МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования   
 **«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**Аналитика эффективности графовой СУБД Neo4j**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А. Пасько

(подпись, дата) (инициалы, фамилия)

Факультет компьютерных технологий и прикладной математики курс 3

Направление 02.03.02 – «Фундаментальная информатика и  
информационные технологии»

Научный руководитель доц.,   
канд.тех. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.А. Приходько

(подпись, дата) (инициалы, фамилия)

Нормоконтролер ст. преп.,   
канд.техн. наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.Е. Полупанова

(подпись, дата) (инициалы, фамилия)

Краснодар 2017

СОДЕРЖАНИЕ

[Введение 3](#_Toc501350409)

[1 Графовые NOSQL базы данных 5](#_Toc501350410)

[1.1 Характеристики графовых СУБД 7](#_Toc501350411)

[1.2 Архитектура и принципы работы Neo4j 7](#_Toc501350414)

[1.3 Язык запросов Cypher 12](#_Toc501350415)

[2 Разработка приложения на основе NOSQL Neo4j 14](#_Toc501350417)

[3 Выполнение анализа данных 15](#_Toc501350418)

[Заключение 21](#_Toc501350447)

[Список используемых источников 23](#_Toc501350448)

ВВЕДЕНИЕ

В мире технологий баз данных существует два основных направления: реляционные (SQL) и нереляционные (NoSQL) базы данных. Различия между ними значительные. Рассмотрим каждую из них.

Первое направление – реляционная база данных (SQL) была введена Эдгаром Коддом 1974-м году. Ее основная идея – это разделение данных на концептуальное и логическое проектирование. Благодаря простоте использования, она получила широкую популярность среди бизнес-приложений, но несмотря на то, что реляционные хранилища обеспечивают наилучшее сочетание простоты, устойчивости, совместимости, ее показатели не всегда выше, чем у аналогичных систем. В связи с этим, в последнее время стала очень распространенной базой данных NoSQL.

Термин NoSQL был применен Карло Строцц в 1998 году к его небольшой реляционной СУБД, которая не использовала SQL для манипулирования данными.

С 2009 года термин «NoSQL» стал применяться уже для обозначения растущего числа распределенных систем управления данными, которые отказывались от поддержки ACID-транзакций (atomicity, consistency, isolation, durability – атомарность, согласованность, изолированность.

Термин NoSQL включает в себя несколько разных видов баз данных такие как: хранилище вида “ключ-значение” (MemcacheDB, Redis, Riak), документоориентированные базы данных( Couchbase, MarkLogic, MongoDB,CouchDB), bigtable-подобные базы данных (HBase, Hypertable,  ElasticSearch) и графовые базы данных (ArangoDB, FlockDB, Giraph и Neo4j).

В данной работе будет рассматриваться графовая СУБД; в среде будет реализована база данных, на примере которой будут рассмотрены возможности декларативного языка Сypher, а также теоретический анализ полученных данных.

Целью работы является изучение принципов функционирования графовых БД на примерах, проведение элементарного анализа данных, визуализация полученных результатов.

1 Графовые NOSQL базы данных

В последнее время, данные, с которыми мы имеем дело, становятся все более взаимосвязанными, и для их удержания и обработки идеальным вариантом может служить графовая база данных, которая за последнее время стала столь популярной.

Интерес к графовым базам данных возник с конца 1980-х годов, однако, всем известно, что теория графов была впервые введена Эйлером в 18-м веке, и активно изучалась и наблюдалась исследователями многих областей. Тем не менее, только в последние несколько лет теория графов приобрела особо значимый вес в области хранения и обработки информации.

Графовые базы данных больше всего подходят для реализации проектов, которые предполагают изначально естественную графовую структуру данных, например, социальные сети. Именно в таких задачах они сильно опережают реляционные БД по производительности и простоте внесения изменений. На рисунке 1 приведен графический обзор некоторых графовых СУБД, представленных сегодня на рынке, которые основаны на разных моделях хранения и обработки информации.

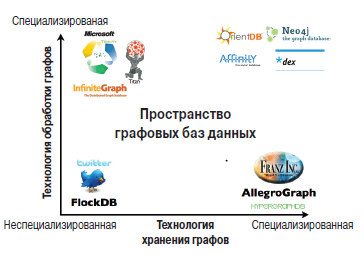


Рисунок 1 - Обзор графовых баз данных

Графовые базы данных помогли решить важнейшие проблемы в области социальных сетей, управления корпоративными данными, геопространством и в некоторых других областях (рисунок 2). Что качается их практического внедрения, то, работая с графовыми СУБД, коммерческий успех получили такие передовые компании, как Facebook, Google, Twitter и другие.



Рисунок 2 - Области применения графовых баз данных

1.1 Характеристики графовых СУБД

Рассмотрим основные характеристики графовых БД. Одним из главных достоинств выбранной СУБД является нативная структура данных или смежность без индексов, когда каждый узел содержит прямые ссылки на смежные с ним узлы. Это очень эффективный способ обработки информации, ведь нода представляет собой микроиндекс, ссылающийся на соседние узлы, а это значительно снижает затраты, если сравнивать с использованием глобальных индексов.Понятно, что время выполнения запроса будет зависеть неот общего размера графа, а от размера части графа, участвовавшемв поиске. Поэтому можно отметить, что нативное хранение баз данных - это ключевой фактор в части скорости и эффективности выбранной СУБД.

1.2 Архитектура и принципы работы Neo4j

Neo4j – это графовая система управления базами данных с открытым исходным кодом, реализованная на языке Java. Была образована американской компанией Neo Technology 2003 году. На 2017 год Neo4j считается самой распространённой графовой СУБД (по рейтингу сайта DB-Engines).

На рисунке3 изображена высокоуровневая архитектура Neo4j. Рассмотрим подробнее некоторые из уровней стека Neo4j.

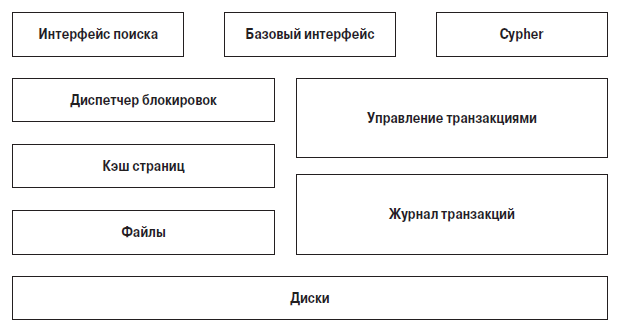


Рисунок 3 - Архитектура Neo4j

Объем памяти с каждым днем все растет и растет, и в связи с этим очень большие графы трудно разместить полностью в памяти, поэтому многие графовые базы данных применяют кэширование, чтобы обеспечить быстрый доступ к графу. Также с версии 2.2 Neo4j использует страничное кэширование LRU-K. Под страничными кэшированием понимается применение ориентированного кэша на страницы, т.е кэширование, разделяющее каждое из хранилищ на области, а затем помещающее в кэш определенное количество фрагментов из файла хранения.

Выбранная СУБД хранит графы в разных файлах; каждый файл хранит определенную информацию части графа. Данное разделение существенно влияет на производительность БД.

Рассмотрим базовый программный интерфейс Neo4j, который предоставляет доступ к графовым узлам, взаимосвязям, свойствам и меткам. Данные извлекаются из графа по мере их востребованности клиентом программного интерфейса. В любой момент пользователь имеет возможность прервать обход. При записи базовый программный интерфейс предоставляет возможность управления транзакциями для обеспечения атомарности, согласованности, изолированности и долговременности.

Транзакции – это стандарт в вычислительных системах, который остается фундаментальной основой, которая обеспечивает надежность. Многие NOSQL-хранилища не поддерживают транзакций, но Neo4j не входит в этот перечень.

Архитектура Neo4j дает возможность управлять информацией, хранить и производить обход узлов и связей. В Neo4j отношения являются важной составляющей, которые позволяют создавать связи между сущностями (рисунок 4).



Рисунок 4 – Пример связанных сущностей

Neo4j оперирует следующими объектами:

- вершины (nodes) – и их помощью представляются сущности графа, также, в зависимости от отношений в графе, могут быть использованы для представления связи. Самый простой граф представляет собой одну вершину. Вершина может не иметь, либо иметь одно или более именованных значений, которые указываются в виде свойств;

- связи (relationships) – соединяют узлы-начальный и конечный. Связи, также, как и узлы, могут иметь свойства;

- свойства (prosperties) – именованные значения, где имя – это строка. Поддерживаемые значения: числовые, строковые;

- метки (labels) – представляются в виде графов, которые были сгруппированы в наборы. Все узлы, помеченные одной меткой, принадлежит к одному набору. Они намного упрощают написание запросов к базе. Вершина может быть помечена любым количеством меток. Метки используются для задания ограничений и добавления индексов для свойств.

В графическом интерфейсе можно видеть, что вся информация отображается в блоке Database Information, где данные, связанные с узлами и метками, хранятся в части Node Labels, а со связями - в Relationship Types, это дает возможность в любой момент восстановить нужную нам информацию о связях или узлах.

В СУБД Neo4j реализован графический интерфейс (рисунок 5), в котором могут быть удобно представлены графовые данные, но данные могут быть представлены не только в виде графа, но еще и в табличном и в виде кода.

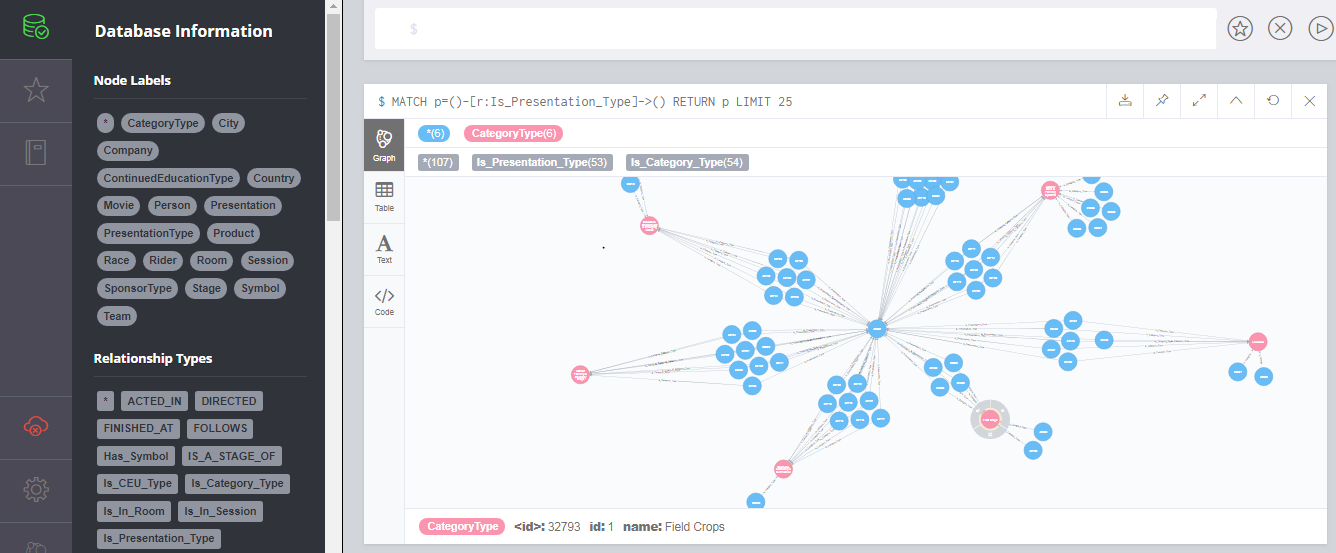


Рисунок 5 - Интерфейс Neo4j

1.3 Язык запросов Cypher

Cypher - декларативный язык. Он фокусируется на ясности выражения того, что необходимо извлечь из графа, а не в том, как получить это. Это и отличает его от императивных языков или скриптовых сценариев. Такой подход существенно упрощает процесс оптимизации, не обременяя пользователя информацией о структуре базы данных и не вынуждая обновлять код запроса только потому, что логическая структура базы данных изменилась (появление новых индексов и т.д.).

Примеры обозначения искомых данных:

- (n)-->(m) - все направленные ребра из вершины n в вершину m;

- (n:Holder) - все вершины с меткой Holder;

- (n:Holder:Account) - все вершины, имеющие обе метки Holder и Account;

- (n:Holder{name:{Mark}}) - все вершины с меткой Holder и отфильтрованные по дополнительному свойству name;

- (n:Holder)-->(m) - ребра между вершинами n с меткой Person и m;

- ()-[:IS\_HOLDER]->() - отобразить вершины, которые связаны между собой именованной связью.

2 Разработка приложения на основе NOSQL Neo4j

Была создана первая база данных в Neo4j, которая хранит данные о научной конференции. Выбранная СУБД позволяет исследовать конференцию различными типами категорий. Она состоит из основных узлов: Company, Person и Presentation и связи: Is\_Sponsor\_Type, Works\_For, Presented\_By, Is\_Presentation\_Type, Is\_Category\_Type, Is\_CEU\_Type, Is\_In\_Room, Is\_In\_Session, Has\_Symbol. Через выбранные узлы (ноды) происходит связь с другими узлами: Sponsor Type, Room, Session, Symbol, Presentation Type, Category Type, CEU Type. Общая модель базы данных представлена на рисунке 6 .

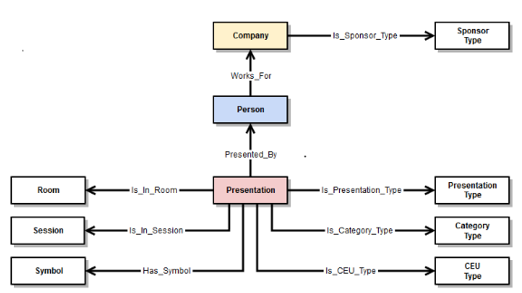


Рисунок 6 - Общая модель базы данных.

Во второй СУБД имеются данные из романа «Шторм мечей».

В задаче требуется подробно описать сеть взаимодействий символов, используя анализ текста; извлечь сущности, чтобы найти символы упомянутые в тексте. Должен быть применен алгоритм анализа социальной сети, чтобы найти наиболее важные символы в сети, также некоторые дополнительные запросы.

3 Выполнение анализа данных

Язык Cypher позволяет строить различные по сложности запросы, сочетая в себе с одной стороны элементы SQL, а с другой графическое изображение связей.

На рисунке 7 изображена графовая база данных, которая представляет общую модель первой базы данных рисунка 6, т.е изображены все основные ноды графа, от которых могут образовывать связи. Можно заметить, что узлы различаются по цветам, это значит, что они имеют разные типы.

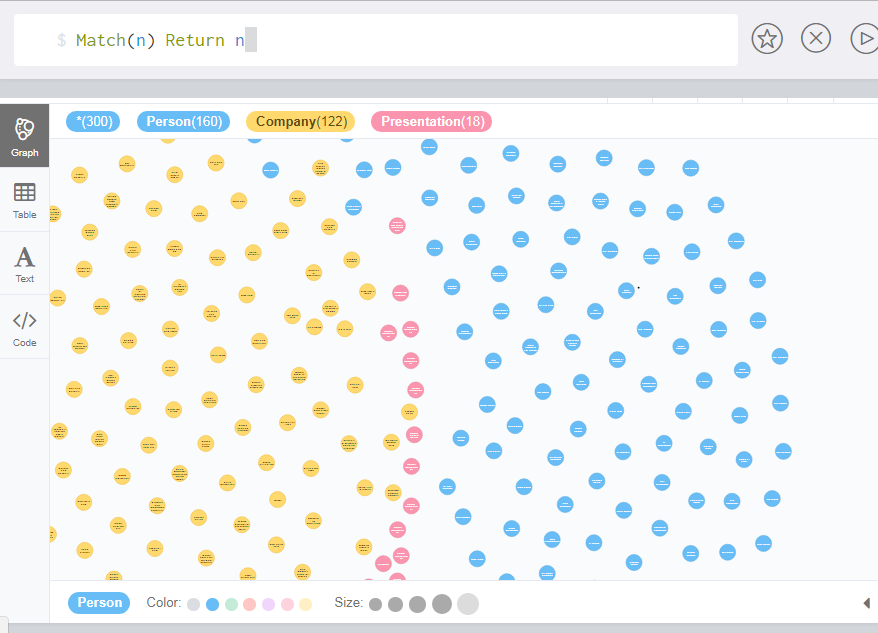


Рисунок 7 – Пример несвязанных узлов графовой СУБД

Для того, чтобы понимать, насколько важны связи в графовых СУБД, приведем пример (рисунок 8), в котором иллюстрируется, как из связи, выбранной в левой части интерфейса, можно получить больше дополнительной информации, кликнув на нужный узел (рисунок 9).

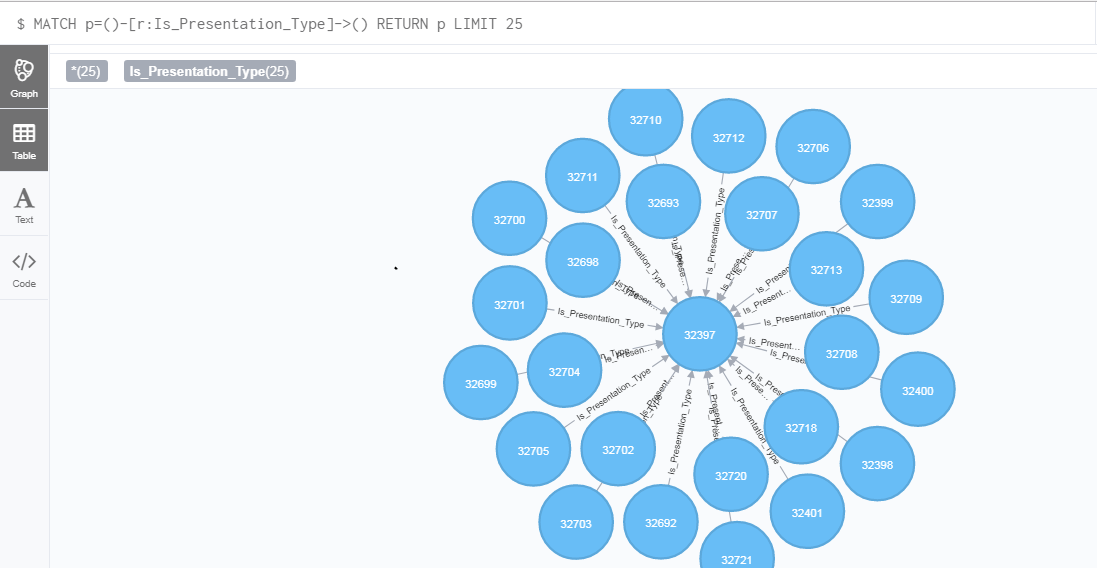


Рисунок 8 – Исходный граф

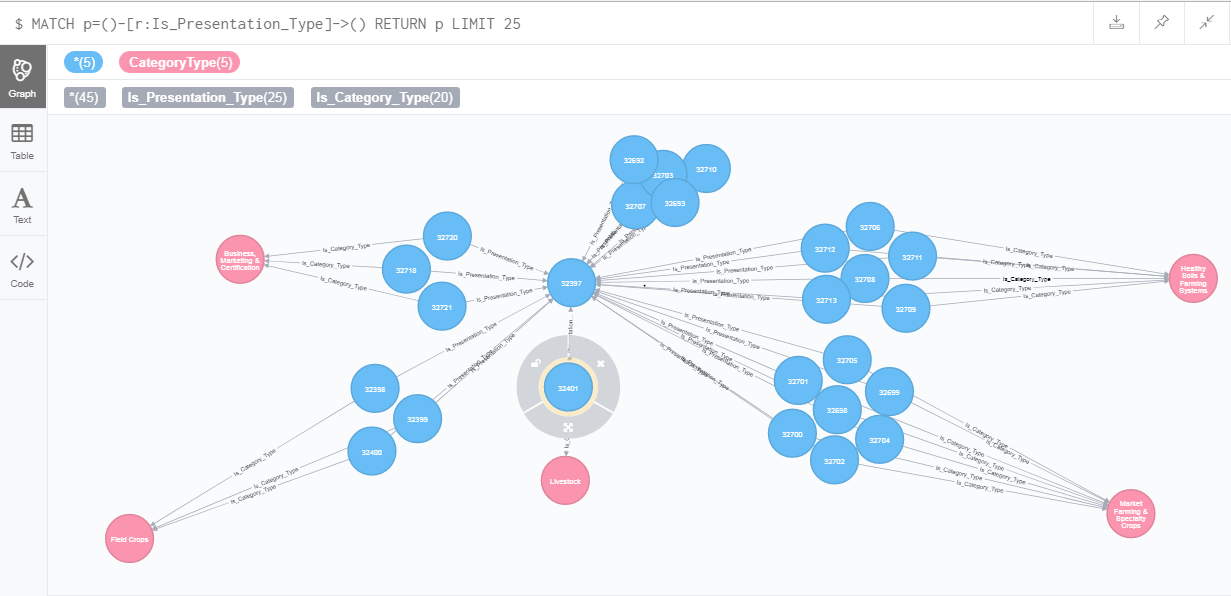


Рисунок 9 – Полученный граф

На рисунке 8 был выбран тип связи Is\_Presentation\_Type. Как уже отмечалось выше, графовые базы данных позволяют получать вложенную информацию, что непосредственно иллюстрирует рисунок 9.

Далее, в примерах приводятся более сложные запросы, с помощью которых можно извлечь различную информацию.

Рассмотрим пример 1.

Описание: количество городов, представители которых участвуют в конференции.

Запрос и результат изображены на рисунке 10.

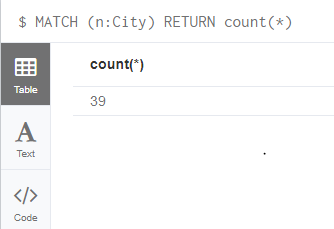


Рисунок 10 – Результат запроса 1

Рассмотрим пример 2.

Описание: количество представленных презентаций в базе данных.

Запрос и результат изображены на рисунке 11.

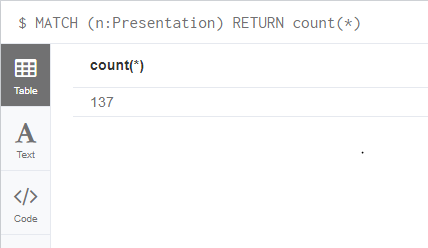


Рисунок 11 – Результат запроса 2

Как видно из приведенных выше примеров, работа с простыми командами похожа на составление SQL-запросов. Следующий шаг – получение данных через описание связей. В этом случае и начинает работать «графическое изображение». В этом запросе данные можно представить, как в графическом виде, так и в табличном.

Рассмотрим пример 3.

Описание: лица, у которых 2 и более презентаций на конференции. Будут выведены компании, в которых они работают.

а) Запрос можно увидеть на рисунке 12.

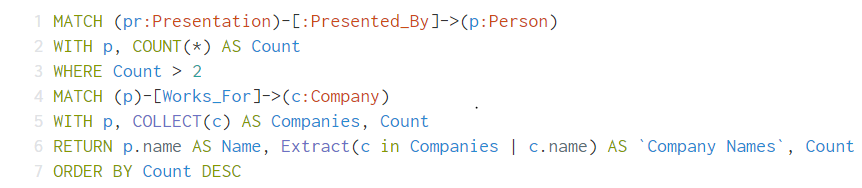


Рисунок 12 – Запрос 3(а)

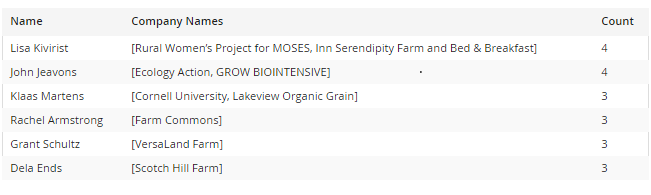


Рисунок 13 – Результат запроса 3(а)

б) Запрос можно увидеть на рисунке 14, а результат на рисунке 15.

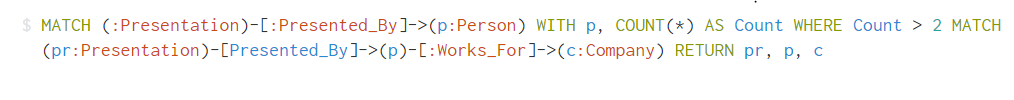


Рисунок 14 – Запрос 3(б)

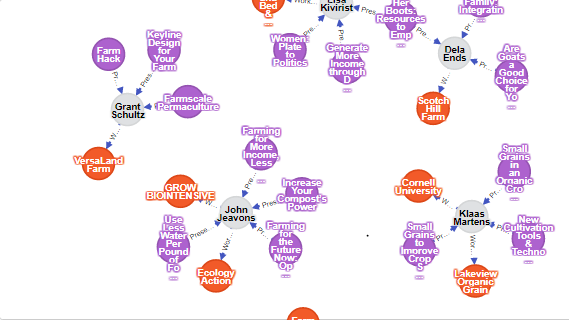


Рисунок 15 – Результат запроса 3(б)

Рассмотрим пример 4.

Описание: необходимо узнать, присутствуют ли на конференции люди с вашего города, это могут быть продавцы, представители компаний или презентаторы.

Запрос можно увидеть на рисунке 16, а результат на рисунке 17

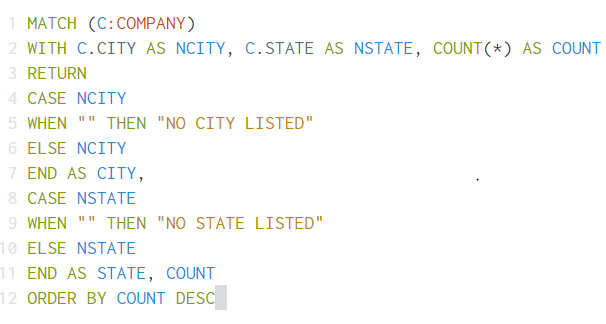


Рисунок 16 – Запрос 4

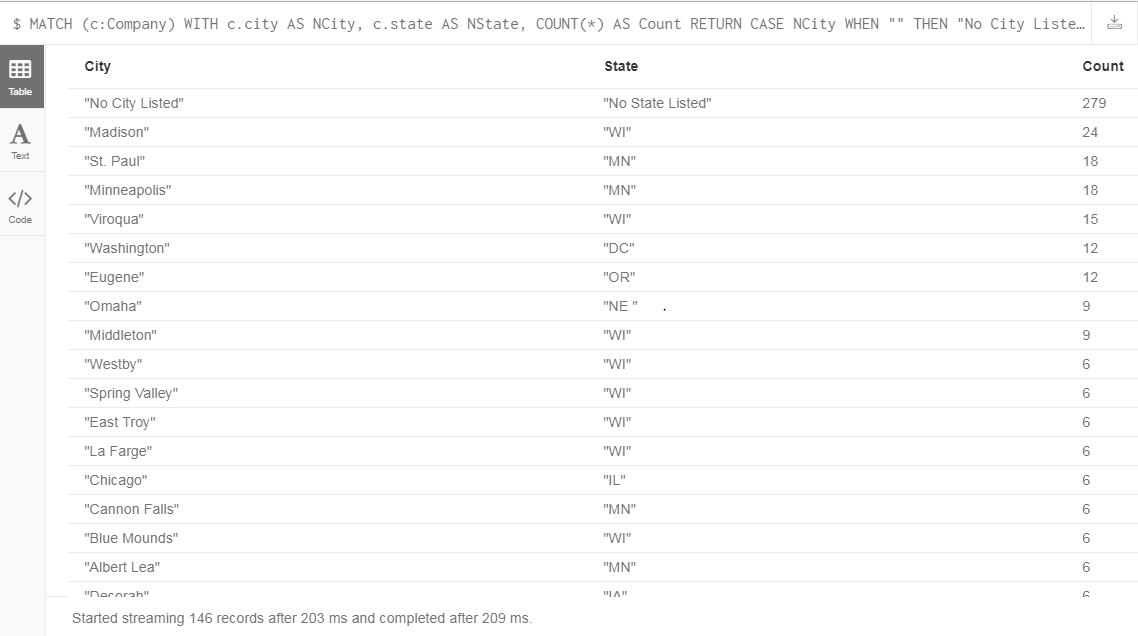


Рисунок 17 – Результат запроса 4

Проанализируем вторую СУБД. Данные представлены в файле csv и для наглядности частично изображены на рисунке 18 :

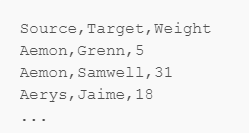


Рисунок 18 – Исходные данные

В таблице представлен список смежности символов и их количество взаимодействий по всему тексту. Используется простая модель данных (:Сharacter {name})-[:INTERACTS {weight}]->(:Character {name}). Узлы с меткой Character обозначают представления символов из текста с одним типом отношений INTERACTS, символы, которые взаимодействовали в тексте. Имя символа хранится как свойство name на узле и количество взаимодействий между двумя символами, как свойство weight отношений.

Используем конструкцию LOAD CSV в среде Neo4j для импорта информации в базу данных (рисунок 19).

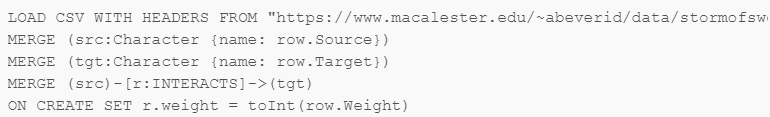


Рисунок 19 - Импорт информации в Neo4j

Первый запрос для данной СУБД будет заключаться в том, что необходимо вывести все ее ноды, запрос и результат изображены на рисунке 20 и рисунке 21.



Рисунок 20 – Запрос к Neo4j

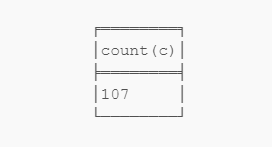


Рисунок 21 – Результат

В следующем примере рассмотрим диметр сети, т.е самый длинный путь в сети . Запрос можно увидеть на рисунке 22, а результат на рисунке 23

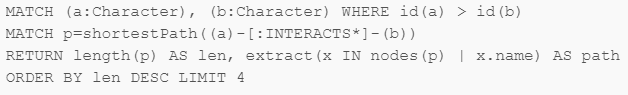


Рисунок 22 – Запрос к Neo4j

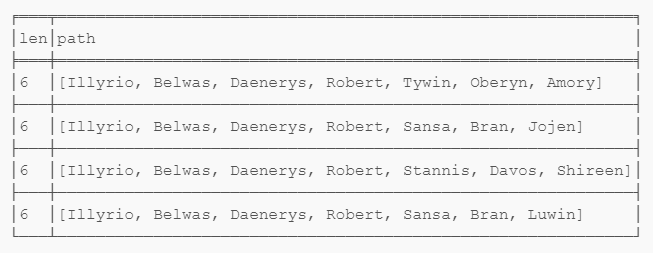


Рисунок 23 – Результат

Из полученного результата видно, что несколько путей имеют длину 6.

Мы также можем использовать функцию  shortestPath для нахождения кратчайшего пути между любыми двумя символами на графике. Найдем кратчайший путь от Catelyn Stark до Kahl Drogo (рисунок 24):

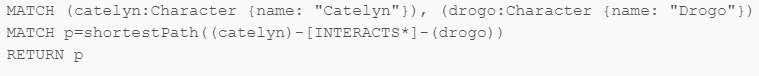


Рисунок 24 – Запрос к Neo4j

Результат можно увидеть на рисунке 25.

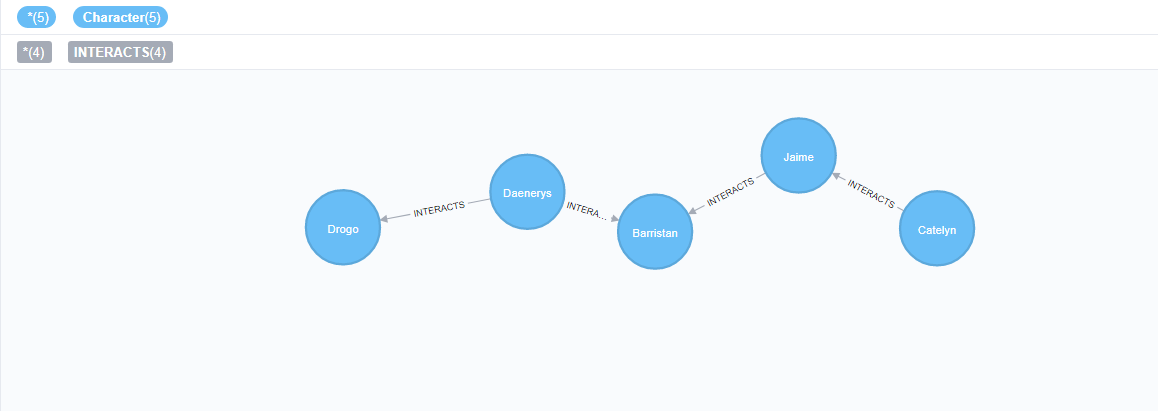


Рисунок 25 – Результат

Далее необходимо найти все кратчайшие пути, соединяющие Catelyn и Drogo. Мы можем найти их с помощью функции allShortestPaths (рисунок26). На рисунке 27 можно увидеть полученный граф:

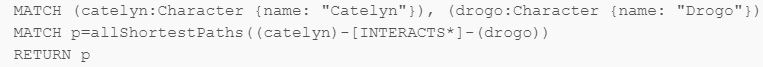


Рисунок 26 – Запрос

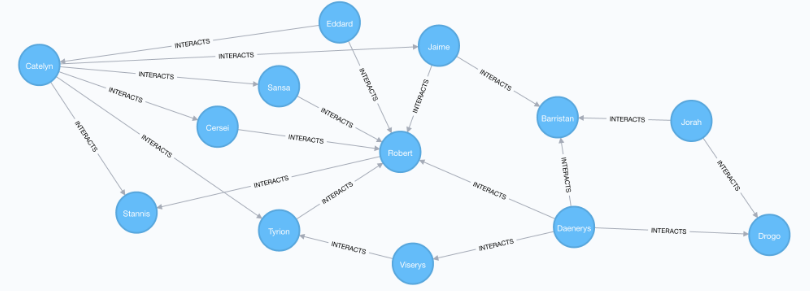


Рисунок 26 – Результат

Выше представленных базах данных были реализованы разные по сложности алгоритмы. Различия в них заключались в том, что в первой безе данных информация прописывалась вручную в среде Neo4j при помощи языка Cypher, где в итоге было получено 1727 строк, которые представляют собой: 663 узла, 6213 свойств и 1064 отношения.

Во второй же СУБД информация была импортирована по ссылке. В работе были построены различные запросы, которые помогают найти нужную информацию.

Используя средства языка Cypher, можно получить большое количество разной информации, в том числе и той, что показана в приведенных выше примерах. В запросах создавались узлы общей информации небольшого объема, но исходя из полученных данных, если грамотно построить базу данных, Neo4j дает возможность получить более глубокую и вложенную информацию.

ЗАКЛЮЧЕНИЕ

Данная курсовая работа, направлена на исследование графовой базы данных Neo4j. Была разработана небольшая база данных на языке Cypher, на примере которой удалось провести больше количество экспериментов, тем самым разобраться в архитектуре и базовых возможностях выбранной СУБД, оценить ее интерфейс и SQL-подобный язык Cypher.

В процессе написания работы возникла проблема, связанная с соединением базы данных Neo4j с языком программирования Python. Изначально хотелось создавать базу данных в Python и там же прописывать все интересующие нас запросы, но т.к многочисленные тонкости подключения не удалось настроить, хотелось бы данную идею реализовать в следующей работе, и оперируя уже большими данными, ведь за свое существование Neo4j накопила большое количество графических алгоритмов, которые доступны в виде открытой библиотеки процедур, тем самым показать, в каких случаях графовые базы данных уместны в использовании, а когда нет.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Ян Робинсон, Джим Вебер, Эмиль Эифрем Графовые базы данных: новые возможности для работы со связанными данными / пер. с англ. Р. Н. Рагимова; науч. ред. А. Н. Кисилев. – 2-е изд. – М.: ДМК Пресс, 2016. – 256 с.

2 Фаулер, Мартин, Садаладж, Прамодкумар Дж. NoSQL: новая методология разработки нереляционных баз данных. : Пер. сангл. - М.: ООО "И.Д.Вильяме", 2013. - 192 с.

3 Силен Дэви, Мейсман Арно, Али Мохамед Основы Data Science и Big Data. Python и наука о данных. –СПб.Питер, 2017.- 336 с.