)-[:ACTED\_IN {roles:['Burt Hammersmith']}]->(TheGreenMile),

(PatriciaC)-[:ACTED\_IN {roles:['Melinda Moores']}]->(TheGreenMile),

(FrankD)-[:DIRECTED]->(TheGreenMile)

CREATE (FrostNixon:Movie {title:'Frost/Nixon', released:2008, tagline:'400 million people were waiting for the truth.'})

CREATE (FrankL:Person {name:'Frank Langella', born:1938})

CREATE (MichaelS:Person {name:'Michael Sheen', born:1969})

CREATE (OliverP:Person {name:'Oliver Platt', born:1960})

CREATE

(FrankL)-[:ACTED\_IN {roles:['Richard Nixon']}]->(FrostNixon),

(MichaelS)-[:ACTED\_IN {roles:['David Frost']}]->(FrostNixon),

(KevinB)-[:ACTED\_IN {roles:['Jack Brennan']}]->(FrostNixon),

(OliverP)-[:ACTED\_IN {roles:['Bob Zelnick']}]->(FrostNixon),

(SamR)-[:ACTED\_IN {roles:['James Reston, Jr.']}]->(FrostNixon),

(RonH)-[:DIRECTED]->(FrostNixon)

CREATE (Hoffa:Movie {title:'Hoffa', released:1992, tagline:"He didn't want law. He wanted justice."})

CREATE (DannyD:Person {name:'Danny DeVito', born:1944})

CREATE (JohnR:Person {name:'John C. Reilly', born:1965})

CREATE

(JackN)-[:ACTED\_IN {roles:['Hoffa']}]->(Hoffa),

(DannyD)-[:ACTED\_IN {roles:['Robert "Bobby" Ciaro']}]->(Hoffa),

(JTW)-[:ACTED\_IN {roles:['Frank Fitzsimmons']}]->(Hoffa),

(JohnR)-[:ACTED\_IN {roles:['Peter "Pete" Connelly']}]->(Hoffa),

(DannyD)-[:DIRECTED]->(Hoffa)

CREATE (Apollo13:Movie {title:'Apollo 13', released:1995, tagline:'Houston, we have a problem.'})

CREATE (EdH:Person {name:'Ed Harris', born:1950})

CREATE (BillPax:Person {name:'Bill Paxton', born:1955})

CREATE

(TomH)-[:ACTED\_IN {roles:['Jim Lovell']}]->(Apollo13),

(KevinB)-[:ACTED\_IN {roles:['Jack Swigert']}]->(Apollo13),

(EdH)-[:ACTED\_IN {roles:['Gene Kranz']}]->(Apollo13),

(BillPax)-[:ACTED\_IN {roles:['Fred Haise']}]->(Apollo13),

(GaryS)-[:ACTED\_IN {roles:['Ken Mattingly']}]->(Apollo13),

(RonH)-[:DIRECTED]->(Apollo13)

CREATE (Twister:Movie {title:'Twister', released:1996, tagline:"Don't Breathe. Don't Look Back."})

CREATE (PhilipH:Person {name:'Philip Seymour Hoffman', born:1967})

CREATE (JanB:Person {name:'Jan de Bont', born:1943})

CREATE

(BillPax)-[:ACTED\_IN {roles:['Bill Harding']}]->(Twister),

(HelenH)-[:ACTED\_IN {roles:['Dr. Jo Harding']}]->(Twister),

(ZachG)-[:ACTED\_IN {roles:['Eddie']}]->(Twister),

(PhilipH)-[:ACTED\_IN {roles:['Dustin "Dusty" Davis']}]->(Twister),

(JanB)-[:DIRECTED]->(Twister)

CREATE (CastAway:Movie {title:'Cast Away', released:2000, tagline:'At the edge of the world, his journey begins.'})

CREATE (RobertZ:Person {name:'Robert Zemeckis', born:1951})

CREATE

(TomH)-[:ACTED\_IN {roles:['Chuck Noland']}]->(CastAway),

(HelenH)-[:ACTED\_IN {roles:['Kelly Frears']}]->(CastAway),

(RobertZ)-[:DIRECTED]->(CastAway)

CREATE (OneFlewOvertheCuckoosNest:Movie {title:"One Flew Over the Cuckoo's Nest", released:1975, tagline:"If he's crazy, what does that make you?"})

CREATE (MilosF:Person {name:'Milos Forman', born:1932})

CREATE

(JackN)-[:ACTED\_IN {roles:['Randle McMurphy']}]->(OneFlewOvertheCuckoosNest),

(DannyD)-[:ACTED\_IN {roles:['Martini']}]->(OneFlewOvertheCuckoosNest),

(MilosF)-[:DIRECTED]->(OneFlewOvertheCuckoosNest)

CREATE (SomethingsGottaGive:Movie {title:"Something's Gotta Give", released:2003})

CREATE (DianeK:Person {name:'Diane Keaton', born:1946})

CREATE (NancyM:Person {name:'Nancy Meyers', born:1949})

CREATE

(JackN)-[:ACTED\_IN {roles:['Harry Sanborn']}]->(SomethingsGottaGive),

(DianeK)-[:ACTED\_IN {roles:['Erica Barry']}]->(SomethingsGottaGive),

(Keanu)-[:ACTED\_IN {roles:['Julian Mercer']}]->(SomethingsGottaGive),

(NancyM)-[:DIRECTED]->(SomethingsGottaGive),

(NancyM)-[:PRODUCED]->(SomethingsGottaGive),

(NancyM)-[:WROTE]->(SomethingsGottaGive)

CREATE (BicentennialMan:Movie {title:'Bicentennial Man', released:1999, tagline:"One robot's 200 year journey to become an ordinary man."})

CREATE (ChrisC:Person {name:'Chris Columbus', born:1958})

CREATE

(Robin)-[:ACTED\_IN {roles:['Andrew Marin']}]->(BicentennialMan),

(OliverP)-[:ACTED\_IN {roles:['Rupert Burns']}]->(BicentennialMan),

(ChrisC)-[:DIRECTED]->(BicentennialMan)

CREATE (CharlieWilsonsWar:Movie {title:"Charlie Wilson's War", released:2007, tagline:"A stiff drink. A little mascara. A lot of nerve. Who said they couldn't bring down the Soviet empire."})

CREATE (JuliaR:Person {name:'Julia Roberts', born:1967})

CREATE

(TomH)-[:ACTED\_IN {roles:['Rep. Charlie Wilson']}]->(CharlieWilsonsWar),

(JuliaR)-[:ACTED\_IN {roles:['Joanne Herring']}]->(CharlieWilsonsWar),

(PhilipH)-[:ACTED\_IN {roles:['Gust Avrakotos']}]->(CharlieWilsonsWar),

(MikeN)-[:DIRECTED]->(CharlieWilsonsWar)

CREATE (ThePolarExpress:Movie {title:'The Polar Express', released:2004, tagline:'This Holiday Season... Believe'})

CREATE

(TomH)-[:ACTED\_IN {roles:['Hero Boy', 'Father', 'Conductor', 'Hobo', 'Scrooge', 'Santa Claus']}]->(ThePolarExpress),

(RobertZ)-[:DIRECTED]->(ThePolarExpress)

CREATE (ALeagueofTheirOwn:Movie {title:'A League of Their Own', released:1992, tagline:'Once in a lifetime you get a chance to do something different.'})

CREATE (Madonna:Person {name:'Madonna', born:1954})

CREATE (GeenaD:Person {name:'Geena Davis', born:1956})

CREATE (LoriP:Person {name:'Lori Petty', born:1963})

CREATE (PennyM:Person {name:'Penny Marshall', born:1943})

CREATE

(TomH)-[:ACTED\_IN {roles:['Jimmy Dugan']}]->(ALeagueofTheirOwn),

(GeenaD)-[:ACTED\_IN {roles:['Dottie Hinson']}]->(ALeagueofTheirOwn),

(LoriP)-[:ACTED\_IN {roles:['Kit Keller']}]->(ALeagueofTheirOwn),

(RosieO)-[:ACTED\_IN {roles:['Doris Murphy']}]->(ALeagueofTheirOwn),

(Madonna)-[:ACTED\_IN {roles:['"All the Way" Mae Mordabito']}]->(ALeagueofTheirOwn),

(BillPax)-[:ACTED\_IN {roles:['Bob Hinson']}]->(ALeagueofTheirOwn),

(PennyM)-[:DIRECTED]->(ALeagueofTheirOwn)

CREATE (PaulBlythe:Person {name:'Paul Blythe'})

CREATE (AngelaScope:Person {name:'Angela Scope'})

CREATE (JessicaThompson:Person {name:'Jessica Thompson'})

CREATE (JamesThompson:Person {name:'James Thompson'})

CREATE

(JamesThompson)-[:FOLLOWS]->(JessicaThompson),

(AngelaScope)-[:FOLLOWS]->(JessicaThompson),

(PaulBlythe)-[:FOLLOWS]->(AngelaScope)

CREATE

(JessicaThompson)-[:REVIEWED {summary:'An amazing journey', rating:95}]->(CloudAtlas),

(JessicaThompson)-[:REVIEWED {summary:'Silly, but fun', rating:65}]->(TheReplacements),

(JamesThompson)-[:REVIEWED {summary:'The coolest football movie ever', rating:100}]->(TheReplacements),

(AngelaScope)-[:REVIEWED {summary:'Pretty funny at times', rating:62}]->(TheReplacements),

(JessicaThompson)-[:REVIEWED {summary:'Dark, but compelling', rating:85}]->(Unforgiven),

(JessicaThompson)-[:REVIEWED {summary:"Slapstick redeemed only by the Robin Williams and Gene Hackman's stellar performances", rating:45}]->(TheBirdcage),

(JessicaThompson)-[:REVIEWED {summary:'A solid romp', rating:68}]->(TheDaVinciCode),

(JamesThompson)-[:REVIEWED {summary:'Fun, but a little far fetched', rating:65}]->(TheDaVinciCode),

(JessicaThompson)-[:REVIEWED {summary:'You had me at Jerry', rating:92}]->(JerryMaguire)

WITH TomH as a

MATCH (a)-[:ACTED\_IN]->(m)<-[:DIRECTED]-(d) RETURN a,m,d LIMIT 10;

Описания шаблонов в языке Cypher.

В Cypher для описания шаблонов извлекаемой из графа информации, используется синтаксис ascii-art. Основные компоненты шаблонов:

* Узлы (вершины, nodes) изображают сущности (объекты), могут быть помечены одной или любым количеством меток, могут иметь одно или несколько поименованных свойств.
* Рёбра (отношения, связи, relationshis) имеют тип и направление, соединяют два узла – один начальный и один конечный. Также могут содержать свойства.
* Свойства представляются парами Key : Value, где Key – строка, а Value любого типа данных.
* Метки (labels). Узлы, помеченные одной меткой, образуют набор. Метки упрощают создание запросов. Используются для задания ограничений и индексов.

Изобразительные средства:

|  |  |
| --- | --- |
| Обозначение в языке Cypher | Описание |
| () | Узел. Сущность или объект. Изображается парой круглых скобок, внутри которых задаются тип вершины и/или значения свойств, например, (a:User) - вершина типа User. |
| [] | Рёбрo изображается парой квадратных скобок, внутри них могут задаваться условия. |
| : | Метка. Тип узла или ребра. Например, [:FRIEND] - связь типа FRIEND. |
| -> или <- или - | Направление связи. Например, ()-[]->(). Если можно не указывать ограничения на связи, опускаются прямые скобки и знаки направления () - -(). |
| {} | Свойство узла или ребра. Например, (mike{name:’Mike’}). |

Обозначение () определяет любой узел. Если необходимо обратиться к конкретному узлу, записываем в него переменную, например, (p). Несколько узлов можно пометить, записав метку после двоеточия. Например, (p: Person).

У рёбер, представляющих отношения, также могут задаваться имена и типы, например, - [rel1:KNOWS] -> . При этом типов может быть несколько, например, - [:KNOWS|:LIKE]-> .

Двунаправленные отношения представляются парой однонаправленных.

Для извлечения данных из хранилища используется запрос, содержащий шаблон фильтрации, позволяющий получать:

* (n)—>(m) — все направленные ребра из вершины n в вершину m.
* (n)—(m) — все ненаправленные ребра между вершинами n и m.
* (n:Person) — все вершины с меткой Person.
* (n:Person:Russian) — все вершины, имеющие обе метки Person и Russian.
* (n:Person {name:{value}}) — все вершины с меткой Person и отфильтрованные по дополнительному свойству.
* (n:Person)- ->(m) — ребра между вершинами n с меткой Person и вершиной m.
* () - [r {name: “some data”}] - () – находим все отношения с указанными свойствами.
* (n) - [\*1 .. 5] -> (m) – путь переменной длины от 1 до 5 ребер между вершинами n и m.
* (n) - [\*] -> (m) –путь любой длины между вершинами n и m.
* shortestPath((n1:Person) - [\*..6] - (n2:Person) ) – единственный короткий путь длиной до 6 ребер между вершинами n1 и n2 с метками Person.
* allShortestPaths((n1:Person)-[\*..6]->(n2:Person)) – все короткие пути до 6 ребер между вершинами n1 и n2 с метками Person.
* size((n) --> () --> () ) – подсчитывает пути между вершинами, указанными в шаблоне.

Например: 3 ПОИСК (примеры запросов для поиска отдельных узлов):

1. Нажмите на любой пример запроса

2. Запустите запрос из редактора

3. Обратите внимание на синтаксис шаблона

4. Попробуйте поискать другие фильмы или актеров

---------------Поиск актера "Tom Hanks"...

MATCH (tom {name: "Tom Hanks"}) RETURN tom

------------- Поиск фильма "Cloud Atlas"...

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

-------------- Поиск 10 персон...

MATCH (people:Person) RETURN people.name LIMIT 10

------------- Поиск фильмов созданных в девяностые (с 1990г.. по 2000).

MATCH (nineties:Movie) WHERE nineties.released >= 1990 AND nineties.released < 2000 RETURN nineties.title

3. ЗАПРОС Нахождение закономерностей внутри графа.

1. Актеры-это люди, которые снимались в фильмах

2. Режиссеры-это люди, которые сняли фильм

3. Какие еще существуют отношения?

Перечислите все фильмы с Томом Хэнксом...

MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED\_IN]->(tomHanksMovies) RETURN tom,tomHanksMovies

MATCH (cloudAtlas {title: "Cloud Atlas"})<-[:DIRECTED]-(directors) RETURN directors.name

Кто снял "Облачный атлас"?("Cloud Atlas")

MATCH (cloudAtlas {title: "Cloud Atlas"})<-[:DIRECTED]-(directors) RETURN directors.name

Актеры, который снимались вместе с Томом Хэнкс.

MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED\_IN]->(m)<-[:ACTED\_IN]-(coActors) RETURN coActors.name

Кто связан с "Облачным атласом"...

MATCH (people:Person)-[relatedTo]-(:Movie {title: "Cloud Atlas"}) RETURN people.name, Type(relatedTo), relatedTo

Вы слышали о классическом "Шести градусах Кевина Бэкона"?

Это запрос кратчайшего пути, называемый "Путем Бекона".

1. Шаблоны переменной длины

2. Встроенный алгоритм кратчайшего пути

Фильмы и актеры на расстоянии до 4 "прыжков" от Кевина Бэкона

MATCH (bacon:Person {name:"Kevin Bacon"})-[\*1..4]-(hollywood)

RETURN DISTINCT hollywood

Путь Бекона, кратчайший путь любых отношений с Мэг Райан

MATCH p=shortestPath(

(bacon:Person {name:"Kevin Bacon"})-[\*]-(meg:Person {name:"Meg Ryan"})

)

RETURN p

ЗАДАЧИ ДЛЯ ТРЕНИРОВКИ:

1. Найдите актеров, с которыми Том Хэнкс еще не работал, но с которыми работали его коллеги.

2. Найдите кого-нибудь, кто может познакомить Тома с его потенциальным соавтором.

Расширьте круг соактеров Тома Хэнкса, чтобы найти соактеров, которые не работали с Томом Хэнксом..

[MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED\_IN]->(m)<-[:ACTED\_IN]-(coActors),](https://translate.google.ru/" \t "_blank)

[(coActors)-[:ACTED\_IN]->(m2)<-[:ACTED\_IN]-(cocoActors)](https://translate.google.ru/" \t "_blank)

[WHERE NOT (tom)-[:ACTED\_IN]->()<-[:ACTED\_IN]-(cocoActors) AND tom <> cocoActors](https://translate.google.ru/" \t "_blank)

[RETURN cocoActors.name AS Recommended, count(\*) AS Strength ORDER BY Strength DESC](https://translate.google.ru/" \t "_blank)

[3. Найдите коллегу, кто представит Tom Hanks Tom Cruise](https://translate.google.ru/" \t "_blank)

[MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED\_IN]->(m)<-[:ACTED\_IN]-(coActors),](https://translate.google.ru/" \t "_blank)

[(coActors)-[:ACTED\_IN]->(m2)<-[:ACTED\_IN]-(cruise:Person {name:"Tom Cruise"})](https://translate.google.ru/" \t "_blank)

[RETURN tom, m, coActors, m2, cruise](https://translate.google.ru/" \t "_blank)

***[!!! УДАЛЕНИЕ – НЕ ДЕЛАЕМ !!!](https://translate.google.ru/" \t "_blank)***

[Когда вы закончите экспериментировать, вы можете удалить набор данных фильма. Примечание: Узлы нельзя удалить, если у них есть связи, поэтому нам нужно отсоединить узлы, чтобы удалить узлы. Удалите все узлы фильма и человека, а также их связи](https://translate.google.ru/" \t "_blank)

https://coderlessons.com/tutorials/bazy-dannykh/uznaite-neo4j/neo4j-uchebnik

Neo4j — пожалуй, самая распространенная графовая БД по состоянию на 2022 год. Полностью ACID-совместима и имеет множество интерфейсов для работы: от REST до нативного адаптера почти для каждого языка программирования. А также нативный GUI с человеческим лицом :) Из небольших минусов — отсутствие поддержки в популярных Database IDE вроде Jetbrains DataGrip.

Однако, графовая БД — не реляционная БД и требует иного подхода к проектированию базы данных. В этой статье я буду использовать примеры из официальной, распространяемой вместе с дистрибутивом, базы данных Movie Graph с фильмами и людьми с ними связанными: актерами, режиссерами и т.д.

Для начала, разберем общие best practices

Прежде, чем начать проектирование БД, необходимо:

* Идентифицировать всех стейкхолдеров проекта: кто, как и зачем будет использовать приложение. Это клиенты, администраторы, сотрудники бек-офиса, техподдержка, разработчики и другие. Без этого понимания будет можно продвигаться в проработке проекта, но не туда :) Это тем более важно сделать в самую первую очередь, потому что на данном этапе такая работа самая дешевая. Переделывать многократно дороже.
* Поняв, кто ваши стейкхолдеры, с каждым нужно:
  + Проработать приложение в деталях.
  + Определить пользователей системы (стейкхолдер не всегда пользователь).
  + Определить сценарии использования (use cases) системы.
  + Приоретизировать сценарии использования.

Если бы мы проектировали IMDB, то примеры хороших use cases, на которые должна отвечать система в контексте чтения данных, могут быть такими:

* Какие люди играли в этом фильме?
* В каких фильмах играл этот актер?
* Кто самый юный актер в этом фильме?
* Сколько драм срежиссировал этот режиссер?
* И так далее.

Таких use cases могут быть сотни и все они имеют значение, если будут использоваться в реальной жизни системы. Один пропущенный, но важный use case может привести к проектированию костылей, которые уберут еще одну букву в вашей реализации CAP-теоремы.

По окончании данного этапа у вас должен получиться некий набор артефактов, рассказывающих о вашей системе. Зачем и для кого она существует, и как ей будут пользоваться. После чего можно приступать к укладыванию этих знаний в ядро — базу данных.

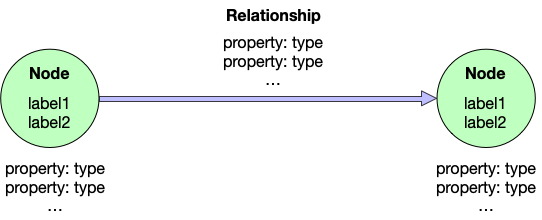
Две главные задачи моделирования — это:

* Уложить все ваши use cases в модель БД.
* Обеспечить при этом наилучшую производительность БД.

Моделирование графа

В Neo4j всего четыре типа компонентов, которыми можно описать схему данных:

* Nodes — сущности
* Labels — метки к сущностям
* Relationships — однонаправленные отношения между сущностями
* Properties — свойства сущностей или отношений



При проектировании необходимо создать два типа моделей: **модель данных** и **модель инстансов** (экземпляров данных)

Почему это важно? Потому что модель данных — это теоретическая часть вашей системы. Необходимо смоделировать как реальные данные лягут на эту теорию и спроектировать практическую модель с конкретными экземплярами данных. Это также позволит еще в песочнице посмотреть справляется ли ваша модель со всеми use cases.

Моделирование нод

Грубо говоря, все существительные в ваших use cases — это ноды.

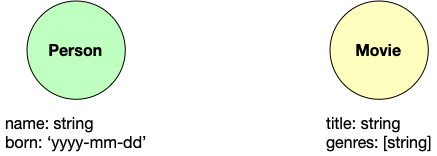
*Актер*, *режиссер*, *фильм* — это все ноды. Однако, небольшой логический тюнинг все-же необходим и мы объединим актера и режиссера в одну ноду с меткой **Person**. Конкретно в Neo4j иногда этого выгоднее не делать, но об этом позже. Также в Neo4j для специфических задач можно указать до 4 меток на одну ноду, но это скорее исключение.

Фильм поместим в ноду с меткой **Movie**.

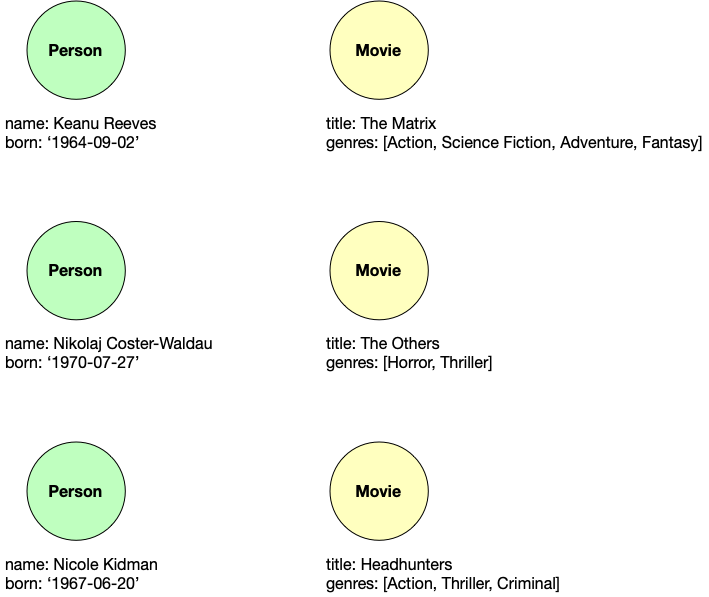
Для названия нод используется CamelCase

Теперь нам нужно обогатить ноды свойствами. Помимо очевидных свойств вроде имени человека и названия фильма, по нашим use cases у нас определяются несколько других свойств, выделим их:

* *"Кто самый юный актер в этом фильме?"* — свойство для хранения возраста человека
* *"Сколько драм срежиссировал этот режиссер?"* — свойство для хранения жанра фильма

Для названия свойств используется camelCase

Теперь на нашу теоретическую модель данных применим реальную модель инстансов:

Ноды с данными разместили, но между ними пока нет взаимосвязей

Кажется, все хорошо и мы успешно спроектировали ноды. Теперь спроектируем взаимосвязи между ними.

Моделирование взаимосвязей

Вновь обращаясь к нашим use cases, мы можем определить взаимосвязи из глаголов:

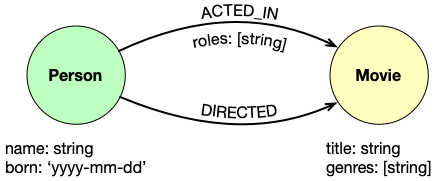
* *"Какие люди играли в этом фильме?"* — взаимосвязь ACTED\_IN
* *"В каких фильмах играл этот актер?"* — взаимосвязь ACTED\_IN
* *"Сколько драм срежиссировал этот режиссер?"* — взаимосвязь DIRECTED

В Neo4j все связи между нодами однонаправлены и это направление должно быть явно указано. Соответственно, выбирая название для связи, необходимо выбрать правильный глагол, который бы отображал как левая часть связи (активная нода) воздействует (глагол) на правую часть (пассивная нода).

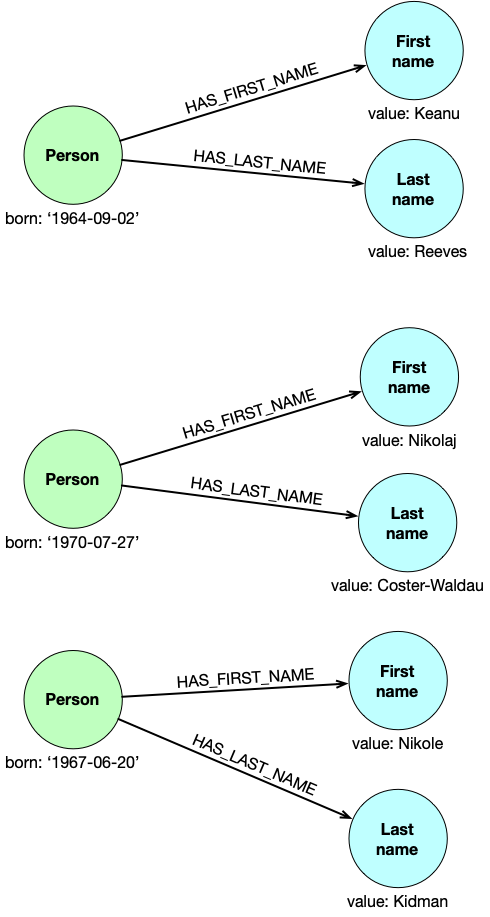
Например:

* *Recipe -- USES --> Ingredient*
* *Person -- MARRIED --> Person*  
  (да, можно ссылаться на ту же самую ноду)
* *Player -- PLAYS --> Game*
* *Book -- CONTAINS --> Chapters*

Доработаем нашу модель данных:



Важно упомянуть об одной рекомендации по проектированию БД в Neo4j: иногда вынесение свойств в отдельные ноды имеет смысл! Например, можно вынести имя и фамилию актера в разные ноды, если того требуют наши use cases:

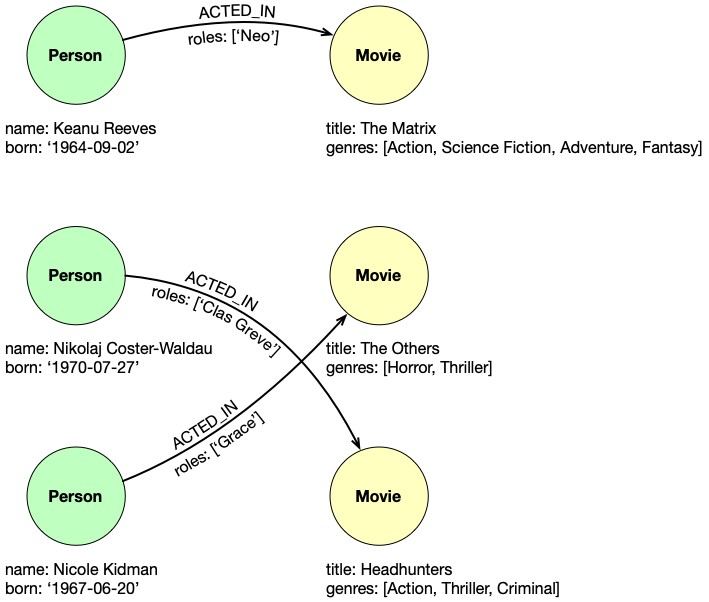


Такое разделение может быть полезно, если нам необходимо отвечать на вопросы вроде: *"Кто из персон имеет фамилию Kidman?".*

В принципе, для такой "денормализации" данных нет предела. Одновременно с ужасной, на первый взгляд, избыточностью и потенциальным появлением *супернод* с 1000+ связей, она несет в себе огромную пользу в ответе на самые нетипичные вопросы. Я бы сказал, что для исследовательских целей имеет смысл дробить базу настолько глубоко, насколько есть вероятная польза от этого. Для продакшн систем дробить только в пределах use cases.

Во взаимосвязях между нодами также можно указывать дополнительные свойства, которые описывают специфику такого взаимодействия.

Доработаем нашу модель инстансов с взаимосвязями и свойствами к ним:



В базовом виде, моделирование нашей БД в Neo4j закончено. Это больше теоретическая статья. В следующих публикациях я постараюсь рассказать о практике с конкретными примерами запросов. Также мы разберем как тестировать модель данных, рефакторить её и другие аспекты.

