**Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И.Ульянова (Ленина)**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 09.03.02– Информационные системы и технологии | |
| **Профиль** | Информационные системы и технологии | |
| **Факультет** | КТИ | |
| **Кафедра** | ИС | |
| *К защите допустить* |  | |
| Зав. кафедрой |  | Цехановский В.В. |

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

БАКАЛАВРА

Тема: **Разработка подсистемы поиска тематических данных в графовой базе данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | Порошин М.А. |
|  |  | *подпись* |  |  |
| Руководитель |  |  |  | Зорин К.М. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
| Консультанты |  |  |  | Иванов И.И. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Санкт-Петербург

2017**ЗАДАНИЕ**

**на выпускную квалификационную работу**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой ИС |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Цехановский В.В. |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Студент: | Порошин М.А. | | | |  | Группа | 3372 |
| Тема работы: Разработка подсистемы поиска тематических данных в графовой базе данных. | | | | | | | |
| Место выполнения ВКР: СПбГЭТУ «ЛЭТИ» | | | | | | | |
| Исходные данные (технические требования):  Разработка прототипа подсистемы, способной находить информацию в графовой базе данных по заданным ключам и представлять ее в понятном пользователю виде | | | | | | | |
| Содержание ВКР:  Создание прототипа модуля анализа информации о событиях, разделѐнное на исследование вопроса, разработку системы сбора информации и анализ данных через работу с графовой базой. | | | | | | | |
| Перечень отчетных материалов: текст ВКР, иллюстративный материал | | | | | | | |
| Дополнительные разделы: технико-экономическое обоснование проекта. | | | | | | | |
|  | | | | | | | |
| Дата выдачи задания | | | Дата представления ВКР к защите | | | | |
| «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. | | | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. | | | | |
|  | | |  | | | | |
| Студент | |  | | Порошин М.А. | | | |
| Руководитель | |  | | Зорин К.М. | | | |

**календарный план выполнения**

**выпускной квалификационной работы**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой ИС |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Цехановский В.В. |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент: | Порошин М.А. |  | Группа | 3372 |
| Тема работы: Разработка подсистемы поиска тематических данных в графовой базе данных. | | | | |

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование работ | Срок выполнения |
| 1 | Обзор литературы по теме работы | 00.00 – 00.00 |
| 2 | Наименование раздела | 00.00 – 00.00 |
| 3 | Наименование раздела | 00.00 – 00.00 |
| 4 | Наименование раздела | 00.00 – 00.00 |
| 5 | Оформление пояснительной записки | 00.00 – 00.00 |
| 6 | Оформление иллюстративного материала |  |

|  |  |  |
| --- | --- | --- |
| Студент: |  | Порошин М.А. |
| Руководитель |  | Зорин К.М. |

*(Уч. степень, уч. звание)***РЕФЕРАТ**

Пояснительная записка 00 стр., 00 рис., 00 табл., 00 ист., 00 прил.

Объектом разработки является система поиска тематических данных в графовой базе данных.

Цель работы – поиск в графовой базе событий по определенным ключам с целью ответа на вопрос пользователя и вывод ответа на человеческом языке.

Одним из основных источников хранения информации служат базы данных. Однако не все умеют писать к ним запросы. Более общим и удобным способом для получения информации была бы возможность задавать вопросы компьютеру на естественном языке. Такую задачу решают вопрос-ответные системы. В данной работе описана разработка прототипа модуля поиска данных по определенным ключам в графовой бд для системы, принимающей вопрос и дающей ответ на естественном языке. Задачей модуля является конструирование запросов к базе и интерпретация ответа. В качестве данных были выбраны события, для хранения данных была использована база neo4j, естественным языком выбран английский. Несмотря на то, что на некоторых шагах были получены неполные результаты, в целом рабочий прототип задуманной системы удалось создать.

**ABSTRACT**

Briefly (10-15 lines) the content of graduating work is specified

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 8 |
| 1 | Современное состояние вопроса | 9 |
| 1.1. | История вопроса | 10 |
| 1.2. | Современные проблемы | 15 |
| 1.3. | Пути решения проблем | 20 |
| 2 | Проектирование системы поиска тематических данных для графовой базы данных | 25 |
| 2.1. | База данных | 30 |
| 2.2. | Информационная модель | 35 |
| 2.3. | Модель, чертежи, электрические схемы | 40 |
| 3 | Результаты исследования (разработки) | 45 |
| 3.1. | Методика эксперимента | 50 |
| 3.2. | Результаты эксперимента | 55 |
| 3.3. | Сравнение и оценка результатов | 60 |
| 4 | Название дополнительного раздела | 65 |
| 4.1. | Название 1-го подраздела дополнительного раздела | 66 |
| 4.2. | Название 2-го подраздела дополнительного раздела | 66 |
| 4.3. | Название 3-го подраздела дополнительного раздела | 67 |
|  | Заключение | 68 |
|  | Список использованных источников | 69 |
|  | Приложение А. Название приложения | 72 |

*Наименования разделов приведены в качестве примера.*

**определения, обозначения и сокращения**

В настоящей пояснительной записке применяют следующие термины с соответствующими определениями:

UML – унифицированный язык моделирования.

Neo4j – реализованная на языке java система управления графовыми базами данных.

SQL – формальный непроцедурный язык программирования, применяемый для работы с данными в реляционных базах данных.

NoSQL – «не только SQL», ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL.

Cypher – декларативный NOSQL язык для работы с данными в графовых базах данных.

QA – (Question answering system) информационная система c естественно языковым интерфейсом, способная принимать вопросы и отвечать на них на естественном языке.

**Введение**

С давних пор информация играет важную роль в нашей жизни; нам нужно иметь определенную информацию, перед тем как принять решение. Недавно с развитием технологий, таких как компьютеры, гаджеты и интернет, у нас появилась возможность получать практически где угодно и какую угодно информацию. В современном мире информации становится настолько много, что человечество создает все новые способы ее хранения и поиска с целью выбирать среди данных нужные.

Одним из главных источников информации служат базы данных. Они содержат набор связанных данных. Для того чтобы извлечь информацию из БД нужно сформулировать запрос понятный для компьютера. Однако не все умеют писать такие запросы. Более общим и удобным способом для получения информации была бы возможность задавать вопросы компьютеру на естественном языке, однако, такие вопросы для компьютера будут всего лишь последовательностью символов. Поэтому в настоящее время существуют и создаются различные системы с вводом и выводом информации на естественном языке, облегчающие поиск информации, дабы восполнить этот пробел. Такие системы должны уметь правильно и точно интерпретировать естественный язык в запрос, избегая чрезмерной траты программных ресурсов. При использовании так называемых сложных запросов (многотабличных SQL – запросов) на больших объемах данных, количество затрачиваемых на поиск ресурсов резко возрастает, что плохо сказывается на производительности системы [1].

На данный момент существуют вопрос – ответные или Question Answering (QA) системы, способные принимать вопросы и отвечать на них на естественном языке. Она позволяет пользователю задавать фактические вопросы на естественном языке (Где находится лувр?) и получить точный конкретный ответ (В Париже, Франция), в отличие от варианта с поисковиком, который выведет страницу со ссылками, среди которых пользователь будет вынужден сам искать необходимую информацию. Такие системы должны где-то хранить данные и иметь возможность добавлять новые, без необходимости менять модель данных. К тому же, для получения информации, по возможности, стоит избегать сложных запросов, поскольку система работает с большими объемами данных и это бы отрицательно сказывалось на ее работе [2].

Большинство баз данный для вопросно-ответных систем были реализованы на реляционных субд. С появлением графовых баз данных и субд для них, появились варианты реализации систем с их использованием как основной базы данных или как дополнительной базы (необходимой например для построения графа обучения системы) [3]. Графовая база данных представляет собой NoSQL бд, использующую графовую модель данных – обобщение сетевой модели данных. Элементами такой базы являются узлы и связи между этими узлами. В общем смысле такая структура позволяет моделировать практически какие угодно сценарии [4]. Запросы осуществляются посредством языка cypher, декларативного языка для графовых баз данных. Такие базы удовлетворяют всем вышеописанным требованиям для базы данных вопрос-ответной системы [3].

Объектом данной работы является поиск информации в базе данных в виде, удобном для человека. Предметом же является поиск тематических данных (в данном случае данных, полученных из новостных статей) в графовой бд.

Целью работы является разработка подсистемы поиска тематических данных для графовой базы данных. Для достижения цели необходимо решить ряд поставленных задач:

1. Проанализировать текущее состояние вопросно-ответных систем и проблемы генерации запросов к базе данных для таких систем.
2. Спроектировать подсистему поиска тематических данных для графовой базы данных.
3. Реализовать прототип данной подсистемы.

**Глава 1. СОВРЕМЕННОЕ СОСТОЯНИЕ ВОПРОСА**

* 1. **История вопроса**

Попытки создания интерфейсов к базам данных, способных понимать естественный язык, начались еще в конце прошлого века. Естественный язык к базам данных NLIDB (Natural Language Interface to DataBases) представляют собой системы, которые переводят запросы с человеческого языка в запрос понятный БД. NLIDB может рассматриваться как классическая задача в области обработки естественного языка. Несмотря на то, что самое раннее исследование началось с конца шестидесятых годов, исследования продолжаются, ввиду того что пока не удалось создать систему полностью на естественном языке, интуитивно понятную даже для пользователя, далекого от таких вещей. Создание полноценной NLIDB несет за собой множество полезных следствий, таких например как иное восприятие пользователями баз данных, удобства работы с ними, а следовательно их используемость в целом [5].

Потомками NLIDB стали вопросно-ответные системы Question Answering (QA), способные принимать вопросы и отвечать на них на естественном языке. Вопросно- ответная система многое переняла от NLIDB и является более современной. Она позволяет пользователю задавать фактоидные вопросы на естественном языке (Where is the Louvre Museum located?) и получить точный конкретный ответ (In Paris, France), в отличие от варианта с поисковиком, который выведет страницу со ссылками, среди которых пользователь будет вынужден сам искать необходимую информацию. В этом отношении QA можно рассматривать как следующее поколение удобных инструментов для поиска в больших объемах текстовых данных, например в сети интернет. Кроме того QA системы и их отдельные модули могут быть использованы для создания ботов для социальных сетей или интерфейсы для управления устройствами или например умным домом.

Современные QA-системы построены на принципах NLP (Natural Language Processing), что позволяет им образовывать классы интеллектуальных систем информационного поиска. Системы имеют как общие принципы обработки естественного языка, так и различия в способе внутреннего представления семантического содержимого и в алгоритмах для приведения текста на естественном языке к внутреннему представлению системы. Следовательно, результаты работы системы напрямую зависят от реализуемых алгоритмов компьютерной лингвистики. Кроме того, для большинства национальных языков таковая задача установки семантических ролей не поставлена в отличие от английского, где подготовлена существенная база аннотированных текстов. Однако, кроме английского встречаются и реализации на национальных языках и мульти-язычные платформы (например система KbQAS имеет особую архитектуру, позволяющую модернизацию системы для взаимодействия со многими естественными языками. Изначальный язык системы KbQAS – вьетнамский) [8].

В общем случае вопросно-ответные системы представляют собой информационную систему, содержащую в себе комплекс интеллектуальных систем и набор справочный материалов. Вопрос-ответные системы могут быть как узкоспециализированными так и общего назначения, старающиеся ответить на любой вопрос пользователя [5],[6].

Среди известных систем можно выделить IBM Watson – суперкомпьютер, оснащённый вопросно-ответной системой искусственного интеллекта или Siri (Speech Interpretation and Recognition Interface) —персональный помощник и вопросно-ответная система, программный клиент которой входит в состав програмных продуктов компании Apple. Из отечественных аналогов стоит отметить систему «Нигма.РФ», первую в русском интернете интеллектуальную метапоисковую система, созданую при поддержке факультетов ВМК и психологии МГУ, а также Стэнфордского университета.

Вопрос – ответные системы зачастую имеют дело с большими объемами произвольно связанных данных, которые необходимо грамотно хранить и иметь механизмы для поиска, обладающие большой производительностью. Большинство подобных систем было реализовано с использование реляционных баз данных. Однако существовали системы, пользующиеся сетевой моделью данных. Позднее, с появлением графовой модели данных (часто рассматриваемой как обобщение сетевой) и графовых субд стали появляться аналоги, также использующие графы для поиска информации или машинного обучения через диалог с пользователем. Одним из вариантов решения данной задачи может стать использование графовых баз данных [6],[4].

* 1. **Современные проблемы**

Вопросно-ответной системе необходимо хранить в базе данных множество объектов с нефиксированным количеством полей и различными связями между ними. Новостные события, о которых идет речь в данной работе, имеют несложную структуру, но при этом вариативность связей и атрибутов, как у сущностей, так и у самих связей. Именно в этом и заключается сложность в реализации модели с помощью реляционных баз данных. Важно чтобы модель была гибкой. А значит, возможность добавлять большое количество объектов (возможно с отличным набором полей и с новыми видами отношений) за короткое время и без вреда для самой модели. Данная задача имеет сложности при реализации с помощью реляционных баз данных, потому что каждый новый добавляемый объект может не вписываться ни в одну из таблиц или может быть связан с уже имеющимися в таблице объектами типом связи, не предусмотренным в модели данных. Реляционные субд требует четко описанную структуру данных до начала работы [6].

Также немаловажно, чтобы система имела высокую производительность, т.к. осуществляется работа с большими объемами данных. При реализации базы данных в виде SQL таблиц и генерации запросов к базе, проблемой могут стать так называемые «сложные запросы». Такие SQL запросы объединяют несколько таблиц, при больших объемах данных, каждая дополнительно присоединенная таблица существенным образов влияет на потребление памяти, что в свою очередь пагубно сказывается на производительности всей системы. Избежать сложных запросов можно было бы, используя заранее заготовленные процедуры. Но подготовить процедуры заранее не представляется возможным, так как запросы генерируются программой, относительно вопроса, который был задан пользователем [5],[1].

Для более качественного анализа вопросов и высокой точности ответов в вопросно-ответных системах часто используют онтологии. В общих чертах под онтологией понимается система понятий некоторой предметной области, которая представляется как набор сущностей, соединенных различными отношениями. Онтологии используются для формальной спецификации понятий и отношений, которые характеризуют определенную область знаний. Преимуществом онтологий в качестве способа представления знаний является их формальная структура, которая упрощает их компьютерную обработку [7]. Онтологии дают возможность системе отвечать на вопросы, требующие некоторые знания в предметной области вопроса.

Описанные выше условия необходимы для работы с базой данных вопросно-ответной системы. А именно для генерации точных и запросов к базе данных, быстродействия и возможности добавления новых данных [8].

* 1. **Пути решения проблем**

Альтернативой использованию реляционной базы данных может стать использование графовых бд бд [2]. Графовая база представляет собой разновидность NoSQL бд с реализацией сетевой модели в виде графа и его обобщений. Компоненты базы – узлы и ребра, которые могут быть дополнены набором полей. Этот вид баз данных отличается от реляционных и, в некоторый случаях, имеет рад преимуществ перед ними.

В реляционных базах данных ссылки на другие строки и таблицы указаны ссылками на их (первичные) ключи через внешние ключи столбцов.

Например, если в базе много отношений многие-ко-многим, то имея реляционную базу, вам придется использовать join или создавать распределительную таблицу с внешними ключами на обе связанные таблицы.

При большом количестве таких соединений, каждое последующее соединение может существенно увеличивать затрату ресурсов памяти программы, к тому же, соединения происходят во время запроса к базе, что может также замедлить работу программы или даже стать причиной сбоя. Такие сложные запросы, соединяющие информацию из многих таблиц, обычно пишутся в виде функций. Такой подход позволяет сэкономить ресурсы, однако не приемлем для решения данной задачи. Пользователь задает вопрос на естественном языке, а не выбирает из списка предложенных, это значит, что для поиска ответа, программа должна динамически сгенерировать запрос к базе данных и написать процедуры заранее невозможно.

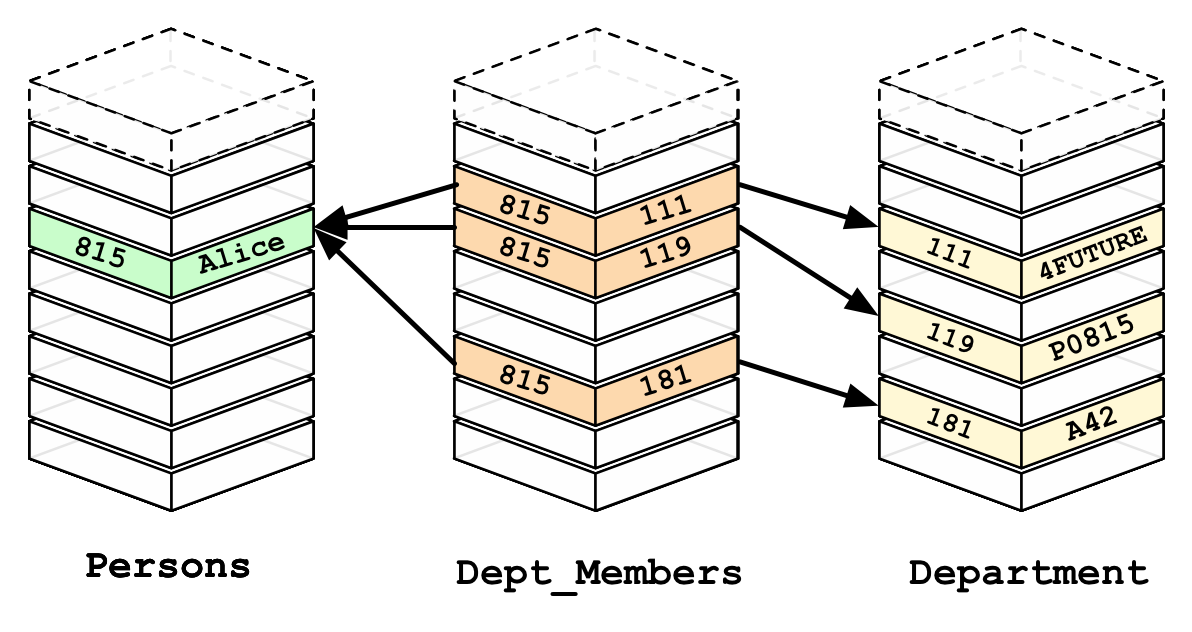


Рисунок 1.1 - соединение таблиц в реляционной базе данных.

В графовых базах отношения являются гражданами первого класса, в отличие от реляционных, в которых необходимо сделать выводы о связях между объектами и ввести определенные ключи. Каждый узел (субъект или атрибут) в модели базы данных графа непосредственно и физически содержит список записей отношений, которые представляют его связи с другими узлами. Эти записи-отношения сгруппированы по типу и направлению может содержать дополнительные атрибуты. Всякий раз, при запуске эквивалента операции присоединения, база данных просто использует этот список и имеет прямой доступ к подключенным узлам, затрачивая на это меньше ресурсов, чем у реляционных баз на операции присоединения. Эта способность обеспечивает графовым бд выигрыш в производительности, особенно на больших объемах данных.

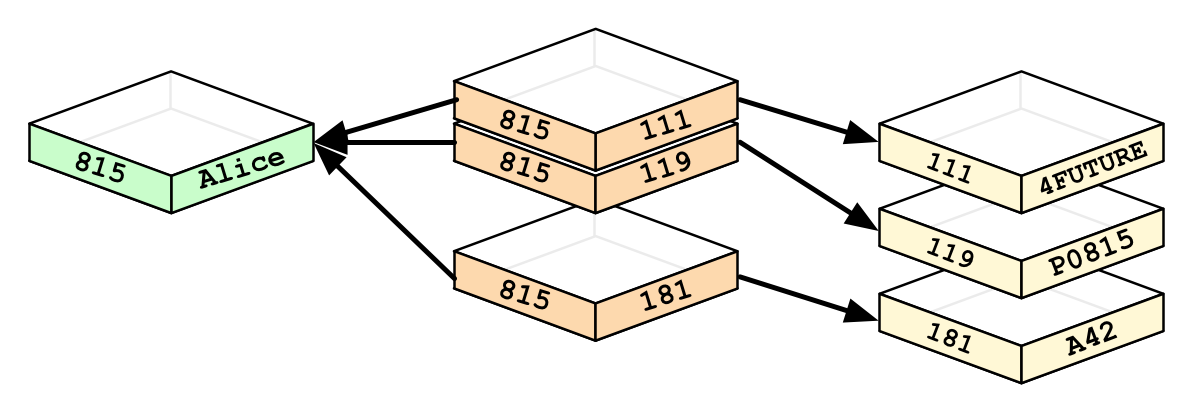


Рисунок 1.2 – модель с графовой базе данных.

В отличие от модели данных на рисунке 1.1 (реляционная), модель на рисунке 1.2 выглядит гораздо проще и выразительнее. Графовые базы данных поддерживают гибкую, мелкозернистую модель данных. Это позволяет добавлять даже существенные изменения, без необходимости модернизировать систему и модель данных [1]. Это свойство также необходимо для базы данных вопросно-ответной системы, поскольку в систему постоянно необходимо загружать новые данные о событиях, которые могут быть связаны с уже имеющимся в базе особым образом [6].

* 1. **Описание графовых СУБД**

Большинство графовых баз данных имеют схожую модель данных, представленную на рисунке 1.3. Узлы, имеющие определенный тип и свойства, соединяются связями – отношениями, которые также могут иметь свойства [4]. В таблице 1 приведены описания некоторых существующих, популярных, развивающихся графовых субд [9].

При всех схожести модели графовых субд, они имеют массу отличий. Различные субд имеют разные языки запросов, или же могут не иметь специального языка запросов вообще. К примеру, neo4j использует для запросов язык cypher. С другими же взаимодействие может происходить с помощью SPARQL – языка запросов а также протокола передачи запросов и ответов для данных, представленных по модели RDF. Или с помощью драйверов для языка программирования. Существуют также специальные языки запросов, которые могут взаимодействовать с разными базами данных. Например, Gremlin – это предметно-ориентированный язык, для работы с графами. Он поддерживает работу сразу с несколькими субд, такими как OrientDB, Neo4j или Titan. Ниже представлены примеры синтаксиса на каждом из упомянутых языков на примере запроса: вывести электронную почту каждого человека старше 25 лет с фамилией Anderson.

SPARQL:  
*PREFIX ex: <http://example.com/example/>*

*SELECT ?email*

*WHERE {*

*?person a ex:Person.*

*?person ex:surname “Anderson”.*

*?person ex:age ?age.*

*?person ex:email ?email.*

*FILTER (?age > 25). }*

Cypher:  
*MATCH (person:Person {surname: 'Anderson'})  
WHERE person.age >25*

*RETURN person*

Gremlin:  
*gremlin> g.V().has ('person', 'surname', 'Anderson').filter(it.age > 25).email*  
SPARQL и Cypher, в отличие от gremlin, напоминают по структуре привычный SQL, что делает их более интуитивно понятными и простыми в освоении [1],[10].

По таблице 1.1 заметно, что различается поддержка драйверов для языков программирования, вид лицензии и имеются некоторые индивидуальные особенности субд. Также имеют различия система хранения данных и быстродействие. Эти факторы необходимо учесть при выборе субд, при проектировании подсистемы поиска для QA системы.

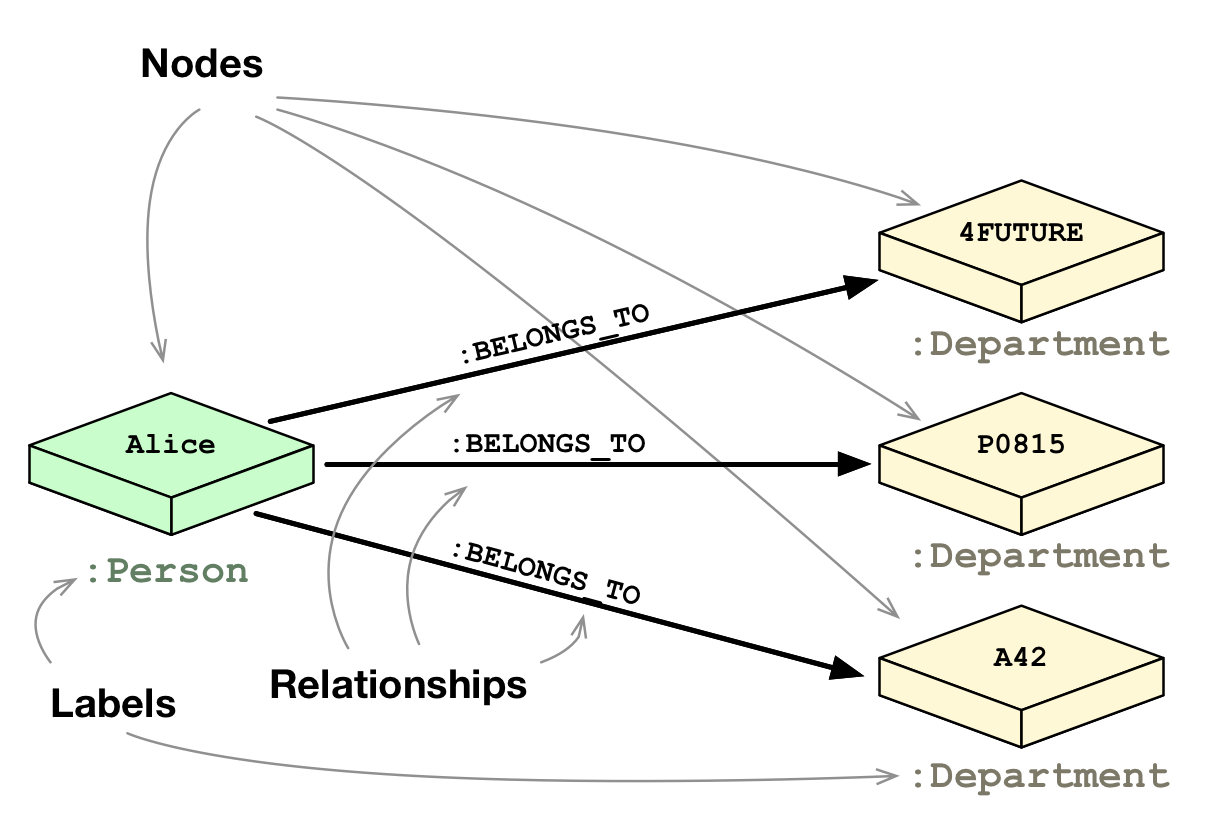


Рисунок 1.3 – узлы и отношения в графовой бд

Таблица 1.1 Сравнительное описание некоторых популярных графовых субд

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| СУБД | Версия | Лицензия | Работа с языками | Описание |
| [Allegro Graph](https://en.wikipedia.org/wiki/AllegroGraph) | 5.1  (05.2015) | [Частная](https://en.wikipedia.org/wiki/Proprietary_software) | [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)), [C](https://en.wikipedia.org/wiki/C_(programming_language)),  [Common Lisp](https://en.wikipedia.org/wiki/Common_Lisp), [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) | [Resource Description Framework](https://en.wikipedia.org/wiki/Resource_Description_Framework) (RDF) и графовая база данных |
| [ArangoDB](https://en.wikipedia.org/wiki/ArangoDB) | 3.1.12 (02. 2017) | Свободная,  [Apache 2](https://en.wikipedia.org/wiki/Apache_License" \l "Version_2.0" \o "Apache License) | [C++](https://en.wikipedia.org/wiki/C%2B%2B),  [JavaScript](https://en.wikipedia.org/wiki/JavaScript) | Одна из самых популярных баз данных под свободной лицензией и открытым исходным кодом |
| GraphBase | 1.0.03b | Частная, коммерческая | [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) | Графовая база данных, поставляемая с набором инструментов от FactNexus. |
| O[rientDB](https://en.wikipedia.org/wiki/OrientDB) | 2.2.0  (05. 2016) | Community Edition -[Apache 2](https://en.wikipedia.org/wiki/Apache_License#Version_2.0), Enterprise Edition - коммерческая | [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) | Документо-ориентированное хранилище гибрид типа граф написанное на Java. Включает в себя разные режимы работы. Имеет HTTP-rest и Json-API |
| [Neo4j](https://en.wikipedia.org/wiki/Neo4j) | 3.1.1 (January 2017)[[20]](https://en.wikipedia.org/wiki/Graph_database#cite_note-20) | [GPLv3](https://en.wikipedia.org/wiki/GNU_General_Public_License) корпоративная и коммерческая версия а так же дополнительные опции для них. | [Java](https://en.wikipedia.org/wiki/Java_(programming_language)),  [.NET](https://en.wikipedia.org/wiki/.NET_Framework), [JavaScript](https://en.wikipedia.org/wiki/JavaScript)[Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)) | Графовая база данных с открытым исходным кодом. Имеет расширения для коммерческого использования Поставляется с Web-API, с возможностью визуализации и администрирования данных. Является самой популярной графовой СУБД по данным на январь 2017 |

**Глава 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ ПОИСКА ТЕМАТИЧЕСКИХ ДАННЫХ ДЛЯ ГРАФОВОЙ БАЗЫ ДАННЫХ.**

Тематические данные - данные, отнесенные к конкретной предметной области. В данном случае в качестве предметной области была выбрана информация, которую можно получить из новостных статей. А значит объектами базы (или узлами, учитывая, что мы используем графовую базу данных) будут являться люди, организации, локации, события и отношения между ними.

Для того чтобы разработать прототип системы хранения поиска и представления информации, необходимо, опираясь на первую главу, разобраться в ней на всех этапах построения и составить задание, учитывающее особенности системы и процесса ее построения. Основные шаги включают в себя выбор, в каком виде система будет получать входные параметры, создание информационной модели базы данных, алгоритмов поиска тематических данных и представления их для пользователя. Данная база должна содержать информацию о множестве событий, людях и связях между ними. В целом процесс формирование задания будет проходить образом, описанным далее.

* 1. **База данных**

Как уже было отмечено, выбор базы данных должен соответствовать поставленной задаче и требованиям. Данные о событиях, людях, местах и прочие данные, которые можно получить из новостных статей могут быть разнообразными, связи между ними произвольными, а значит, база должна удовлетворять следующим требованиям:

* Высокая производительность при больших объемах данных и связей между ними, в основном для сложных запросов.
* Возможность вносить крупные изменения без необходимости менять модель данных.
* База не должна быть сложной в заполнении и обслуживании (так как предполагается, что новые данные в базу будет заносить программа, на основе новостных статей)

Всем вышеперечисленным условиям соответствуют графовые бд. Среди графовых баз была выбрана субд Neo4j, так как по заявлениям разработчиков, ее логическая модель в то же время является и физической, что позволяет быстро вносить изменения в базу. Язык субд - Cypher представлением похож на SQL и прост в изучении, поддерживаются ряд API и сообщество, которое поддерживает создание справочных материалов и тематических форумов. Что должно облегчить освоение данного инструмента. Кроме всего перечисленного, Neo4j является достаточно надежной и производительной субд [1].

Рассмотренные альтернативы Neo4j не обладали таким количеством справочных материалов и удобством использования и представления информации.

**2.2 Информационная модель**

В качестве предметной области были выбраны данные которые можно получить из новостных ресурсов, такие как события и люди, которые в них участвуют. Построение информационной модели подразумевает грамотное представление этих данных в виде графа.

Полученная из статей информация может включать себя данные событиях, людях, имеющих к ним отношение, месте, в котором они могли происходить и подобное. Эти данные можно представить в виде объектов базы данных, в данном случае в виде узлов графа. Объекты связаны между собой отношениями. Например, два человека могут быть связаны между собой отношением «знакомы» и человек и событие отношением «участвует» Отношения между объектами в данном случае можно представить как ребра графа - отношения между узлами. Узлы графа можно разделить на типы, такие как «событие», «человек», «место» и подобные. Пример представлен на рисунке 2.1. Каждый узел может иметь больше одного отношения с другими узлами. Например, Человек 1 может быть «знаком» с Человеком 2, участвовать в Событии 1, быть «организовать» Событие 2 и «встретиться» с Человеком 3. Количество связей и типы связей могут быть произвольными. Также любой узел или связь могут иметь свойства. Например, у человека это может быть должность или дата рождения, а у связи «встретился» дата или место встречи.



Рисунок 2.1 – узлы и отношения в графовой базе данных

**2.3 Описание подсистемы поиска**

В этом разделе будет описана разрабатываемая подсистема поиска данных для вопросно-ответной системы. Предполагается, что система будет состоять из 3 подсистем:

1. модуля извлечения информации о событиях из новостных статей. Данный модуль отвечает за извлечение и запись данных из статьи в базу в виде схемы субъект - предикат - объект.
2. Подсистемы естественно языкового интерфейса, задача которой заключается в разборе вопроса, заданного пользователем. Вопрос разбирается по схеме субъект - отношение - объект. Каждому элементу схемы соответствует набор токенов, полученных их вопроса. Например, для связи «встретился» токенами могут быть дата или место встречи, а для субъекта – участника встречи токенами могут быть имя и должность. Также система определяется тип вопроса (это понадобится для того чтобы понимать в каком виде давать ответ) и формирует из этих данных структуру, с которой уже будет взаимодействовать подсистема поиска.
3. Подсистема поиска данных. На основе информации, полученной от подсистемы естественного языка, формирует запрос к базе данных, формирует ответ и выводит его в понятном для пользователя виде.

В работе более детально разобрана подсистема поиска, ее диаграмма деятельности представлена на рисунке 2.2. Подсистема поиска получает от подсистемы естественного языкового интерфейса вопрос и структуру, содержащую тип вопроса, типы узлов для субъекта и объекта, тип связи и также набор токенов для них. На основе полученных данных генерируется вопрос к базе данных. Если запрос валидный, то на основе информации генерируется ответ на понятном человеку языке, в противном случае система выдает ошибку.

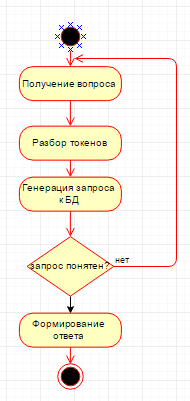


Рисунок 2.2 – диаграмма деятельности для подсистемы поиска данных

Модель структуры данных на вход системе представлена на рисунке 2.3.

Таким образом, данные в базе представлены как частный случай RDF модели. RDF (Resource Description Framework) – это модель представления мета-данных, разработанная консорциумом всемирной паутины. Она представляет собой утверждения о ресурсах, удобные для машинной обработки. Является стандартной моделью обмена данными. Триплет RDF представлен на рисунке 2.2.

**Список использованных источников**

1. From Relational to Neo4j [Электронный ресурс] // neo4j.  
   URL: <https://neo4j.com/developer/graph-db-vs-rdbms/> (дата обращения: 14.05.2017)
2. How to Build Templates for RDF Question/Answering —An Uncertain Graph Similarity Join Approach [Текст] / Weiguo Zheng, Lei Zou, Xiang Lian, Jeffrey Xu Yu, Shaoxu Song, Dongyan Zhao. [SIGMOD '15](http://dl.acm.org/www.sigmod2015.org), 2015, 1824 стр.
3. Constructing knowledge graph from unstructured text [Текст] / Kundan Kumar, Siddhant Manocha. 2015,17 стр.
4. Survey of graph database models [Текст] / Renzo Angles, Claudio Gutierrez. ACM Comput. Surv. 40, 1, Article 1, 2008, 39 стр.
5. Natural Language Interfaces to Databases – An Introduction [Текст] / G.D. Ritchie, P. Thanisch. Department of Artificial Intelligence, University of Edinburgh, 1995, 50 стр.
6. Natural language interface to databases [Текст] / Yohan Chandra. UNIVERSITY OF NORTH TEXAS, 2006, 62 стр.
7. Онтологии как системы хранения знаний [Текст] / Н.С. Константинова, О.А. Митрофанова. 54 стр.
8. Ripple Down Rules for Question Answering [Текст] / Dat Quoc Nguyen, Dai Quoc Nguyen, Son Bao Pham. IOS Press, 2015, 22 стр.
9. DB-Engines Ranking of Graph DBMS [Электронный ресурс] / db-engines.  
   URL: <https://db-engines.com/en/ranking/graph+dbms> (дата обращения: 14.05.2017)