МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Информационные системы и технологии корпоративного управления
(направленность (профиль))

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему «Исследование архитектур построения и алгоритмов работы систем хранения и анализа больших данных»

Студент М.А. Серебряков _____
(И.О. Фамилия) (личная подпись)

Научный руководитель А.В. Очеповский _____
(И.О. Фамилия) (личная подпись)

Руководитель программы д.т.н., доцент, С.В. Мкртычев _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____» _____ 20_____г.

Допустить к защите
Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____» _____ 20_____г.

Тольятти 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Большие данные - это общий термин для любого набора наборов данных, настолько больших или сложных, что становится трудно обрабатывать их с использованием традиционных методов управления данными, таких как, системы управления реляционными базами данных. Долгое время реляционные системы управления базами данных (РСУБД) считались универсальным решением, но экспоненциально растущие объемы, скорость и неоднородность данных показали непригодность РСУБД для использования в системах хранения и анализа больших данных. Наука о данных предполагает использование методов для анализа огромных объемов данных и извлечения содержащихся в них знаний.

В соответствии с национальной программой «Цифровая экономика Российской Федерации» количество опорных центров обработки данных должно быть увеличено с двух в 2018 году к восьми в 2024 [1, 2]. Общий объем финансирования программы составляет 1634,9 млрд. руб. Выполнение указанной программы основано на структуре из шести федеральных проектов, среди которых присутствуют следующие:

- Информационная инфраструктура;
- Информационная безопасность;
- Цифровые технологии;
- Цифровое государственное управление.

В соответствии с паспортами указанных проектов предполагается значительное увеличение объемов хранимых и анализируемых данных.

Тольяттинский государственный университет в 2018 году включён в перечень организаций, рекомендованных для присвоения статуса Федеральной инновационной площадки. Проект «Умный университет», предложенный ТГУ, признан одним из наиболее значимых инновационных образовательных проектов в 2018 году и предполагает цифровую трансформацию процессов вуза [62]. Кроме того, программа трансформации университета предусматривает

широкое внедрение цифровых технологий и становление ТГУ как цифрового университета. Однако, в университете отсутствует достаточный опыт практического использования технологий хранения и анализа больших данных.

Таким образом, **актуальность магистерской работы** обусловлена необходимостью апробации технологий и систем хранения и анализа больших данных.

Объектом исследования процессы хранения и анализа больших данных.

Предметом исследования является системы хранения и анализа больших данных.

Целью работы является апробация этапов доступа, очистки и загрузки в системы хранения и анализа больших данных и выдача практических рекомендаций по реализации указанных этапов.

Для достижения поставленной цели были выделены следующие задачи:

- проанализировать процесс анализа и хранения больших данных;
- проанализировать подходы к построению систем хранения и анализа больших данных;
- провести исследование способов выполнения этапов доступа, очистки и загрузки данных с использованием Java технологий;
- дать практические рекомендации по реализации и выполнению этапов доступа, очистки и загрузки в систему хранения и анализа больших данных с использованием Java технологий.

Методы исследования. В процессе исследования были использованы следующие методы: методы системного анализа, экспертной оценки, методы объектно-ориентированного анализа и проектирования.

Новизна исследования заключается в разработке практических рекомендаций по реализации этапов доступа, очистки и загрузки данных в систему хранения и анализа больших данных.

Практическая значимость исследования заключается в возможности практического применения апробированных технологий хранения и анализа больших данных.

Теоретической основой диссертационного исследования являются научные труды российских и зарубежных ученых, занимающихся проблемами хранения и анализа больших данных.

На защиту выносятся:

- результаты анализа способов построения систем хранения и анализа больших данных;
- результаты апробации реализации этапов доступа, очистки и загрузки в системы хранения и анализа больших данных.

Публикации. По результатам проведённого исследования подготовлена к печати научная статья.

Диссертация состоит из введения, трех разделов, заключения, списка используемой литературы и приложений.

В первом разделе проанализирована актуальность применения технологий хранения и анализа больших данных. Описаны и проанализированы особенные характеристики больших данных. Рассмотрен процесс Data Science применительно к большим данным.

Во втором разделе выполнен анализ архитектурных и программных решений при построении систем хранения и анализа больших данных. Произведены обзор и анализ применимости NoSQL баз данных.

Третий раздел посвящена апробации использования стандартных и сторонних библиотек языка Java при реализации этапов доступа, очистки и загрузки в системы хранения и анализа больших данных.

В заключении приводятся результаты исследования.

Работа изложена на 86 страницах, включает 44 рисунка и 5 приложений.

Магистерская диссертация выполнена по заданию Центра IT Student Тольяттинского государственного университета.

1 ПРОЦЕСС АНАЛИЗА БОЛЬШИХ ДАННЫХ

1.1 Роль больших данных в развитии Data Science

Развитие современного общества и технологий на современном этапе неразрывно связано не только с информатизацией новых областей человеческой деятельности, а сколько с повсеместным внедрением технологий изучения и анализа данных для выработки грамотных управленческих решений.

Большое вниманию развитию этой задачи отводится как во всем мире, так и в Российской Федерации в частности. Важнейшим документом, определяющим основные направления цифрового развития страны является принятая 24 декабря 2018 года национальная программа «Цифровая экономика Российской Федерации» [1, 2].

В соответствии с паспортом национальной программы одной из основных целей является «создание устойчивой и безопасной информационно-телекоммуникационной инфраструктуры высокоскоростной передачи, обработки и хранения больших объемов данных, доступной для всех организаций и домохозяйств».

Среди задач, для решения которых создан Национальный проект «Цифровая экономика Российской Федерации», можно выделить следующие, касающиеся темы исследования:

- создание глобальной конкурентоспособной инфраструктуры передачи, обработки и хранения данных преимущественно на основе отечественных разработок;
- обеспечение информационной безопасности на основе отечественных разработок при передаче, обработке и хранении данных, гарантирующей защиту интересов личности, бизнеса и государства;
- создание сквозных цифровых технологий преимущественно на основе отечественных разработок;

- внедрение цифровых технологий и платформенных решений в сферах государственного управления и оказания государственных услуг, в том числе в интересах населения и субъектов малого и среднего предпринимательства, включая индивидуальных предпринимателей;
- преобразование приоритетных отраслей экономики и социальной сферы, включая здравоохранение, образование, промышленность, сельское хозяйство, строительство, городское хозяйство, транспортную и энергетическую инфраструктуру, финансовые услуги, посредством внедрения цифровых технологий и платформенных решений.

График финансирования государственной программы представлен на рис. 1.1.

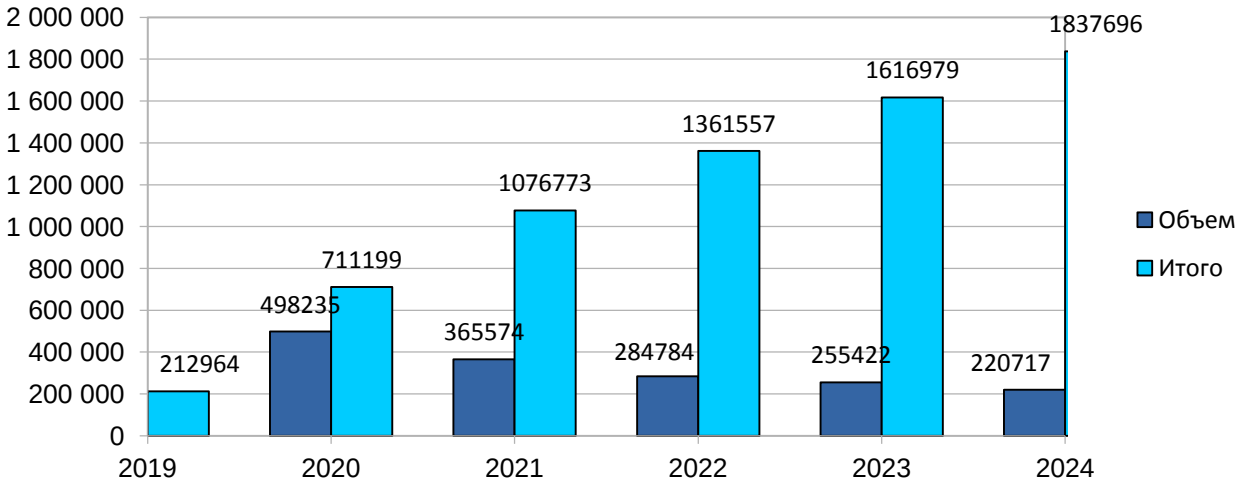


Рисунок 1.1 — Объем финансового обеспечение национальной программы «Цифровая экономика Российской Федерации», млн. руб.

В соответствии с паспортом национальной программы «Цифровая экономика Российской Федерации» среди шести федеральных проектов можно выделить четыре, в большей степени касающихся темы исследования:

- Информационная инфраструктура;
- Информационная безопасность;
- Цифровые технологии;

- Цифровое государственное управление.

Анализируя содержательную часть федеральных проектов можно выделить группы задач и результатов, реализация которых потребует применения современных методов накопления, хранения, анализа и представления результатов, относящихся с области больших и быстрых данных (Big Data, Fast Data):

- оказание универсальных услуг связи на всей территории РФ;
- создание защищенной цифровой среды аудиовизуального взаимодействия;
- функционирование системы распределенных ситуационных центров высших органов государственной власти;
- развитие, поддержание и эксплуатация инфраструктуры российского государственного сегмента сети «Интернет»;
- создание инфраструктуры передачи данных для медицинских и ряда образовательных организаций;
- создание сквозных цифровых технологий и т.д.

С другой стороны, изучая аналитические документы, касающиеся динамики роста объема хранения данных в мире, можно также подтвердить актуальность темы исследования.

Так в соответствии с терминологией аналитической компании IDC [61] в настоящее время мы находимся на третей технологической платформе (рис. 1.2 [28]).

Предложенная IDC парадигма платформ предполагает развитие нескольких платформ, первой из которых является компьютерные системы на основе мэйнфреймов. Первая платформа зародилась с конца 1950-х годов и отдельные ее элементы существуют и по настоящее время.

Вторая платформа основана на архитектуре «клиент-сервер» и началась в середине 1980-х годов, когда ПК подключались к базам данных и приложениям мэйнфреймов. Существует по настоящее время.

Рисунок 1.2 — Третья технологическая платформа IDC

Третья платформа включает технологии социального интернета, мобильные, облачные решения, технологии больших данных и частично интернета вещей (IoT) была выделена в начале 2010 года и развивается на сей день.

В январе 2016 года журнал «The Economist» предложил следующий вариант: «Третья платформа основана на облачных онлайн-вычислениях и взаимодействии со всеми видами устройств, включая беспроводные, такие как смартфоны, оборудование и датчики (все вместе известные как Интернет вещей») [48].

Стоит также отметить, что иногда упоминают четвертую платформу [49]. Данная платформа характеризуется активным внедрением интеллектуальных технологий, IoT, массивно распределенных грид-вычислений и, возможно, квантовых вычислений.

Таким образом, можно сделать вывод, что технологии получения, хранения и анализа больших данных являются неотъемлемой частью современных технологических платформ.

В свою очередь, по прогнозам, приведенным в [19, 32], выручка от продажи программного обеспечения и услуг на мировом рынке больших данных вырастет с 42 млрд долларов в 2018 году до 103 млрд долларов в 2027 году, достигнув совокупного годового темпа роста (CAGR) в 10,48% (рис. 1.3 [32]).

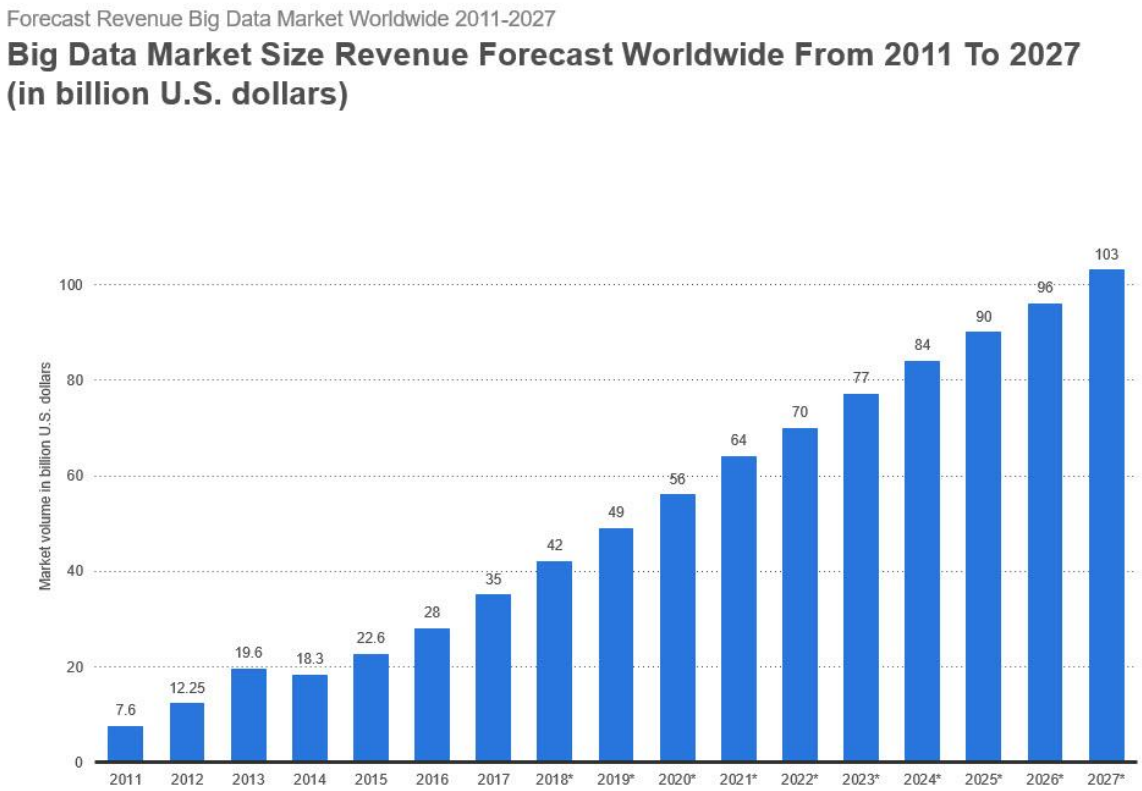


Рисунок 1.3 — Прогноз рынка Больших данных

Согласно исследованию Accenture [5], 79% руководителей предприятий согласны с тем, что компании, которые не используют большие данные, потеряют свои конкурентные позиции и могут оказаться на грани исчезновения. Более того, 83% руководителей реализовали проекты Big Data, чтобы увеличить конкурентное преимущество предприятия на рынке.

Технологии больших данных активно используют преимущества интеллектуальных технологий. Так 59% руководителей говорят, что Big Data в их компании будет улучшена за счет использования ИИ.

Продажи и маркетинг, исследования и разработки (R&D), управление цепочками поставок (SCM), включая дистрибуцию, управление рабочими

местами и операциями, - это то, где передовая аналитика, включая большие данные, вносит наибольший вклад в рост выручки сегодня. Исследование McKinsey [7], представляет собой исчерпывающий обзор того, как аналитические технологии и большие данные позволяют создавать совершенно новые экосистемы, служащие фундаментальной технологией для искусственного интеллекта. (AI). McKinsey считает, что аналитика и большие данные вносят наиболее ценный вклад в отрасли базовых материалов и высоких технологий.

Почти 50% респондентов, принявших участие в опросе McKinsey Analytics, считают, что аналитика и большие данные коренным образом изменили методы ведения бизнеса в области продаж и маркетинга.

По данным NewVantage Venture Partners [15], большие данные приносят наибольшую пользу предприятиям за счет сокращения расходов (49,2%) и создания новых возможностей для инноваций (44,3%). 69,4% предприятий начали использовать большие данные для создания бизнес процессов, управляемыми данными

Прогнозируется, что рынок Hadoop и Big Data вырастет с 17,1 млрд долларов в 2017 году до 99,31 млрд долларов в 2022 году, достигнув 28,5% CAGR (рис. 1.4). Наибольший период прогнозируемого роста приходится на 2021 и 2022 годы, когда, согласно прогнозам, рынок вырастет на 30 млрд долларов в течение одного года [27].

В соответствии с прогнозом [32], что приложения и аналитика для больших данных вырастут с 5,3 млрд долларов в 2018 году до 19,4 млрд долларов в 2026 году, достигнув показателя CAGR 15,49%. Мировой рынок больших данных, включающий в себя профессиональные услуги, вырастет с 16,5 млрд долларов в 2018 году до 21,3 млрд долларов в 2026 году.

Сравнивая мировой спрос на современные технологии аналитики данных и связанное с большими данными оборудование, услуги и программное обеспечение, становится очевидным доминирование последней категории. Прогнозируется, что сегмент программного обеспечения увеличится быстрее

всего относительно других категорий с 14 миллиардов долларов в 2018 году до 46 миллиардов долларов в 2027 году (рис. 1.5), достигнув показателя CAGR в 12,6% [32].

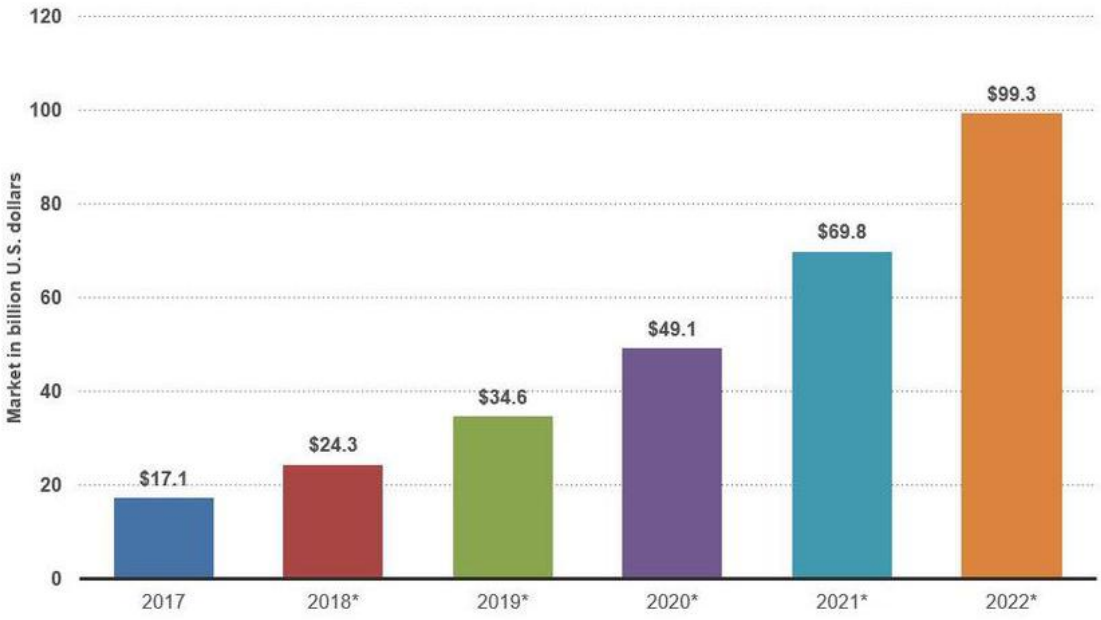


Рисунок 1.4 — Прогноз объема рынков Hadoop и Big Data

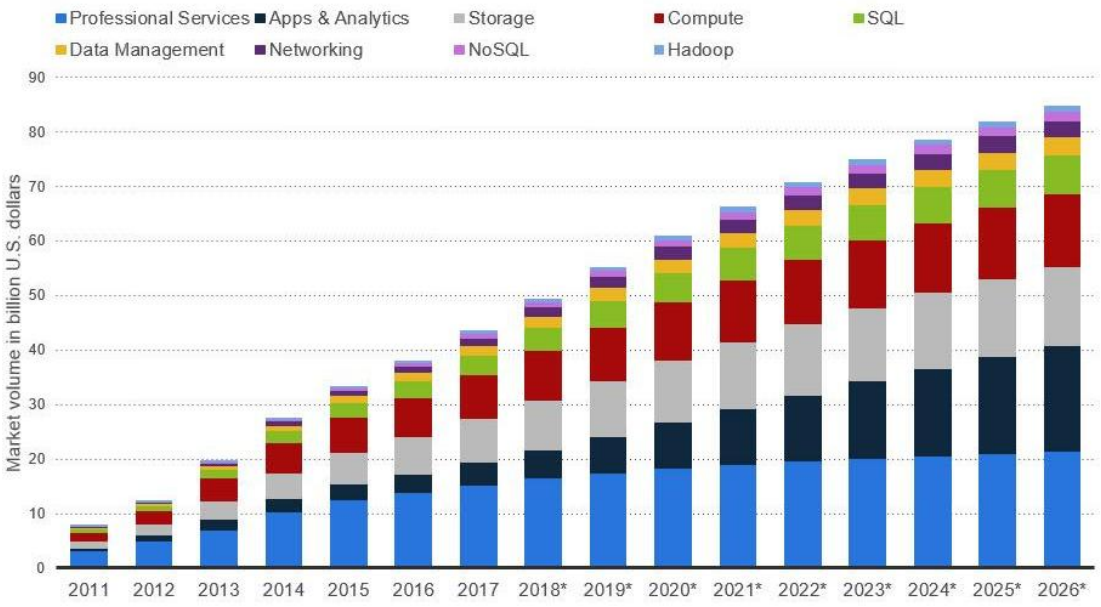


Рисунок 1.5 — Прогноз рынка приложений Big Data по сегментам программного обеспечения

Приведенные прогнозные данные говорят о безусловной актуальности темы научных и инженерных исследований в области больших данных.

1.2 Понятие и особенности больших данных

Если рассматривать термин «большие данные» (Big Data) непосредственно, то именно характеристика большого объема данных не является основополагающей, поскольку именно иные аспекты больших данных определяют суть ново технологии. Первоначально характерными особенностями больших данных были три характеристики, объединённые под известной аббревиатурой «3V» [64, 68]. Однако последние исследования в области больших данных говорят о четырех [40], или даже пяти «V» [13] (рис. 1.6):

- Volume – объем;
- Velocity – скорость;
- Variety – разнообразие;
- Value – ценность;
- Veracity – достоверность.

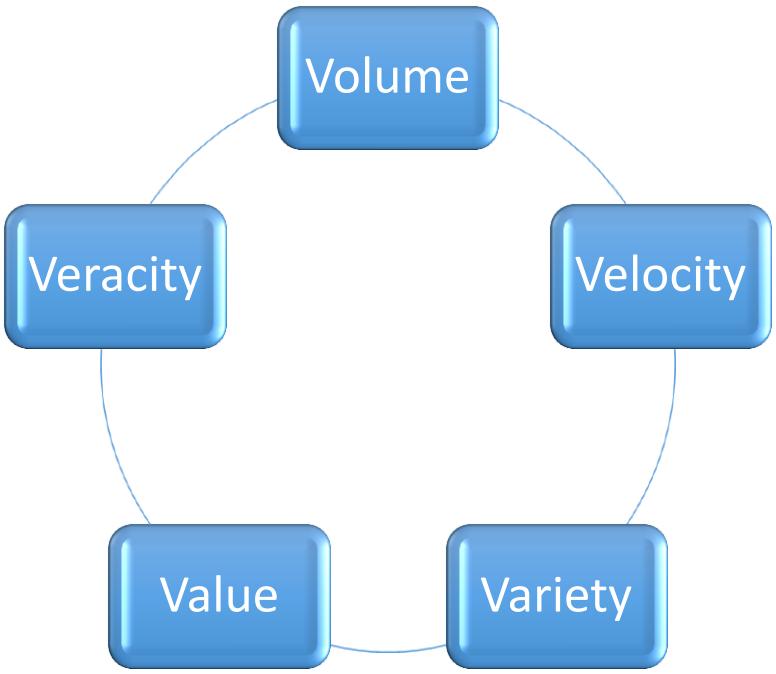


Рисунок 1.6 – Пять «V» больших данных

Термин «Объем» означает, что с генерацией и сбором массивных данных объем данных становится все более большим и, как правило, даже его хранение, не говоря уже о загрузке данных в память одной машины, становится невозможным.

Термин «Скорость» означает, что скорость всей обработки больших данных должна быть быстрой, чтобы своевременно использовать ценность больших данных.

Термин «Разнообразие» указывает на различные типы данных, которые включают частично структурированные и неструктурированные данные, такие как аудио, видео, веб-страницы и текст, а также традиционные структурированные данные.

Четвертым термином стало «Ценность» требующее того, чтобы результаты обработки больших данных приносили ценность заказчику, ощутимым изменениям с учетом затрат на внедрение достаточно дорогостоящих технологий Big Data.

И, наконец, пятым, но не последним по значимости является характеристика «Достоверность». Термин означает насколько точным или правдивым может являться набор данных. В контексте больших данных это требование приобретает намного большее значение по сравнению с обычной бизнес-аналитикой. В частности, когда речь идет о точности больших данных, речь идет не только о качестве самих данных, но о том, насколько надежным является источник данных, их тип и обработка. Устранение ошибок, таких как отклонения или несоответствия, дублирование и нестабильность - аспекты, которые влияют на повышение точности больших данных.

Следует отметить, что на сегодняшний день есть и иные мнения по поводу набора «V» для больших данных. Так, можно встретить информацию о семи «V» [38].

С учетом сказанного под большими в дальнейшем данными будем понимать набор технологий и техник, подразумевающих «работу с информацией огромного объема и разнообразного состава, весьма часто

обновляемой и находящейся в разных источниках в целях увеличения эффективности работы, создания новых продуктов и повышения конкурентоспособности» [53].

Большие данные являются новой технологической ступень. в области Data Science, поэтому большие данные следует отличать от более ранних технологий, например, бизнес аналитики.

Крейг Бати, исполнительный директор по маркетингу и директор по технологиям Fujitsu Australia [53], считает, что «бизнес аналитика является описательным процессом анализа результатов, достигнутых бизнесом в определенный период времени, между тем как скорость обработки больших данных позволяет сделать анализ предсказательным, способным предлагать бизнесу рекомендации на будущее. Технологии больших данных позволяют также анализировать больше типов данных в сравнении с инструментами бизнес-аналитики, что дает возможность фокусироваться не только на структурированных хранилищах».

По мнению Мэтта Слокума из O'Reilly Radar несмотря на то, что большие данные и бизнес-аналитика имеют одинаковую цель – поиск ответов на поставленные вопросы, они отличаются друг от друга по тремя аспектами:

- во–первых, большие данные предназначены для обработки более значительных объемов информации, чем бизнес-аналитика;
- во–вторых, технологии Big Data предназначены для обработки более быстро получаемых и меняющихся сведений, что означает глубокое исследование и интерактивность;
- в-третьих, большие данные предназначены для обработки в том числе и неструктурированных данных, способы использования которых сами являются предметом научных исследований.

Другим отличием, больших данных является также источники данных, которые ранее при решении задач аналитики не использовались. На предыдущих ступенях обработки данных использовались, как правило, внутренние данные, получаемые из систем класса ERP, CRM, корпоративных

баз данных. Большие данные, помимо внутренних источников активно используют также внешние (рисунок 1.7):

- социальные сети
- источники в сети Интернет
- специализированые открытые наборы данных, количество которых в последнее время неуклонно растет.

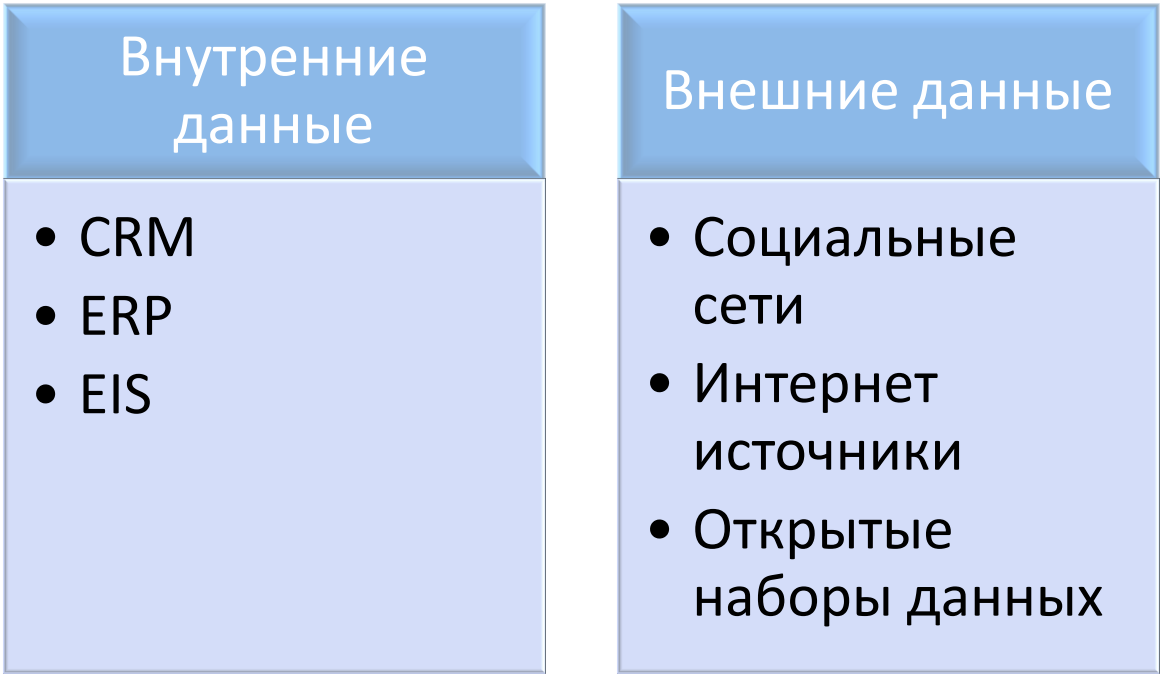


Рисунок 1.7 – Источники Big Data

Кроме рассмотренных отличий сам процесс работы с большими данными также отличается от ранее используемых.

1.3 Процесс Big Data

Процесс Data Science применительно к большим данным может быть представлен в последовательности шести этапов (рис. 1.8), которые отличны, к примеру, от бизнес-аналитики [63]:

1. Определение цели исследования.
2. Этап сбора данных.
3. Этап подготовки данных.
4. Этап исследования данных.

5. Этап моделирования данных.
6. Этап отображения и автоматизации.

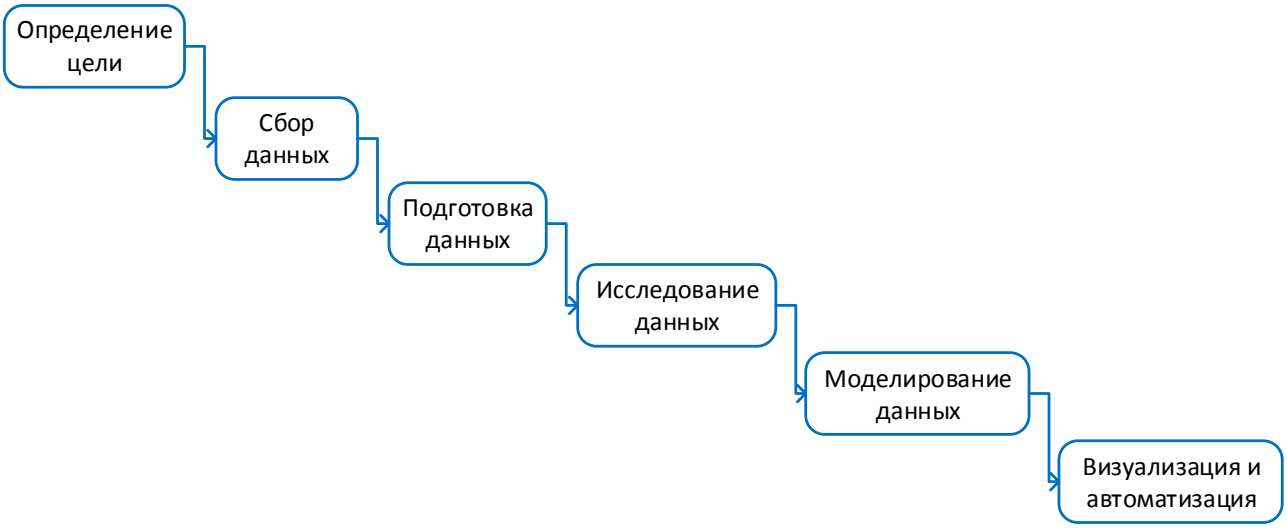


Рисунок 1.8 – Процесс Data Science применительно к Big Data

На первом этапе готовится проектное задание, в котором определяется предмет исследования, предполагаемый результат от внедрения технологии больших данных, какие данные будут использованы, какие ресурсы будут задействованы. В задании также дается описание выходных результатов.

На втором этапе происходит непосредственно сбор данных в соответствии с проектным заданием. Поэтому в задании указываются источники данных. На этом этапе проводится анализ доступности и качества данных. Как уже отмечалось ранее, данные могут существовать во внутренних и/или внешних источниках.

Данные, полученные на втором этапе, могут содержать различные ошибки, поэтому третий этап повышает качество данных, готовя их к дальнейшему использованию. На этом этапе могут быть применены следующие фазы:

- очистка;
- интеграция;
- преобразование.

На фазе очистки алгоритмы устраняют некорректные значения и, возможно, согласуют данные между источниками (если одни и те же данные из разных источником имеют различные значения). На фазе интеграции, как и следовало ожидать, происходит объединение информации, полученной из нескольких источников. И, наконец, фаза преобразования переводит данные в формат, пригодный для использования на последующих этапах.

Четвертый этап – исследование данных – направлен на выяснение характеристик данных, таких как распределение, корреляция, наличие выбросов, кластеров и т.п. На этом этапе активно используются методы описательной статистики и методы визуализации данных. В [63] этот этап называется «исследовательский анализ данных» (Exploratory Data Analysis – EDA).

Моделирование данных, пятый этап, служит для построения модели данных, то есть этот этап непосредственно отвечает на вопросы цели исследования. Этот этап часто бывает итеративным, на котором исследовать пробует те или иные наборы моделей и определяет их характеристики. На этом этапе используются методы статистики, исследования операций, машинного обучения и т.п.

Наконец, на последнем, шестом этапе данные, как правило визуализируются для презентации заказчику. Если разработанная технология необходима заказчику не единовременно, то разрабатывается автоматическое приложение, основанное на полученных моделях данных и примененных алгоритмах.

Реальный процесс анализа больших данных в виду своей сложности и многофакторности часто на выполняется в последовательном виде – исследователю часто приходится возвращаться на предыдущие шаги и вносить коррективы. Поэтому реальный процесс является итеративным (рис. 1.9).

Выводы по первому разделу

Технология хранения и анализа больших данных является перспективной на основе анализа прогнозов.

Большие данные характеризуются различными характеристиками, обозначаемыми «Vs». Анализ литературы показывает, что на сегодняшний день важнейшей характеристикой больших данных является разнородность данных. Именно анализ разнородных данных может дать ощутимый результат при моделировании данных.

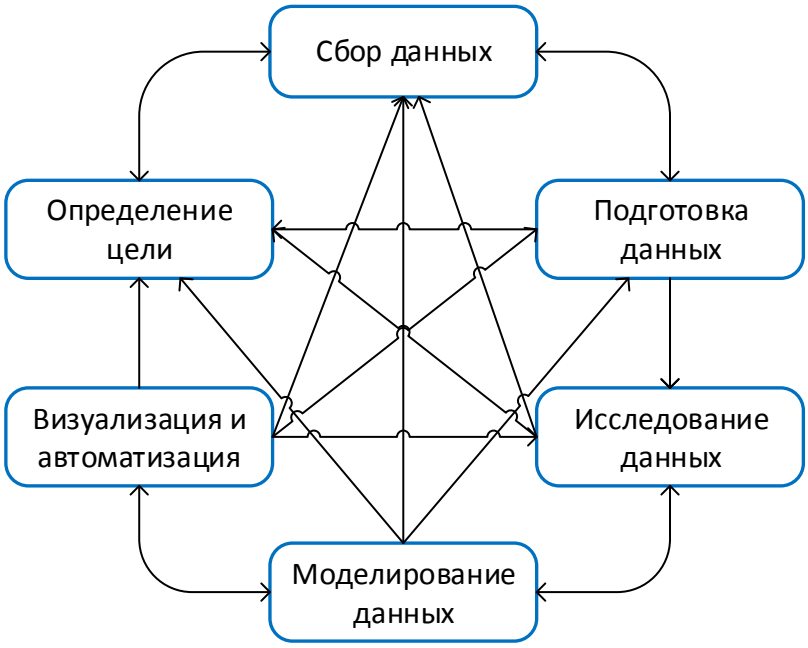


Рисунок 1.9 – Итеративный характер процесса Big Data

Рассмотрены этапы процесса Data Science применительно к хранению и анализу больших данных.

2 АРХИТЕКТУРЫ ПОСТРОЕНИЯ СИСТЕМ ХРАНЕНИЯ И АНАЛИЗА БОЛЬШИХ ДАННЫХ

2.1 Виды архитектурных решений больших данных

Основной целью решений в рассматриваемой области является распределение хранения и обработки данных.

На сегодняшний день существует огромное количество архитектурных решений и инструментов, применяемых для больших данных. Самой распространенной основой для построения систем больших данных является экосистема Hadoop - проект фонда Apache Software Foundation, представляющий собой набор свободно распространяемых утилит, библиотек и фреймворков для разработки распределённых программ и их выполнения на кластерах [29].

По мнению автора все многообразие архитектурных решений для приложений больших данных можно классифицировать по характеристикам: типы больших данных и применимость облачных вычислений. По типу обрабатываемых данных можно выделить два класса приложений:

- системы пакетной обработки больших данных (Batch Processing/Data Processing);
- системы потоковой обработки больших данных (Stream Processing)

Системы пакетной обработки обрабатывают уже накопленные за определенный период времени данные. Например, данные за день могут содержать миллионы записей и быть сохранены в виде файла. Этот файл может подвергаться обработке в конце текущего дня или на следующий день. При этом, возможно, что для обработки дневных данных потребуется достаточно много времени. Hadoop MapReduce, по оценке многих специалистов, лучшая платформа для обработки данных в пакетном режиме. На рисунке 2.1 показана схема обработки данных с помощью MapReduce. [50].

Системы пакетной обработки применимы в случаях, когда нет необходимости обработки в реальном времени, а основной целью анализа является получение детальной информации.

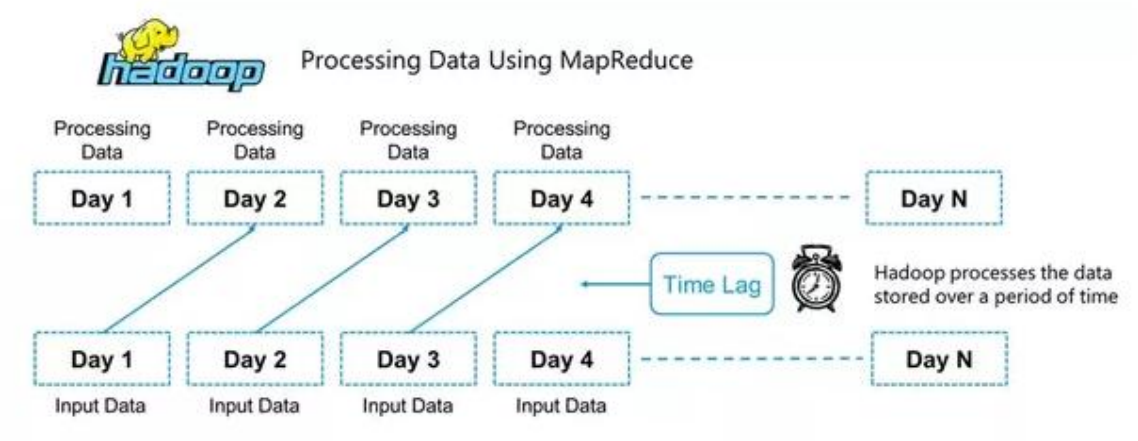


Рисунок 2.1 – Схема пакетной обработки данных в Hadoop MapReduce

Системы потоковой обработки применяются в ситуации, когда необходимо получать аналитические результаты в реальном времени. Потоковая обработка позволяет обрабатывать данные в реальном времени по мере их поступления и быстро обнаруживать определенные ситуации в течение небольшого периода времени с момента получения данных. Основной причиной высокой скорости потоковой обработки является то, что система анализирует данные до их записи в файловую систему.

Потоковая обработка полезна для таких задач, как мониторинг активности сайтов, обнаружение мошенничества при проведении транзакций в бизнесе или онлайн обучении. Например, при обработке данных транзакций в потоковом режиме, возможно обнаружение аномалий, сигнализирующих о мошенничестве в режиме реального времени для их блокировки. На рисунке 2.2 приведена схема обработки данных в реальном времени с использованием Apache Spark.

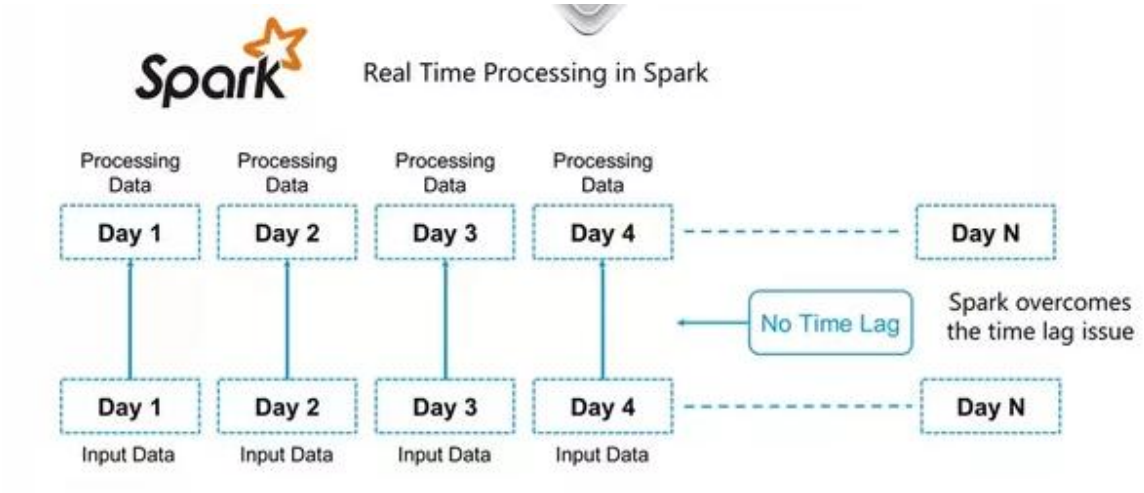


Рисунок 2.2 – Обработка больших данных в реальном времени с использованием Apache Spark

Определенный интерес для построения систем потоковой обработки данных могут представлять системы, построенные на основе лямбда архитектуры [35]. Однако, в данной работе этот вопрос не рассматривается.

С точки зрения применимости облачных вычислений системы больших данных могут классифицироваться на следующие виды (см. рис. 2.3) [69]:

- инфраструктура как сервис – Infrastructure as a Service (IaaS);
- платформа как сервис – Platform as a Service (PaaS);
- даныве как сервис – Data as a Service (DaaS);
- бизнес решения для больших данных – Big Data Business Functions as a Service (BFaaS).

Системы класса IaaS включают хранилища, сервера и сетевые решения в качестве базовых, относительно недорогих решений для создания стека больших данных. Распределенные файловые системы являются частью этого уровня.

Системы класса PaaS предоставляют хранилища данных NoSQL и распределенные кэши, которые могут быть логически запрошены с использованием языков запросов дл создания платформенного уровеня

больших данных. Этот уровень обеспечивает логическую модель для необработанных, неструктурированных данных, хранящихся в файлах.

Рисунок 2.3 – Облачные уровни архитектур для построения систем больших данных

DaaS системы включают весь набор инструментов, доступных для интеграции со слоем PaaS с использованием поисковых систем, адаптеров интеграции, пакетных программ и т.д. API-интерфейсы, доступные на этом уровне, могут использоваться всеми оконечными системами в режиме вычислительной эластичности (elastic-computing mode) [23].

Системы BFaaS могут создаваться как сквозные решения над уровнем DaaS. BFaaS решения создаются для определенных отраслей таких как здравоохранение, энергетика, банковское дело, розничная и электронная торговля.

Как отмечалось выше, для построения систем больших данных может быть использован большой набор фреймворков, продуктов, библиотек, которые принято называть экосистемой больших данных

2.2 Экосистема больших данных

Экосистема больших данных включает в себя следующие группы инструментов [63]:

- распределенные файловые системы (Distributed File Systems – DFS);
- инструменты развертывания;
- базы данных NoSQL и NewSQL;
- иструменты интеграции данных;
- инстументы машинного обучения;
- инструменты программирования служб;
- иструменты планирования;
- инструменты сравнительного анализа;
- инструменты безопасности.

Для хранение больших объемов данных используются распределенные файловые системы, обладающие следующими отличиями:

- способность хранить файлы, размер которых превышает размер отдельного диска сервера хранения;
- хранимые файлы автоматически реплицируются на серверах хранения, что позволяет осуществлять параллельную обработку и создавать избыточность для обеспречения надежной работы кластера;
- система может легко масштабироваться, используя принцип горизонтального масштабирования.

Наиболее распространенной распределенной файловой системой является Hadoop File System. Примерами DFS также являются: Red Hat ClasterFS, QuantCast File System, Ceph File System.

Для хранения больших данных требуются системы баз данных и средства доступа к данным. В виду известных ограничений использование классических реляционных бfз данных, таких как Oracle SQL, DB2, невозможно. В системах анализа и хранения больших данных применяются системы, получившие название NoSQL и их дальнейшее развитие NewSQL. Более подробно указанные системы баз данных будут рассмотрены ниже. Здесь отметим, что

помимо самих систем баз данных, необходимы системы сбора, преобразования пакетных и потоковых данных.

Инструменты интеграции данных, как это следует из названия, служит для слияния данных посредством перемещения данных из одного источника в другой. Как уже отмечалось выше, существуют два основных класса решений больших данных для обработки пакетных и потоковых данных. Примерами решений для пакетной обработки являются Apache Sqoop. Примерами решения для использования в обработке потоковых данных являются Неординарность задачи привела к появлению таких технологий, как Apache Flume, Apache Storm и Apache Kafka.

После перемещения данных в распределенную файловую систему необходимо перейди к извлечению информации из данных. Для этих целей в процессе анализа больших данных используются методы прикладной математики, математической статистики, машинного обучения. Однако важным отличием алгоритмов, используемых при анализе больших данных является то, что алгоритмы должны быть распределёнными. На сегодняшний день существует много библиотек и фреймворков, реализующих указанные алгоритмы. Примерами языков программирования, активной используемых в данной задаче являются Python, Java, R. Например, для Python имеются такие библиотеки, как:

- NLTK – Natural Language Toolkit – библиотека обработки данных на естественном языке;
- Scikit-learn – одна из известнейших библиотек машинного обучения;
- TenzorFlow – библиотека глубокого машинного обучения от Google.

Примером системы машинного обучения в реальном времени может служить Apache Spark.

Инфраструктуры распределенного программирования упрощают реализацию распределённых алгоритмов поскольку реализуют «низкоуровневые распределенные» задачи, скрывая их от программиста. К таким задачам можно отнести: перераспределение задач в случае их сбоя на

узле вычислений, коммуникация между подпроцессами. Примерами инфраструктур распределенного программирования являются Apache Thrift, Zookeeper.

Инструменты планирования служат для автоматизации повторяющихся заданий. Например, запуск задач MapReduce при появлении нового набора данных. Представителями данной группы являются Hadoop YARN.

Инструменты сравнительного анализа служат для оптимизации инфраструктур больших данных за счет использования стандартизированных профилей. Каждый профиль строится на основе определенного набора инструментов для хранения и обработки больших данных.

2.3 Обобщенная архитектура систем хранения и анализа больших данных

Любая информационная система по хранению и анализу больших данных должна строиться на определенной аппаратно-программной архитектуре. Существует обобщенная архитектурная схема приложений больших данных, определяющая технологический стек больших данных (big data tech stack). В общем случае архитектура больших данных может быть представлена в виде, изображенном на рисунке 2.4 [69].

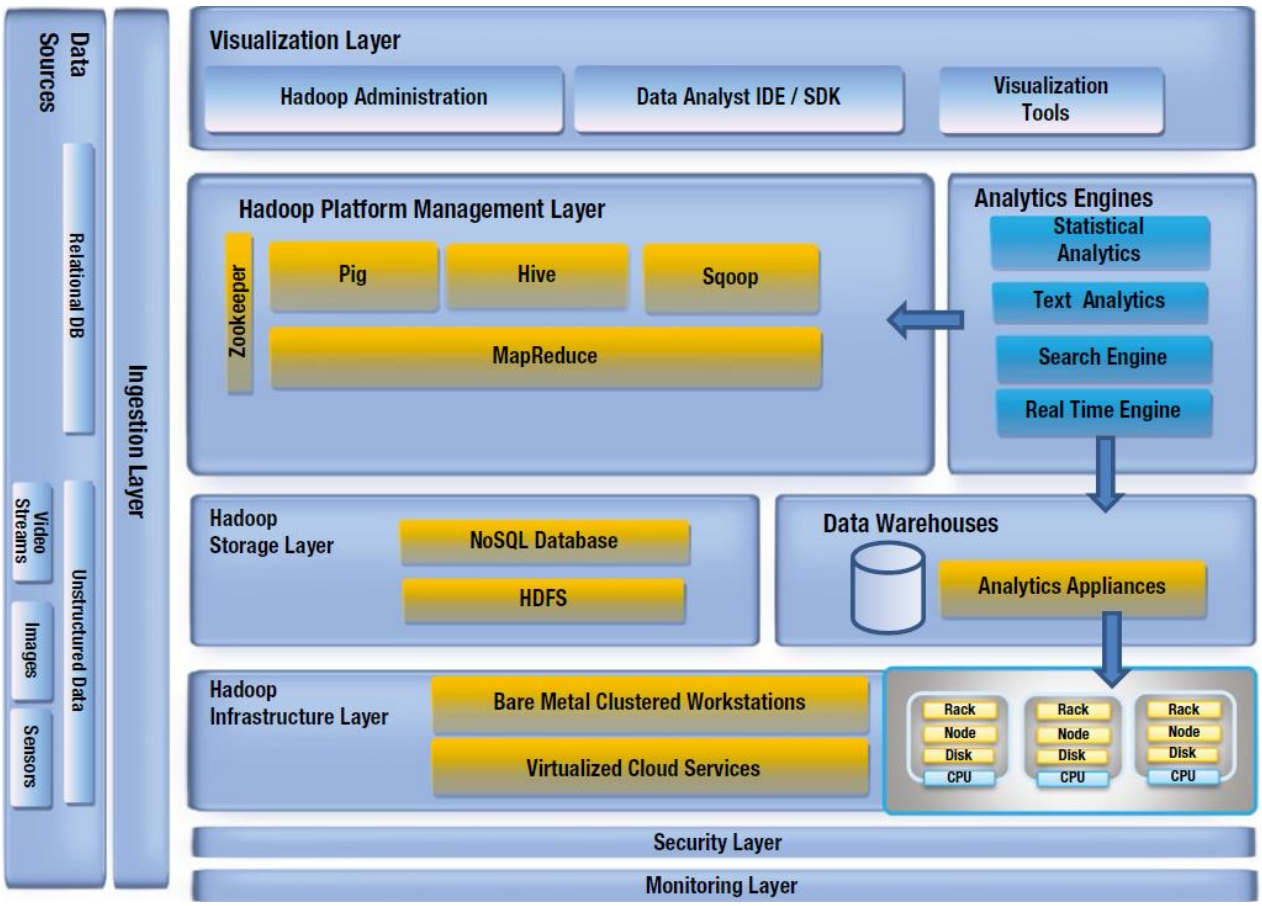


Рисунок 2.4 – Архитектура системы хранения и анализа больших данных

Опишем уровни представленной архитектуры более подробно.

Уровень источников данных (Data Sources). Для предприятий обычно доступны несколько внутренних и внешних источников данных. При этом существует требования, чтобы перед записью данные должны быть очищены, верифицированы, масштабированы.

Данные могут поставляться в различных форматах: результаты запросов к реляционным базам данных и хранилищам данных, сообщения электронной почты, XML, JSON, HTML, мгновенные сообщения, видео и аудио данные, форматы документов офисных приложений (Word, Excel, pdf), а также потоковые данные.

Как уже отмечалось выше данные источники характеризуются большой скоростью выдачи данных, большим разнообразием форматов, большим объемом данных.

Слой загрузки данных (Ingestion Layer). Слой загрузки (Рисунок 2.5) – новый слой обработки данных предприятия. Этот слой несет ответственность за

отделение шума от соответствующей информации. Алгоритмы этого слоя должны иметь возможность проверять, очищать, преобразовывать, сокращать и агрегировать данные в технический стек больших данных для дальнейшей обработки. Это новое промежуточное программное обеспечение, которое должно быть масштабируемым, отказоустойчивым, гибким и регулирующим в архитектуре больших данных. В соответствии с процессом Data Science ошибки в данном слое могут свести на нет всю дальнейшую работу.

Уровень загрузки загружает окончательную релевантную информацию без шума в распределенный уровень хранения Hadoop. Алгоритмы этого уровня должны проверять, очищать, преобразовывать, сокращать и интегрировать данные в технологический стек больших данных для дальнейшей обработки.

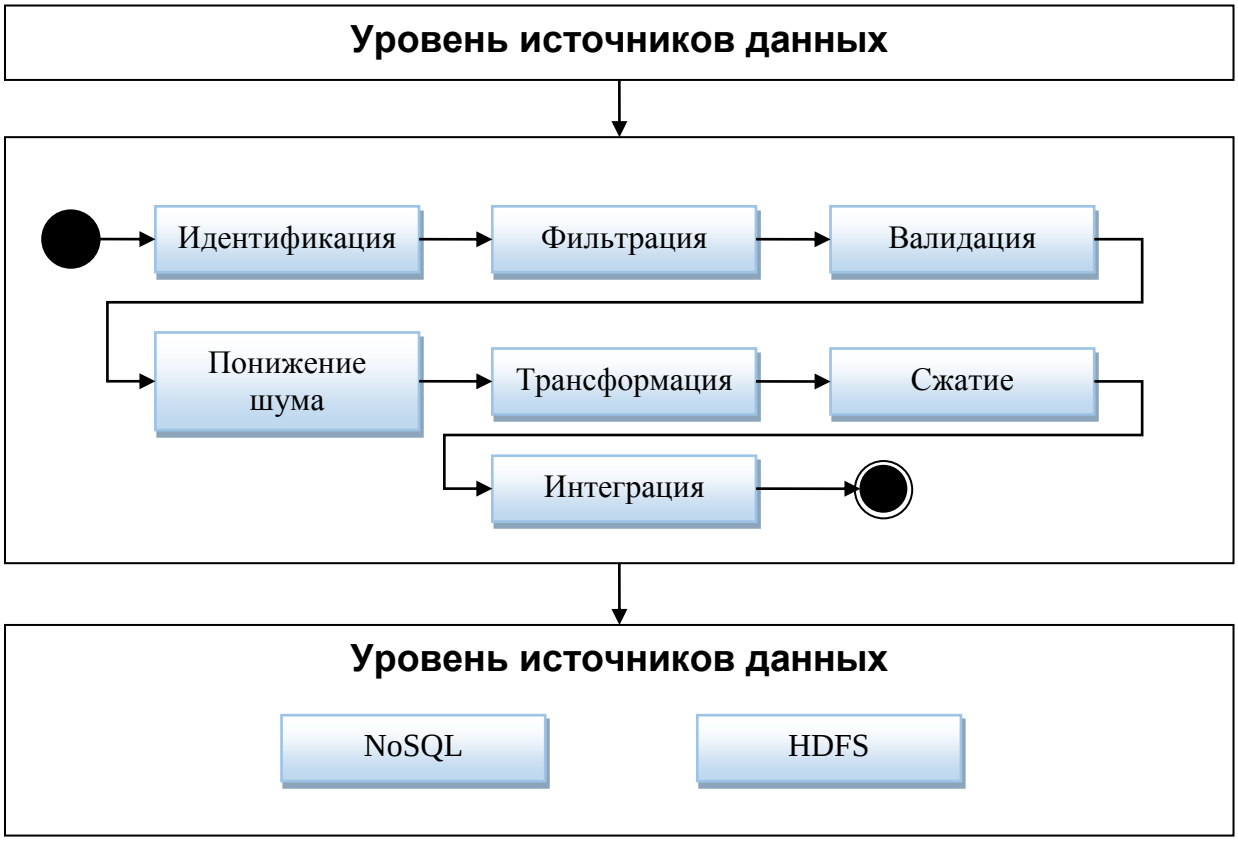


Рисунок 2.5 – Роль уровня загрузки данных в архитектуре приложений хранения и анализа больших данных

Архитектурные шаблоны слоя загрузки описывают решения часто встречающихся проблем источников данных с точки зрения влияния на уровень загрузки. Эти решения могут быть выбраны на основе требований к производительности, масштабируемости и доступности. Мы рассмотрим эти шаблоны (которые показаны на рисунке 3-1) в последующих разделах. В этой главе мы рассмотрим следующие распространенные шаблоны загрузки пакетных и потоковых данных [69]:

- шаблон извлечения данных из множества источников (Multisource Extractor Pattern) является подходом для эффективного использования нескольких типов источников данных;
- шаблон преобразователя протокола (Protocol Converter Pattern). В этом шаблоне используется посредник протокола для обеспечения абстракции для входящих данных с разных уровней протокола;
- шаблон многоцелевого назначения (Multidestination Pattern): этот шаблон используется в сценарии, когда слой загрузки должен переносить данные в несколько компонентов хранения, таких как распределенная файловая система Hadoop, витрины данных или аналитические механизмы реального времени;
- шаблон преобразования «точно в срок» (Just-in-Time Transformation Pattern). Большие объемы неструктурированных данных могут быть загружены в пакетном режиме с использованием традиционных инструментов и методов ETL (извлечения, передачи и загрузки). Однако данные преобразуются только тогда, когда это необходимо для экономии времени вычислений;
- шаблоны потоковой передачи в реальном времени (Real-Time Streaming patterns). Некоторые бизнес-проблемы требуют мгновенного анализа данных, поступающих на предприятие. В этих условиях необходимы загрузка и анализ данных в режиме реального времени.

Уровень распределенного хранения (Distributed (Hadoop) Storage Layer) обеспечивает надежное, масштабируемое окружение для вычисления

параллельных алгоритмов обработки больших данных. Распределенная файловая система Hadoop является основным элементом данного уровня. Непосредственное управление доступом в распределённым данным осуществляют NoSQL базы данных, рассматриваемые ниже.

Инфраструктурный уровень (Hadoop Infrastructure Layer) – уровень, поддерживающий уровень хранения, то есть физическую инфраструктуру. Инфраструктурный уровень является основополагающим для работы и масштабируемости архитектуры больших данных. Для поддержки неожиданного или непредсказуемого объема, скорости или разнообразия данных физическая инфраструктура для больших данных должна отличаться от инфраструктуры для традиционных реляционных данных.

Уровень физической инфраструктуры Hadoop (Hadoop physical infrastructure layer – HPIL) основан на модели распределенных вычислений. Это означает, что данные физически хранятся на многих местах и связываются вместе через сети и распределенную файловую систему. Это архитектура «без разделения ресурсов», в которой данные и функции, необходимые для управления ими, находятся вместе на одном узле. В отличии от традиционной модели клиент-сервер, данные больше не передаются на монолитный сервер, где для их обработки применяются функции SQL. В инфраструктуру этого уровня встроена избыточность.

Уровень безопасности (Security Layer). Поскольку анализ больших данных становится одной из главных задач для организаций, безопасность этих данных становится также главной задачей. Хранимые данные и результаты их обработки должны быть защищены как для соблюдения соответствующих требований, так и для защиты частной жизни человека. Поэтому с самого начала должны планироваться средства авторизации и аутентификации.

Чтобы реализовать базовую основу безопасности, необходимо спроектировать определенный стек технологий, который минимально выполнял бы следующие действия:

- аутентификация узлов, используя протоколы, такие как Kerberos;

- шифрование на уровне файлов;
- подписка на службу управления ключами для доверенных ключей и сертификатов;
- использование таких инструментов, как Chef или Puppet, для проверки во время развертывания наборов данных или при применении исправлений на виртуальных узлах;
- регистрация связи между узлами и использование распределенных механизмов регистрации для отслеживания аномалий по уровням;
- гарантия того, что связь между узлами является безопасной. Например, использование SSL, TLS и т.д.

Мониторинговый уровень (Monitoring Layer). Системы мониторинга применяются для отслеживания состояния распределенных кластеров и собирают информацию об используемых операционных системах, оборудовании и т.п. Для выполнения данной задачи машины должны взаимодействовать с инструментом мониторинга через протоколы высокого уровня, такие как XML, вместо двоичных форматов, которые зависят от машины. Мониторинговые системы также должны предоставлять инструменты для хранения и визуализации данных.

Для мониторинга стеков больших данных широко используются такие инструменты с открытым исходным кодом, как Ganglia и Nagios.

Приложения аналитического уровня (Analytics Engine) служат для выполнения поисковых запросов над распределенными данными, выполняют аналитическую обработку текста, статистическую аналитику и выполняют интеллектуальные алгоритмы над данными. Более подробно алгоритмы анализа больших данных будут рассмотрены в следующей главе.

Средства визуализации (Analytics Engine). Как правило, «сырые» выходные аналитические данные не могут использоваться для решения бизнес-задач. Необходим перевод аналитических данных в табличную или графическую форму, а также возможность взгляда на данные под различными

углами. Поэтому средства визуализации являются неотъемлемой частью систем хранения и анализа больших данных.

Средства визуализации работают поверх консолидированных и агрегированных выходных данных. В том случае, когда требуется изучение результатов аналитики в режиме реального времени, могут использоваться механизмы реального времени, работающие на механизмах комплексной обработки событий (Complex Event Processing – CEP) и управляемых событиями архитектурах (Event-driven Architectures – EDA). На рис. 2.6 показано взаимодействие между различными уровнями стека больших данных и традиционными средствами BI.

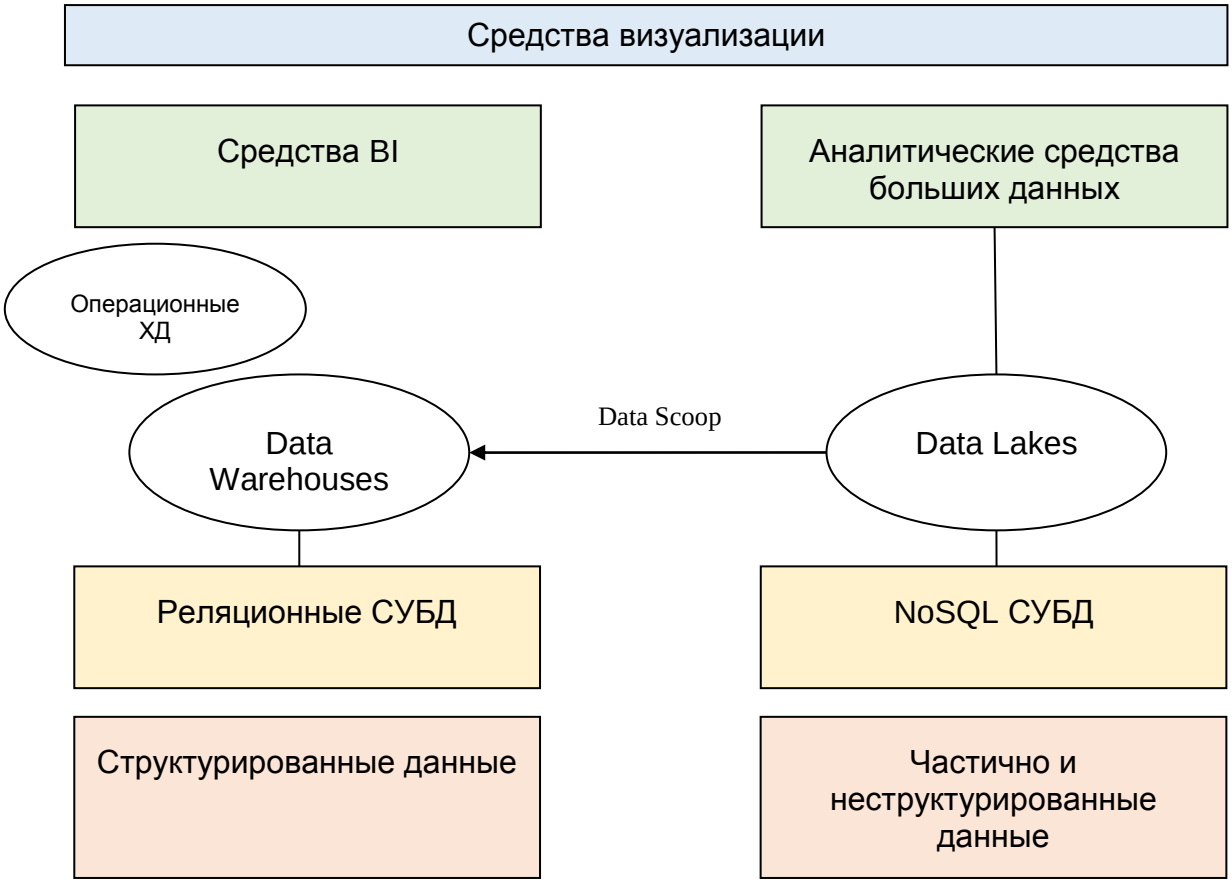


Рисунок 2.6 – Концептуальная архитектура слоя визуализации

2.4 Виды и области применения NoSQL баз данных

Основу современных транзакционных систем составляют реляционные базы данных. Однако, несмотря на их распространение, реляционные базы имеют всем известное ограничение, связанное с проблемами производительности при выполнении агрегирующих запросов к данным. Для решения этой проблемы в BI системах используются решения, основанные на использовании хранилищ данных (ХД). Однако ХД в силу своих ограничений не могут быть напрямую использованы при построении систем хранения и анализа больших данных.

Второй проблемой традиционных систем управления данных является то, что в основном они призваны для развертывания на одной машине. Однако, уже достаточно давно были разработаны решения для распараллеливания работы на распределенных кластерах. Такие решения получили обобщенное название NewSQL [69].

В системах хранения и анализа больших данных самое широкое применение получили системы управления базами данных под обобщенным названием NoSQL (Not Only SQL – не только SQL). Особенностью NoSQL БД является их способность работать не только с большими данными (Volume), но и с остальными «V».

На сегодняшний день общепринятой является классификация NoSQL баз данных по четырем видам:

- столбцовые;
- «ключ-значение»;
- хранилища документов;
- графовые.

Эти разновидности NoSQL баз данных позволяют по разному организовывать данные в зависимости от задачи. Кроме того, указанные разновидности применимы к разным ситуациям с точки зрения сложности организации и размера данных (рис. 2.7).

Кроме различных форматов и подходов к организации хранимых данных реляционные и NoSQL базы данных имеют разные базовые принципы, определяющие состояние системы хранения данных.

Так для реляционных и графовых баз данных должен выполняться принцип ACID, описывающий свойства классических транзакций в реляционных базах. Принцип ACID подразумевает четыре составляющих: Atomicity, Consistency, Isolation, Durability – ACID:

- Atomicity – атомарность. Если происходит CRUD операция над блоком данных, состоящая из нескольких связанных действий, то операция считается успешной, если все элементы последовательности успешны. Иначе операция отменяется. Этот принцип известен также под названием «все или ничего»;

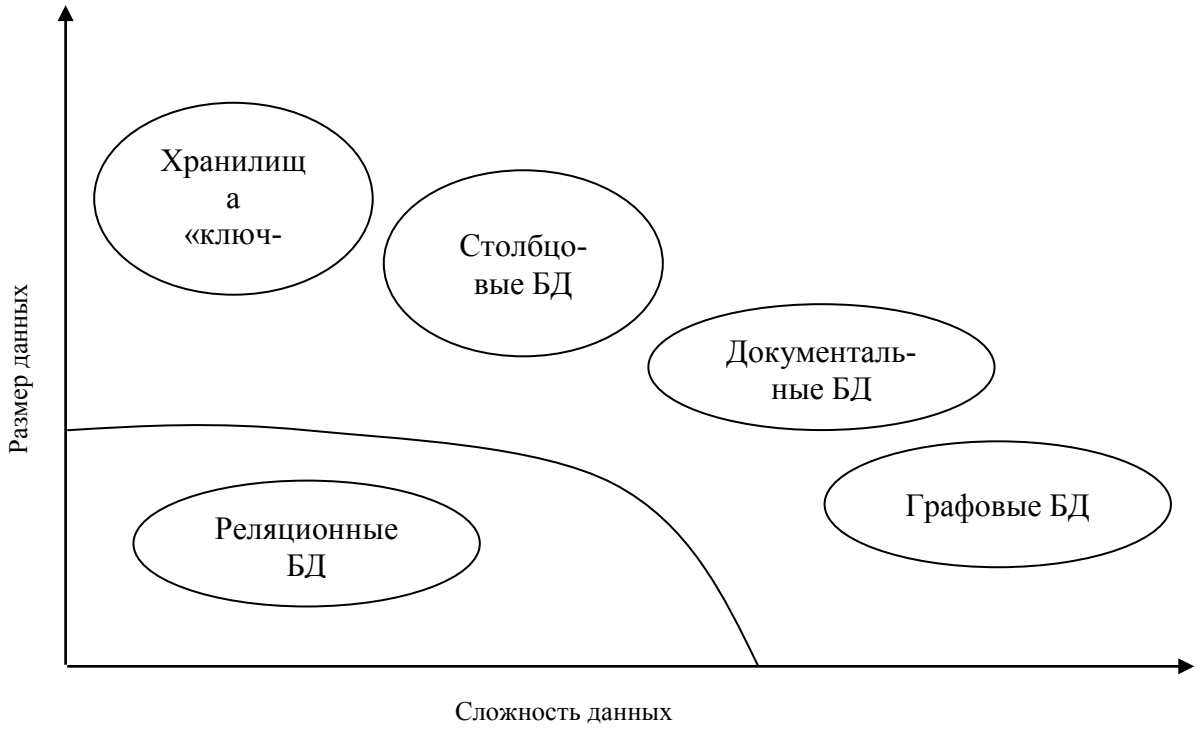


Рисунок 2.7 – Области применения NoSQL баз данных
в системе «Сложность-размер» данных

- Consistency – согласованность. Если состояние системы является в непротиворечивом состоянии до выполнения транзакции, то после

выполнения транзакции система также должна находиться в непротиворечивом состоянии;

- Isolation – изолированность. Если над данными происходит несколько параллельных транзакций, то их результат должен быть такой же, как если бы эти транзакции выполнялись последовательно;
- Durability – постоянство. Если данные внесены в базу данных, то они должны там находиться постоянно. Никакие сбои, пожалуй, за исключением физического разрушения носителя, не могут привести к потере данных.

Соблюдение принципов ACID приводит в высоконагруженных реляционных системах к большим проблемам, связанным в основном с обеспечением принципа изолированности параллельных транзакций. Действительно, при выполнении параллельных транзакций могут происходить следующие ошибки [3]:

- потерянное обновление (Lost Update);
- грязное чтение (Dirty Read);
- неповторяемое (размытое) чтение (Non-repeatable, Fuzzy Read);
- фантом (фиктивные элементы) (Phantom).

Для обеспечения приемлемой производительности системы приходится идти на некоторые компромиссы, определяя механизмы блокировок и уровни изоляции:

- чтение незафиксированных данных (READ UNCOMMITTED). Самый низкий уровень, при которой транзакция может читать незафиксированные данные;
- чтение зафиксированных данных (READ COMMITED). При этом уровне транзакция не может читать незафиксированные данные;
- повторяемое чтение (REPEATABLE READ). Транзакция не может изменить данные, которые считываются другими транзакциями;

- сериализуемое чтение (SERIALIZABLE). Самый высокий уровень изоляции. Транзакция имеет эксклюзивное право на чтение. Другие транзакции не могут ни читать, ни записывать эти же данные.

Однако, даже несмотря на наличие различных уровней изоляции, применение реляционных систем управления базами данных в распределенных системах не является приемлемым. Для распределенных систем хранения, когда данные распределены по разным серверам, соблюдение принципов ACID невозможно ввиду действия теоремы CAP [63]. Согласно теореме CAP распределенная база данных может обладать любыми двумя из трех свойств, но никогда всеми тремя. К указанным свойствам теорема CAP относит:

- согласованность (Consitency). Запрашивающий узел получит одни и те же данные, несмотря на то от какого узла-источника эти данные пришли;
- доступность (Availability). Если удалось связаться с узлом и это узел находится в работоспособном состоянии, то узел обязательно ответит, даже если пропадет связь с другими узлами базы;
- устойчивость к разделению (Partition tolerance). Данные в базе не перестанут быть доступными в результате сетевой сегментации или сетевого сбоя.

Таким образом, если система хранения данных секционирована и обладает устойчивостью к разделению, то она либо доступна, либо согласована. Исходя из сказанного, NoSQL базы данных обеспечивают выполнение принципов BASE, а не ACID:

- базовая доступность (Basically Available). Гарантируется доступность в смысле CAP. На одном узле хранятся несколько сегментов данных с других узлов, что позволяет решать проблему доступа к информации при сетевом сбое;
- неустойчивое состояние (Soft state). Состояние системы меняется со временем и может переходить в несогласованное состояние, но в итоге действует принцип «согласованности в конечном счете»;

- согласованность в конечном счете (Eventual consistency). Данный принцип говорит о том, что система рано или поздно придет в согласованное состояние. Например, если на узле А были изменены данные, реплицированные на узле В, то через какое-то время данные будут обновлены и на узле В. При этом никаких дополнительных действий от самих узлов (пользователя) не требуется.

2.4.1 Столбцовые БД

Традиционные реляционные базы данных являются строково-ориентированными, когда каждой строке таблицы соответствует уникальный идентификатор – первичный ключ (Primary Key – PK). В этих базах для поиска информации приходится сканировать данные в таблице по определённым столбцам, выполняя соответствующие запросы. Несмотря на то, что для ускорения выполнения запросов применяется индексирование, в разреженных таблицах поиск может оказаться малоэффективным в принципе. Кроме того, определенные проблемы может составить добавление одного или нескольких атрибутов (столбцов).

Поэтому в столбцовых базах данных, как правило, столбцы хранятся по раздельности, что приводит к сокращению времени сканирования по отдельным столбцам.

В отличии от реляционных баз данных отлично зарекомендовавших себя при построении OLTP систем, когда часто выполняется CRUD операция над всей записью (строкой), столбцовые базы данных применимы при построении аналитических систем. Часто базы данных аналитических систем обновляются по расписанию (например, по ночам), перенося данные из реляционных хранилищ OLTP систем. В дальнейшем выполнение аналитических запросов (поиск и агрегация) происходит за очень короткий промежуток времени с использованием алгоритмов MapReduce.

Примерами столбцовых баз данных являются Apach HBase [10], HyperTable [33], Apache Cassandra [8], Google BigTable [16], Yandex ClickHouse [17].

2.4.2 NoSQL базы данных «ключ-значение»

Базы данных «ключ-значение» представляют собой набор записей, которые имеют уникальное поле «ключ» и соответствующее поле «значение». Таким образом, данные тип хранилищ реализует известную структуру данных map. Благодаря своей простоте хранилища «ключ-значение» обладают наилучшей масштабируемостью и, соответственно, позволяют хранить огромные объёмы информации.

С учетом известного приема организации данных в коллекциях map, когда в качестве «значения» может быть иная структура данных (строка, список, множество и даже отображение (map)), структура хранимых данных может быть весьма разнообразной.

Примерами хранилищ «ключ-значение» являются Reddis [47], MemCache [39], Amazon DynamoDB [6], Microsoft Azure Cosmos DB [12].

2.4.3 Документальные NoSQL базы данных

Документальные NoSQL базы данных прежде всего предполагают определенную структуру документов. Структура документов задается схемой документов. Наиболее часто встречающиеся структуры основаны на языках разметки таких как XML и JSON. При использовании в качестве единицы хранения документов можно избежать большой структуры связанных таблиц реляционных хранилищ, что позволяет снизить до минимума когнитивную нагрузку при разработке алгоритмов работы с документальными NoSQL базами данных.

Примерами документальных NoSQL баз данных являются односекционная база MongoDB [41], Elasticsearch [43], Apache CouchDB [9].

2.4.4 Графовые NoSQL базы данных

Графовые базы данных ориентированы на построение различных отношений между сущностями и с этой точки зрения являются самой сложной по организации данных. Если данные обладают высокой связанностью, например, социальные сети, то графовые базы данных являются наилучшим решением. В рассматриваемых базах непосредственно данные состоят из двух видов компонент:

- узел – компонента, представляющая непосредственно узел и связанную с ним информацию;
- ребро – компонента, определяющая отношение между двумя ребрами. Ребро может иметь направление и дополнительные атрибуты

В отличии от других видов NoSQL баз данных поддерживают соблюдение принципов ACID.

Области применения графовых баз данных огромны. Это могут быть рекомендательные системы [63], приложения в области онлайн и дополнительного образования и т.д.

Примерами графовых баз данных является Neo4j [42], Microsoft Azure Cosmos DB [12], OrientDB [46], ArangoDB [11], Virtuoso [45].

Сегмент баз данных в целом и NoSQL баз данных в частности является достаточно динамическим рынком программных продуктов. Поэтому появляются новые предложения, которые занимают лидирующие места в определенных сегментах. На интернет ресурсе DB-Engines [22] можно познакомится с актуальным рейтингом баз данных.

Таким образом, проведя обзор NoSQL баз данных можно сделать следующие выводы:

- максимальным быстродействием при выполнении поисковых и агрегирующих запросов обладают столбцовые NoSQL базы данных;
- наилучшие показатели сегментации и объема хранения информации присущи хранилищам «ключ-значение»;

- документальные NoSQL базы данных позволяют создавать четкие структуры разнородных данных, что обеспечивает минимум когнитивной нагрузки при разработке алгоритмов обработки структур данных;
- графовые базы данных обладают способностью хранения данных с высоким уровнем связанности.

2.5 Дистрибутивы для развертывания систем хранения и анализа больших данных

Как показывает выполненный обзор архитектур и инструментов систем хранения и анализа больших данных, существует огромный набор возможных вариантов построения. Количество возможных решений может быть очень большим, если взять во внимание разрабатываемые перечни инструментов (ландшафтов - Landscape) для создания систем хранения и анализа больших данных. Примером может служить интернет ресурс, посвящённый изучению решений в области больших данных [14]. Например, на рисунке 2.8 приведен ландшафт приложений уровня предприятия больших данных. Основная цель приложений Big Data - помочь компаниям принимать более информативные бизнес-решения, анализируя большие объемы данных. Эти данные могут включать в себя журналы веб-серверов, данные о потоке интернет-кликов, контенты в социальных сетях и отчеты об активности, тексты из электронных писем клиентов, сведения о звонках с мобильных телефонов и технические данные, снимаемые датчиками по технологии IoT. Как отмечалось выше, эти могут быть структурированными, частично структурированными и неструктурированными.

Рисунок 2.8 – Ландшафт приложений Big Data уровня предприятия

Как мы видим, существует большое количество инструментов. Поэтому остро возникает вопрос об их интеграции и развертывании на вычислительных кластерах. Совершено очевидно, что только развертывание такой системы может стать сложнейшей задачей. Для быстрого создания систем хранения и анализа больших данных могут использоваться специальные дистрибутивы — программные комплексы для развертывания кластера, его мониторинга и управления. На сегодняшний день существует ряд таких дистрибутивов:

- платформы развертывания от фирмы Hortonworks [26]:

    - Hortonworks Data Platform (HDP) – виртуальная машина с полным набором инструментов для пакетной обработки данных,
    - Hortonworks DataFlow (HDF) – виртуальная машина с полным набором инструментов для потоковой обработки данных;
- Cloudera Distribution including Apache Hadoop (CDH) [18] – дистрибутив программного открытого обеспечения, содержащий Apache Hadoop и ключевые компоненты, такие как Apache Flume, Apache Hive, Apache Kafka и т.д.;
- MapR Converged Data Platform [36] - единая платформа, выполненная на одной кодовой базе, объединяющая ключевые технологии: распределенную файловую систему, многомодельную базу данных NoSQL, механизм публикации/подписки потоковых событий, ANSI SQL и широкий набор технологий анализа данных с открытым исходным кодом;
- Microsoft HDInsight [31] – служба, работающая в облаке Windows Azure, позволяющая осуществляь быстрый запуск таких популярных платформ с открытым кодом как Apache Hadoop, Spark и Kafka;
- Arenadata Hadoop (ADH) [54] – также российский дистрибутив, в состав которого входят актуальные стабильные версии всех наиболее популярных инструментов, такие как Apache Hive, Apache Spark и Apache Atlas.

Так как рассмотренные дистрибутивы обладают примерно одинаковыми характеристиками в дальнейших исследованиях будут использованы виртуальные машины от Hortonworks.

Выводы по второму разделу

Системы хранения и анализа больших данных строятся по двум архитектурным типам: системы пакетной и/или потоковой обработки больших данных.

Существует большое количество инструментов, применяемых для создания систем хранения и анализа больших данных и образующих экосистему больших данных, которая включает в себя группы инструментов: распределенные файловые системы, инструменты развертывания, базы данных NoSQL и т.д.

Существует четыре вида баз данных NoSQL: столбцовые, документальные, «ключ-значение», графовые. Каждый из видов NoSQL баз данных обладает своими преимуществами и наилучшей применимостью при решении определенного круга задач.

Развертывание систем хранения и анализа больших данных представляет сложную техническую и инженерную задачу. Для облегчения процесса развертывания систем хранения и анализа больших данных могут использоваться специальные дистрибутивы, выполненные в виде виртуальных машин или облачных служб.

3 АЛГОРИТМЫ СИСТЕМ ХРАНЕНИЯ И АНАЛИЗА БОЛЬШИХ ДАННЫХ

Процесс Data Science применительно к большим данным, описанный в первой главе, должен опираться на набор алгоритмов и языков программирования, которые поддерживают все этапы, начиная с подключения к источникам данным до визуализации результатов исследований.

В данной главе, в соответствии с целью исследования, относительно подробно будут рассмотрены алгоритмы первых этапов процесса хранения и анализа больших данных.

Как известно, для программирования систем хранения и анализа больших данных может применяться достаточно большой набор языков программирования: Python, R, Java и т.д. В работе для реализации элементов систем хранения и анализа больших данных будет использован язык Java по следующим причинам:

- платформы Java SE и Java EE отлично подходят для разработки хорошо масштабируемых, высоконагруженных приложений хранения и обработки больших данных;
- язык Java обладает рядом стандартных возможностей для построения систем обработки больших данных, наиболее интересными из которых являются:
    - базовые методы обработки строк, включающие в себя методы класса String, изменяемые класс StringBuilder, потокобезопасный класс StringBuffer, парсер строк – класс StringTokenizer, а также мощный механизм обработки текста, основанный на регулярных выражениях – Regular Expressions;
    - методы функционального программирования посредством лямбда-выражений, введенных в Java 8, позволяют использовать мощные и выразительные средства создания приложений;
    - потоковые возможности коллекций (collections streaming), облегчающую работу с коллекциями данных;

- сторонние Java библиотеки такие как, дают широкий спектр возможностей для более простой и надежной реализации алгоритмов хранения и анализа больших данных:
    - Google Guava (https://github.com/google/guava) и Apache Common Collections (https://commons.apache.org/collections), расширяющие технологию работы с коллекциями;
    - Apache Commons IO (https://commons.apache.org/io) для реализации процесса загрузки/выгрузки данных;
    - AOL Cyclops-React (https://github.com/aol/cyclops-react) для более функционального параллельного потокового обмена;
- поддержка интерактивного вычисления (аналогично терминалу Python) Read-Evaluate-Print Loop (REPL) в Java 9. Поставка Java 9 включает в себя интерактивный терминал jshell, что сделало язык Java полностью пригодным для использования на этапе исследовательского анализа данных – Exploratory Data Analysis (EDA).

3.1 Методы получения, обработки и анализа больших данных

Наука о данных связана с обработкой и анализом больших объемов данных и ставит цель построения моделей данных, которые можно использовать для прогнозирования или достижения конкретной научной или бизнес цели. Выбор или разработка подхода к решению конкретной задачи проблемы зависит от характера самой задачи. Тем не менее можно выделить следующие группы высокоуровневых методов, используемые в процессе работы с большими данными:

- получение данных – Acquiring the data;
- очистка данных – Cleaning the data;
- анализ данных – Analyzing the data.

Рассмотрим подробнее высокоуровневые методы, входящие в указанные группы.

Получение данных. Прежде чем данные могут быть обработаны, они должны быть получены. Данные часто хранятся в различных форматах и поступают из самых разных источников данных, которые можно разделить на внешние и внутренние.

Очистка данных. После того, как данные были получены, их часто необходимо преобразовать в другой формат, прежде чем их можно будет использовать. Кроме того, данные должны быть предварительно обработаны, очищены и приведены в форму, пригодную для последующего анализа.

Анализ данных. Этап анализа данных может быть выполнен с использованием ряда общих методов таких как:

- статистический анализ данных. На этом шаге используется множество статистических методов, позволяющих получить информацию о характере распределений дынных и их связей. Среди статистических методов можно выделить элементарные, такие как расчет характеристик случайных величин – среднего, моды, медианы, квартилей, дисперсии, среднего квадратического отклонения и т.д. Ряд методов можно отнести к продвинутым: проверка статистических гипотез, методы корреляционного и регрессионного анализа.
- методы анализа на основе искусственного интеллекта (ИИ). Эти методы можно сгруппировать в методы машинного обучения, нейронные сети и методы глубокого обучения:
    - методы машинного обучения позволяют создавать алгоритмы, которые могут обучаться без специального программирования для выполнения конкретной задачи;
    - нейронные сети построены вокруг моделей, созданных по образцу нейронной связи мозга;
    - глубокое обучение пытается выявить более высокие уровни абстракции в наборе данных
- анализ текста. Это распространенная форма анализа, которая работает с естественными языками для определения таких характеристик, как

имена людей и мест, взаимосвязь между частями текста и подразумеваемое значение текста.

- визуализация данных. Визуализация является важным инструментом анализа. Именно визуализация данных позволяет лицу принимающему решение воспользоваться результатами анализа, который представляется в числовой форме, практически непригодной длЯ восприятия человеком;
- Обработка и анализ видео, изображений и аудио. Это более специализированная форма анализа, которая становится все более распространенной по мере обнаружения более совершенных методов анализа и появления более быстрых процессоров.

В дополнение необходимо сказать о методах непосредственно не относящихся к хранению и анализу больших данных, но позволяющих создавать высокоэффективные приложения. К таким методам следует отнести методы параллельного, распределенного программирования, а также программирование с использованием графических процессоров (GPU).

В дополнении к сказанному, можно добавить, что методы машинного обучения можно разделить на следующие группы:

- методы контролируемого обучения. Данные методы обучаются на наборах помеченных данных. Для пометки данных требуется участие человека. После обучения алгоритмы способны различать результаты. Наивный байесовский классификатор является представителем данной группы;
- методы неконтролируемого обучения. Эти методы не требуют пометки данных и стараются отыскать закономерности в данных без участия человека. Метод главных компонент относится к методам неконтролируемого обучения;
- Методы с частичным контролем могут работать как с использованием помеченных данных, так и без пометки.

В соответствии с темой исследования необходимо провести анализ способов и технологий, используемых на первых стадиях процесса хранения и анализа больших данных: получение, очистка и загрузка данных в распределенную файловую систему средствами языка Java.

3.2 Способы и технологии получения больших данных

Как уже отмечалось выше, для последующего анализа больших данных необходимо использовать не только внутренние данные компании, но и внешние данные.

Как правило, если компания осознает внедрение систем хранения и анализа больших данных, то внутренние данные уже хранятся в различных корпоративных хранилищах в согласованном виде. Внутренние данные компании хранятся в базах данных, витринах данных, хранилищах (складах) данных и озерах данных. Как правило, технических доступ к внутренним данным предприятия не представляет трудностей, поскольку осуществляется посредством хорошо известных технологий, таких как JDBC, JPA (Java Persistence Architecture), Java Connectors или технологий, основанных на web сервисах (например, Restful), реализующих соединения типа B2B.

Важность загрузки данных из внешних источников, а значит и наличие таких источников давно осознано в мире. Более того, данные с этой точки зрения становятся весьма дорогостоящим продуктом. Например, в марте 2019 года стало известно о том, что мэрия города Москвы с 2015 года потратила 516 миллиардов рублей на покупку у сотовых операторов данных о перемещении жителей по городу. Эти данные используются для анализа передвижения граждан и оптимизации транспортных маршрутов [56].

В Российской Федерации ведется планомерная работа по созданию системы ресурсов открытых данных, особый импульс эта работа приобрела в связи с выполнением национальных проектов в целом и проекта «Цифровая экономика» [1], в частности.

Среди важнейших официальных ресурсов можно указать следующие:

- портал открытые данные России [21];

- портал открытых данных правительства Москвы [59];
- открытые данные Федеральной службы государственной статистики [57];
- открытые данные ФНС России [58].

В качестве зарубежных ресурсов открытых данных можно привести следующие примеры:

- официальный ресурс правительства США, предоставляющий более 100 000 наборов данных из разных категорий [20];
- открытые данные мирового банка (World Bank Group), охватывающие демографию и достаточное количество экономических показателей и показателей развития со всего мира [51];
- открытые данные Международного валютного фонда о международных финансах, ставках долга, валютных резервах, ценах на сырьевые товары и инвестициях [34];
- Проект European Data Portal, содержащьй свыше 433 800 наборов открытых данных со всего континента [24].

Перечень из тридцати ресурсов открытых данных приведен в обзоре [37].

Кроме того, традиционными внешними источниками внешних данных являются социальные сети и различные сервисы, такие как YouTube, Wikipedia и т.д.

Для целей обучения и тестирования алгоритмов, применяемых для хранения и анализа больших данных могут быть рекомендованы ресурсы, позволяющие генерировать наборы данных. Например, сервис generatedata.com [25] позволяет сгенерировать набор и сохранить его в различных форматах (рисунок 3.1).

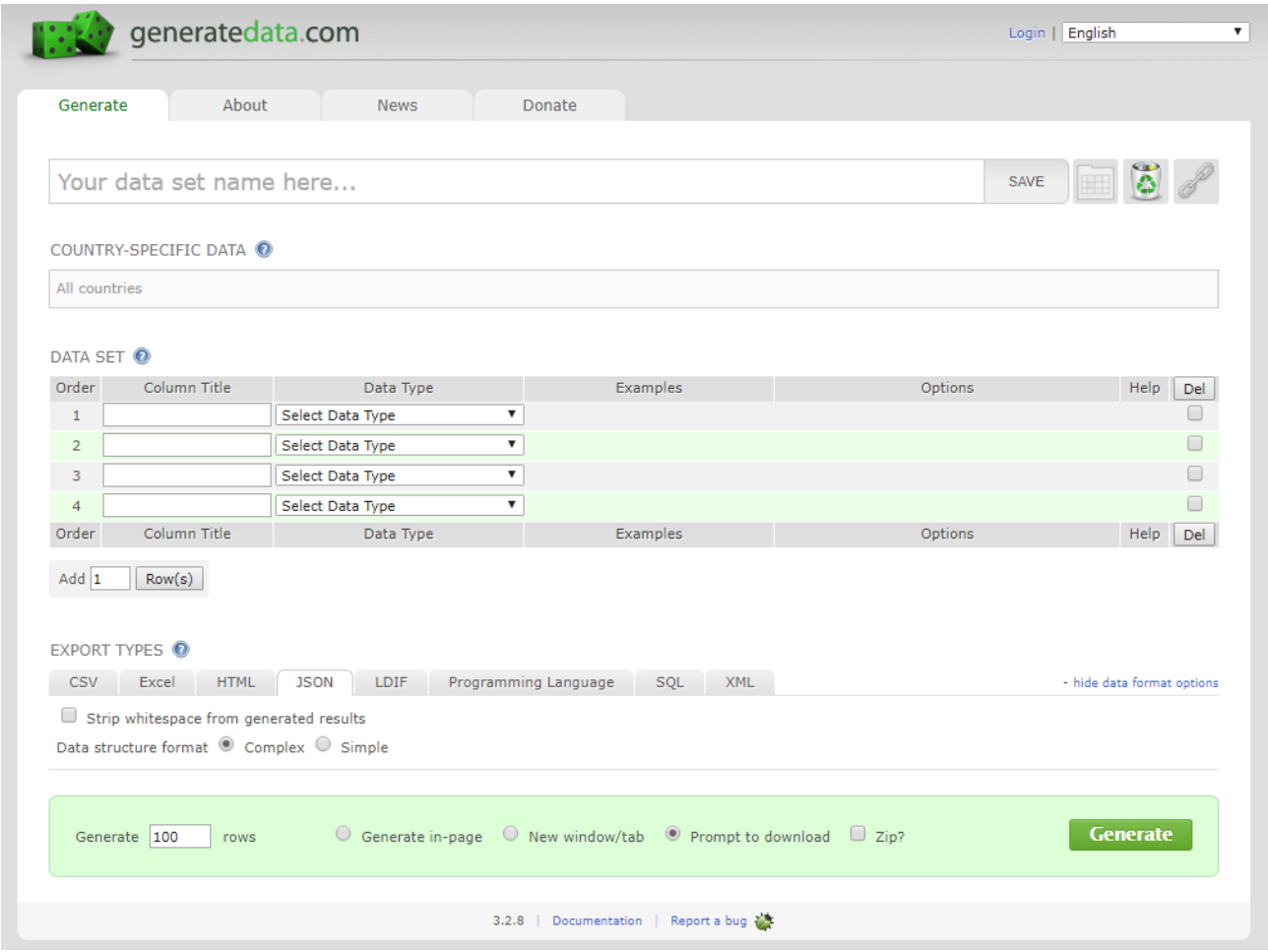


Рисунок 3.1 – Сервис generatedata.com

Рассмотрим способы получения данных из внешних источников более подробно. По мнению автора можно разделить способы получения внешних данных на два типа:

1. Методы, основанные на анализе контента сайтов.
2. Методы, основанные на использовании специальных API интернет ресурсов.

К методам первого типа можно отнести веб-скрепинг (соскабливание, scraping) и веб-сканирование (crawling). Мы можем получить доступ к данным из Интернета, загрузив определенные файлы или с помощью процесса, известного как веб-скрепинг, который включает извлечение содержимого веб-страницы. Второй метод – веб-сканирование, основан на разработке приложения, которое исследует веб-сайт на возможность применения его

контента в исследовании и следуют по гиперссылкам для анализа других потенциально релевантных страниц.

Разработчики больших интернет ресурсов, таких как социальные сети и YouTube, учитывая негативные эффекты от массового использования на их сайтах алгоритмов скрепинга и сканирования, разработали специальное API, через которые можно получать необходимый для исследования контент. Что и определило появление методов второго типа.

3.2.1 Вэб-скрепинг

Веб-скрепинг – это процесс извлечения данных и метаданных с вэб страницы [66]. По своей сути для осуществления веб-скрепинга необходимо реализовать HTML парсер (веб-скебок), поскольку результат http запроса представляется обычно в виде HTML кода.

Анализ технологий веб-скрепинга применительно к языку Java показал, что наилучшим решением является использование открытой Java библиотеки Jsoup, представляющей из себя HTML парсер. Парсер Jsoup позволяет также проводить поиск определенных элементов HTML разметки и либо их использовать для последующего анализа, либо, наоборот, фильтровать эти элементы, проводя очистку содержимого HTML страницы.

Рассмотрим пример JSoup парсинга (рис. 3.2) статьи википедии, посвященной большим данным [52]. Сначала через jsoup соединение создается документ, у который загружается содержимое HTML страницы по указанному адресу. Далее, используя различные методы объекта document получает такие элементы, как название, тело, список гиперссылок и т.п. Далее можно, например, обрабатывать тело статьи, представляющий собой стоку, используя разбивку на лексемы при помощи стандартных классов (String, StringTokenizer), технологии регулярных выражений (Regular Expressions) или сторонние библиотеки Java.

```
import java.io.IOException;
import java.util.List;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.select.Elements;

public class JsoupParser {

    //https://ru.wikipedia.org/wiki/Большие_данные
    private static String url =
"https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B5
_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5";

    public static void main(String[] args) {
        try {
            Document document = Jsoup.connect(url).get();
            //get a title
            String title = document.title();
            System.out.println("Название статьи: " + title);
            //get a body
            Elements body = document.select("body");
            System.out.println(" Содержание: " + body.text());
            //get links
            Elements links = document.select("a[href]");
            //System.out.println(links.text());
            List<String> linksList = links.eachText();
            int linksNum = linksList.size();
            System.out.println("Число гиперссылок: " + linksNum + "\n
Первые пять:");
            for(int i=0; (I < 5) & (I < linksNum); i++) {
                System.out.println(linksList.get(i));
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

}
```

Рисунок 3.2 – Код JSoup парсера,

реализующего веб-скрепинг страницы википедии

Пример работы JSoup парсера приведен на рисунке 3.3

3.2.2 Веб-сканирование

Веб-сканирование (Web crawling) – это процесс обхода множества взаимосвязанных веб-страниц и извлечения соответствующей информации из этих страниц.

Название статьи: Большие данные — Википедия

—

big data, [' bɪ ɡ ' deɪ tə])
— обозначение структурированных и неструктурированных данных огромных объёмов и значительного многообразия, эффективно обрабатываемых горизонтально масштабируемыми программными инструментами, появившимися в конце 2000-х годов и альтернативных традиционным системам управления базами данных и решениям класса Business Intelligence[1][2][3]. В широком смысле о «больших данных» говорят как о социально-экономическом феномене, связанном с появлением технологических возможностей анализировать огромные массивы данных, в некоторых проблемных областях — весь мировой объём данных, и вытекающих из этого трансформационных последствий[4]. В качестве определяющих характеристик для больших данных традиционно выделяют «три V»: объём (англ. volume, в смысле величины физического объёма), скорость (velocity в смыслах как скорости прироста, так и необходимости высокоскоростной обработки и получения результатов), многообразие (variety, в смысле возможности одновременной обработки различных типов структурированных и полуструктурированных данных)[5][6]; в дальнейшем возникли различные вариации и интерпретации этого признака [⇨].
…
некоммерческой организации Wikimedia Foundation, Inc. Политика конфиденциальности Описание Википедии Отказ от ответственности Свяжитесь с нами Разработчики Соглашение о cookie Мобильная версия
Число гиперссылок: 435
Первые пять:
Перейти к навигации
Перейти к поиску
англ.
' bɪ ɡ ' deɪ tə
неструктурированных данных

Рисунок 3.3 – Парсинг страницы статьи русскоязычной википедии, посвященной большим данным

Процесс веб-сканирования осуществляется путем обработки текущей страницы и последующего перехода по ссылкам на странице. Веб-сканирование полезно, когда несмотря на наличие доступных наборов данных, все же может потребоваться сбор данных непосредственно из Интернета. Например, некоторые источники, такие как новостные сайты, ленты сообщений в социальных сетях постоянно обновляются, и время от времени их необходимо пересматривать.

Веб-сканер – это приложение, которое посещает различные сайты и собирает информацию. Процесс веб-сканирования состоит из серии шагов [Java Data Science]:

1. Определение URL для посещения.
2. Получение страницы.
3. Разбор страницы.
4. Извлечение соответствующего контента.
5. Извлечение соответствующих URL для дальнейшего посещения.

Указанная последовательность шагов повторяется для каждого посещенного URL.

Однако, существует несколько проблем, которые необходимо учитывать при реализации веб-сканирования:

- важность страницы: нет необходимости обрабатывать нерелевантные страницы;
- обработка конкретного контента: например, будет обрабатываться только HTML контент, а переход по ссылкам на изображения осуществляться не будет;
- исключение бесконечного анализа (Spider traps): необходимо обходить сайты, которые могут привести к бесконечному количеству запросов. Указанная ситуация может происходить с динамически генерируемыми страницами, когда один запрос ведет к другому;
- исключение повторения: необходимо исключить сканирования одной и той же страницы более одного раза;
- «вежливость» сканера: не следует делать чрезмерного количества запросов на сайт. Для этого необходимо просматривать файлы robot.txt, которые указывают, какие части сайта не следует сканировать.

Как следует из приведенной информации создание веб-сканера с нуля является сложной задачей. Поэтому можно использовать один веб-сканеров с открытым исходным кодом, среди которых можно указать:

- crawler4j (https://github.com/yasserg/crawler4j);
- Nutch (http://nutch.apache.org);
- WebSPHINX (http://www.cs.cmu.edu/~rcm/websphinx);
- JSpider (http://j-spider.sourceforge.net);
- Heritrix (https://webarchive.jira.com/wiki/display/Heritrix).

Рассмотрим построение веб-сканера на основе библиотеки crawler4j. Для этого необходимо разработать два класса. Первый класс TltsuCrawler будет реализовывать логику сканера, обследующего контент сайта Тольяттинского государственного университета [62]. Второй класс представляет из себя контроллер или приложение. Диаграмма указанных классов представлена на рисунке 3.4.

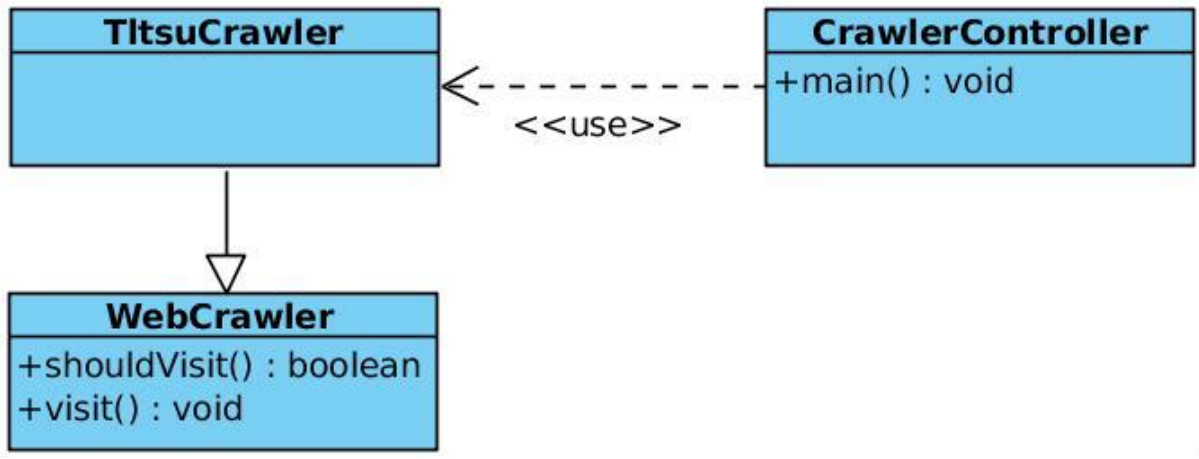


Рисунок 3.4 – Диаграмма классов реализации веб-сканера

Класс веб-сканера перегружает два метода библиотечного класса WenCrawler:shoudVisit() (определяет фильтр посещения веб-страниц) и visit() (определяет алгоритм обработки страницы). Код метода visit(), изображенный на рисунке 3.5, использует регулярное выражение для отсева ссылок, ведущих на ресурсные (css), мультимедийные (gif, png, …) и архивные (zip, gz) файлы, которые в данном примере пропускаются для отбора.

```
public class TltsuCrawler extends WebCrawler {
    private final static Pattern FILTERS =
Pattern.compile(".*(\\.(css|js|gif|jpg|png|mp3|mp4|zip|gz))$");

    /**
     * Specify whether the given url should be crawled
     */

    @Override
    public boolean shouldVisit(Page referringPage, WebURL url) {
        System.out.println("shouldVisit: " + url.getURL().toLowerCase());

        String href = url.getURL().toLowerCase();
        boolean result = !FILTERS.matcher(href).matches();
        if(result) {
         System.out.println("Visited " + url.getURL().toLowerCase());
        } else {
         System.out.println("Not visited " +
                 url.getURL().toLowerCase());
        }
        return result;
    }
```

Рисунок 3.5 – Фрагмент кода метода фильтрации страниц веб-сканера

Код метода visit() веб-сканера представлен на рисунке 3.6. Данный метод сначала проверяет, что текущая страница имеет разметку HTML и может быть разобрана HTML парсером. Далее страница анализируется по ряду параметров, которые могут быть использованы для дальнейшего анализа. В примере запись данных идет на консоль, хотя можно организовать вывод данных в файл, для его дальнейшей предварительной обработки и загрузки в систему хранения больших данных.

Фрагменты работы веб-сканера двумя вычислительными потоками, на глубину трех уровней и максимальным количеством обозреваемых страниц, равным 50, приведен на рисунке 3.7.

```
    /**
     * This function is called when a page is fetched and ready
     * to be processed by the program.
     */
    @Override
    public void visit(Page page) {
        String url = page.getWebURL().getURL();
        System.out.println("URL: " + url);

        if (page.getParseData() instanceof HtmlParseData) {
            HtmlParseData htmlParseData = (HtmlParseData)
page.getParseData();
            String text = htmlParseData.getText(); //extract text
from page
            String html = htmlParseData.getHtml(); //extract html
from page
            Set<WebURL> links = htmlParseData.getOutgoingUrls();

            System.out.println("----------------------------------
------------------------");
            System.out.println("Page URL: " + url);
            System.out.println("Text length: " + text.length());
            System.out.println("Html length: " + html.length());
            System.out.println("Number of outgoing links: " +
links.size());
            System.out.println("----------------------------------
------------------------");
        }
    }
```

Рисунок 3.6 – Фрагмент кода метода обработки веб-сканером HTML страницы

В соответствии с приведенным логом работы веб-сканера, были пройдены ряд HTML страниц и по ним сканер выделил необходимую информацию. Лог также показывает, что ряд ресурсов не был просканирован (на рисунке файл extension.min.js), как не прошедший фильтрацию. После завершения работы всех вычислительных потоков сканера и ожидания в 10 секунд приложение завершило свою работу.

3.2.3 Доступ к данным через API Интернет ресурсов

Как уже отмечалось выше ряд интернет ресурсов является хранилищами огромной и разнообразной информации, которая интересна для анализа данных большому количеству сторон.

```
shouldVisit: http://tltmuseum.ru/index.php?format=feed&type=atom
Visited http://tltmuseum.ru/index.php?format=feed&type=atom
shouldVisit: http://tltmuseum.ru/foto/2010-god.html
Visited http://tltmuseum.ru/foto/2010-god.html
shouldVisit: http://tltmuseum.ru/images/stories/partneri/thumbnails/thumb_азот%20копия.jpg
Visited http://tltmuseum.ru/images/stories/partneri/thumbnails/thumb_азот%20копия.jpg
shouldVisit: http://daytlt.ru/
Visited http://daytlt.ru/
…
16:29:46.970 [Crawler 2] DEBUG edu.uci.ics.crawler4j.crawler.WebCrawler - Not visiting: http://tltmuseum.ru/templates/youtempl/css/blue.css as per the server's "robots.txt" policy
…
shouldVisit: http://tltmuseum.ru/news.html
Visited http://tltmuseum.ru/news.html
URL: http://tltmuseum.ru/
-------------------------------------------------------------
Page URL: http://tltmuseum.ru/
Text length: 13336
Html length: 57000
Number of outgoing links: 185
-------------------------------------------------------------
16:29:56.249 [Crawler 2] DEBUG edu.uci.ics.crawler4j.crawler.WebCrawler - Skipping: https://www.tltsu.ru/bitrix/js/main/loadext/extension.min.js?15409606651304 as it contains binary content which you configured not to crawl
…
16:30:17.086 [Thread-0] INFO edu.uci.ics.crawler4j.crawler.CrawlController - It looks like no thread is working, waiting for 10 seconds to make sure...
16:30:27.093 [Thread-0] INFO edu.uci.ics.crawler4j.crawler.CrawlController - No thread is working and no more URLs are in queue waiting for another 10 seconds to make sure...
16:30:37.093 [Thread-0] INFO edu.uci.ics.crawler4j.crawler.CrawlController - All of the crawlers are stopped. Finishing the process...
16:30:37.094 [Thread-0] INFO edu.uci.ics.crawler4j.crawler.CrawlController - Waiting for 10 seconds before final clean up...
16:30:47.327966 application stopped
```

Рисунок 3.7 – Результат работы веб-сканера

Если для сбора и последующего анализа информации использовать ранее рассмотренные способы веб-скрепинга и веб-сканирования, то указанные ресурсы могут блокированы на длительное время такими запросами на получение данных. Поэтому ресурсы социальных сетей, хранилища информации и иные ресурсы, специализирующиеся по предоставлении данных в том числе и для их анализа, разрабатывают специализированное API посредством которого возможно не только искать и скачивать необходимые данные, но и работать с метаинформацией.

Рассмотрим доступ к открытым данным на портале открытых данных Российской Федерации data.gov.ru. Доступ к открытым данным на портале

помимо непосредственного скачивания с веб-страниц возможен двумя способами: использование специального API и через интерактивные SPARQL-запросы. Рассмотрим доступ к данным через специализированный API [60]. Для работы с API требуется получить личный ключ, который доступен после регистрации на Портале. Получить данные можно в двух форматах JSON и XML. Так как формат JSON наиболее подходит для использования в системах хранения и анализа больших данных, то мы будем использовать именно этот формат данных.

На рисунке 3.8 приведен фрагмент кода доступа к порталу открытых данных РФ и запрос на перечень всех имеющихся наборов. Сначала формируется запрос к набору данных в соответствии с правилами портала (в коде скрыт реальный ключ доступа). Затем с помощью IOUtils библиотеки apache.commons.io формируется массив JSON. После чего определяется его длина и происходит вывод не более пяти элементов.

```
public class Main {

    public static void main(String[] args) {
        try {
            URL url = new URL ("https://data.gov.ru/api/dataset/?" +
            "access_token=XXX");

            JSONArray datasets = new JSONArray(
                IOUtils.toString(url, Charset.forName("UTF-8")));
            int numRec = datasets.length();
            System.out.println(numRec);
            JSONObject e;
            for (int i=0; (i < 5) && (i < numRec); i++) {
                e = (JSONObject) datasets.get(i);
                System.out.println(e.toString());
            }
        } catch (MalformedURLException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

Рисунок 3.8 – Получение доступных наборов данных на портале открытых данных РФ

Пример вывода приведен на рисунке 3.9. Получено всего 10 000 источников. Данное количество источников является максимальным для простых запросов.

Модифицируя запросы к порталу, можно сузить диапазон поиска, например, найти наборы данных, относящихся к разделу «Образование/Education» (рис. 3.10).

10000

{"identifier":"8912001599-muzikalka","organization":"8912001599","topic":"Education","organization_name":"Муниципальное бюджетное учреждение дополнительного образования \"Толькинская детская школа искусств\" муниципального образования Красноселькупский район в Ямало-Ненецком автономном округе","title":"Музыкальная школа с. Толька ЯНАО"}

{"identifier":"8901012609-regprojects","organization":"8901012609","topic":"Government","organization_name":"Аппарат Губернатора Ямало-Ненецкого автономного округа","title":"Региональные проекты, реализуемые аппаратом Губернатора Ямало-Ненецкого автономного округа"}

{"identifier":"8901017727-reestrteplo","organization":"8901017727","topic":"Economics","organization_name":"Департамент тарифной политики, энергетики и жилищно-коммунального комплекса Ямало-Ненецкого автономного округа","title":"Реестр организаций, которым установлена плата за технологическое присоединение к системе теплоснабжения"}

{"identifier":"8901017283-poiskoviki","organization":"8901017283","topic":"Government","organization_name":"Департамент молодёжной политики и туризма Ямало - Ненецкого автономного округа","title":"Перечень поисковых отрядов Ямало-Ненецкого автономного округа"}

{"identifier":"8901003315-culturalheritage","organization":"8901003315","topic":"Culture","organization_name":"Администрация муниципального образования город Салехард в Ямало-Ненецком автономном округе","title":"Объекты культурного наследия города Салехарда"}

Рисунок 3.9 – Первые пять доступных наборов данных на портале открытых данных РФ

```
URL url = new URL ("https://data.gov.ru/api/dataset/?" +
                   "topic=Education&" +
                   "access_token=XXX");
```

Рисунок 3.10 – Запрос на доступные наборы данных на портале открытых данных РФ из раздела «Образование»

На рисунке 3.11 приведен пример отклика портала. В разделе «Образование» было найдено 1201 набор и показаны первые пять наборов.

1201

{"identifier":"7710539135-DO","organization":"7710539135","topic":"Education","organization_name":"Министерство образования и науки Российской Федерации","title":"Сведения о функционировании системы дошкольного образования"}

{"identifier":"7710539135-trud","organization":"7710539135","topic":"Education","organization_name":"Министерство образования и науки Российской Федерации","title":"Трудоустройство выпускников учреждений профессионального образования"}

{"identifier":"7710539135-ob_obr","organization":"7710539135","topic":"Education","organization_name":"Министерство образования и науки Российской Федерации","title":"Сведения о функционировании системы общего образования"}

{"identifier":"7710539135-SPO","organization":"7710539135","topic":"Education","organization_name":"Министерство образования и науки Российской Федерации","title":"Сведения о функционировании системы среднего профессионального образования"}

{"identifier":"7710539135-podved","organization":"7710539135","topic":"Education","organization_name":"Министерство образования и науки Российской Федерации","title":"Перечень подведомственных организаций Минобрнауки России"}

Рисунок 3.11 – Первые пять доступных наборов данных из раздела образование на портале открытых данных РФ

Анализируя полученные списки, можно выбрать набор для скачивания. Код, приведённый на рисунках 3.8-3.11 показывает алгоритм скачивания набора №7710539135-trud, имеющего название «Трудоустройство выпускников учреждений профессионального образования». В соответствии с описанием API портала необходимо определить версию набора. Мы будем использовать последнюю версию. Алгоритм нахождения последней версии искомого набора приведен на рисунке 3.12. Сначала формируется url запроса, по которому получаем набор JSON объектов, отвечающих за версии. У каждой версии

набора открытых данных представлено единственное свойство created – уникальная дата версии набора открытых данных. При обращении к первому элементу, мы получаем последнюю версию. Так в нашем случае был получен результат в формате:

{"created":"20150609T121501"}

```
public class Main {

    static final String ACCESS_TOKEN = "/?access_token=XXX";
    static final String ACCESS_URL =
            "https://data.gov.ru/api/json/dataset" +
            "/7710539135-trud" +
            "/version";

    public static void main(String[] args) {
        try {
            //получаем набор версий
            //url для получения набора версий
            URL url = new URL (ACCESS_URL + ACCESS_TOKEN);
            JSONArray versions = new JSONArray(
                IOUtils.toString(url, Charset.forName("UTF-8")));
            int numRec = versions.length();
            JSONObject lastVersion = (JSONObject) versions.get(0);
            String created = lastVersion.getString("created");
…
            } catch (MalformedURLException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

Рисунок 3.12 – Определение последней версии запрашиваемого набора данных

Далее необходимо получить информацию по искомой версии набора. Алгоритм получения информации представлен на рисунке 3.13. Сначала модифицируется url под версию набора. После этого программа обращается к порталу и получает список JSON объектов, отвечающих за информацию о наборе, которая, в нашем случае, представляется в следующем виде:

{"provenance":"Внесены изменения в описание набора данных. Опубликована дополнительная информация",
"created":"20150609T121501",
"format":"csv",
"source":"https://data.gov.ru/sites/default/files/opendata/7710539135-trud/data-2015-06-09T12-15-01-structure-2013-07-03T00-00-00.csv",
"source_id":"832895","updated":"20150609T121501"}

```
//получаем адрес последней версии набора данных
////url для получения информации о наборе данных
    url = new URL (ACCESS_URL +
            "/" + created +
            ACCESS_TOKEN);
    JSONArray dataset = new JSONArray(
            IOUtils.toString(url, Charset.forName("UTF-8")));
    JSONObject data = (JSONObject) dataset.get(0);
    String dataURI = data.getString("source");
```

Рисунок 3.13 – Получение системной информации о версии набора данных

Далее происходит выделение имени файла и, используя каналы обмена происходит запись данных с портала в файл на диске (рис. 3.14). В нашем случае файл имеет имя:

data-2015-06-09T12-15-01-structure-2013-07-03T00-00-00.csv

```
//выделяем имя файла из url
String fname = dataURI.substring(
            dataURI.lastIndexOf('/')+1, dataURI.length());
//transfer bytes between 2 Channels
//without buffering them into the application memory
//read the file from our URL
ReadableByteChannel readableByteChannel =
        Channels.newChannel(new URL(dataURI).openStream());
//The bytes read from the ReadableByteChannel
//will be transferred to a FileChannel corresponding
//to the file that will be downloaded
try (FileOutputStream fileOutputStream = new
        FileOutputStream(fname);
    FileChannel fileChannel = fileOutputStream.getChannel()) {
    /use the transferFrom() method from the
    //ReadableByteChannel class
    //to download the bytes from the given URL to our FileChannel
    fileOutputStream.getChannel()
        .transferFrom(readableByteChannel, 0, Long.MAX_VALUE);
}
```

Рисунок 3.14 – Использование каналов для сохранения набора данных на диск

После выполнения кода файл может быть загружен в систему хранения и анализа больших данных для последующей обработки. Полный код приложений, загружающих данные с портала открытых данных Российской Федерации представлен в Приложениях В-Г. Фрагмент содержимого файла data-2015-06-09T12-15-01-structure-2013-07-03T00-00-00.csv приведен на рисунке 3.15.

region	npo	ntu	our	ppa	tu	all_vyp
Белгородская область	4738	720	1050	5006	18292	29806
Белгородская область	60	10	6	8	114	198
Белгородская область	122	4	4	26	120	276
Белгородская область	428	200	78	180	2064	2950
Белгородская область	58	2	14	6	38	118
Белгородская область	378	24	130	136	1270	1938
Белгородская область	436	62	112	48	1026	1684
…						
Брянская область	3644	922	674	6748	11884	23872
Брянская область	8	14	8	32	62	124
Брянская область	98	18	8	32	110	266
Брянская область	586	174	112	556	2422	3850
Брянская область	20	6	4	14	18	62
Брянская область	186	44	126	98	824	1278
Брянская область	216	52	56	768	732	1824
Брянская область	4	6	6	24	30	70
Брянская область	0	0	0	0	0	0

Брянская область	68	20	0	278	216	582
Брянская область	334	58	2	870	604	1868
Брянская область	0	0	0	0	0	0
Брянская область	0	0	0	0	0	0
Брянская область	0	0	0	0	0	0
Брянская область	210	74	4	1304	890	2482
Брянская область	14	14	0	10	24	62
Брянская область	22	0	0	56	52	130

Рисунок 3.15 – Фрагмент содержимого набора данных портала открытых данных Российской Федерации

Технологии доступа к социальным сетям и сервисам типа YouTube хорошо описаны в литературе и не являлись предметом исследования в работе.

3.3 Очистка данных

Реальные данные часто являются грязными – содержат разного рода ошибки – и неструктурированными, поэтому перед использованием они должны быть переработаны. Данные могут содержать ошибки, иметь повторяющиеся записи, существовать в неправильном формате или быть противоречивыми. Процесс решения этих типов проблем называется очисткой данных. Очистка данных также называется обработкой, массированием, изменением формы или обработкой данных [66]. Слияние данных, при котором данные из нескольких источников объединяются, часто считается очисткой данных.

Очистка данных является неотъемлемым этапом работы с большими данными, поскольку любой анализ, основанный на неточных данных, может привести к ошибочным результатам. Таким образом, необходимо убедиться, что данные, с которыми мы работаем, являются качественными данными. Качество данных включает в себя:

- пригодность (Validity) требование обеспечения правильной формы или структуры данных;
- точность (Accuracy): значения в данных действительно являются репрезентативными для набора данных;
- полнота (Completeness): нет отсутствующих элементов;

- согласованность (Consistency): изменения в данных синхронизированы;
- однородность (Uniformity): используются те же единицы измерения.

Можно указать несколько методов и инструментов, используемых для очистки данных:

- обработка разных типов данных;
- очистка и манипулирование текстовыми данными;
- заполнение недостающих данных;
- проверка данных.

Для проведения очистки данных с использованием языка Java, как и прежде, могут применяться как стандартные технологии, так и сторонние библиотеки.

Одной из первых задач, как правило, является проверка корректности форматов входных данных, которые часто могут поступать в виде:

- CSV наборов;
- таблиц различного вида;
- файлов в формате pdf;
- файлов в формате JSON.

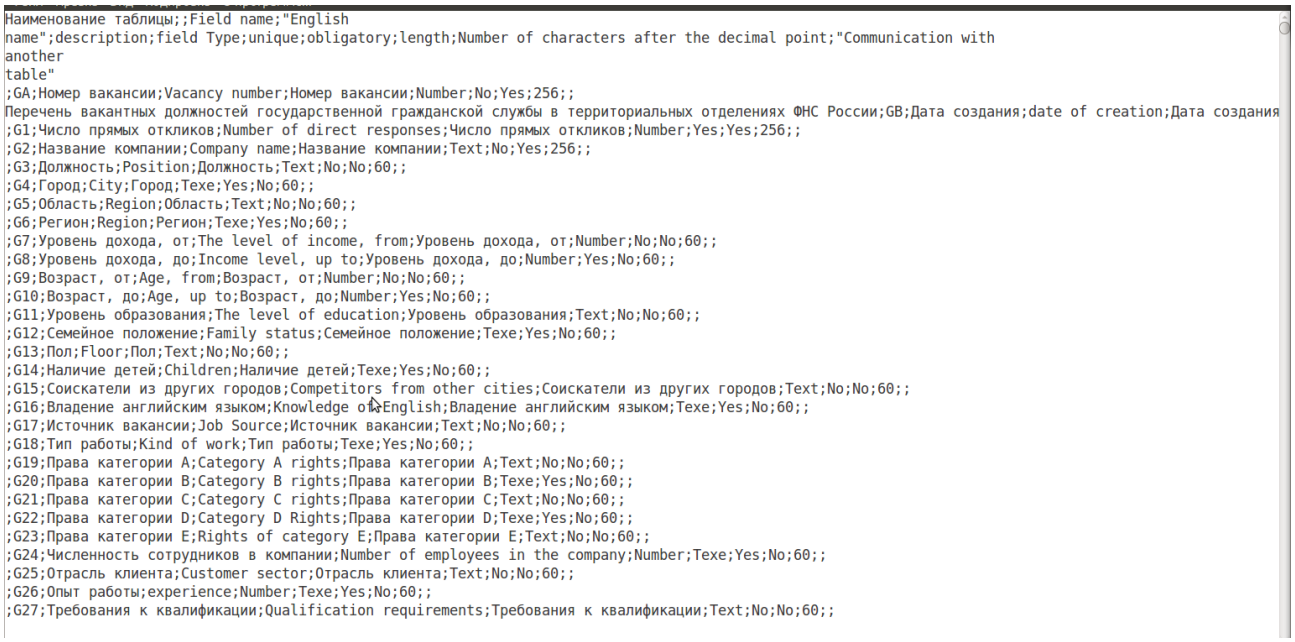
Отдельную задачу представляет проверка файлов мультимедиа.

При использовании порталов открытых данных, как правило, приходится иметь дело с форматом CSV. Эти файлы часто получены путем автоматической выгрузки из информационных систем, однако и они нуждаются в очистке. Рассмотрим анализ структуры открытых данных с сайта ФНС России [58]. В качестве примера рассмотрим «Перечень вакантных должностей государственной гражданской службы в территориальных органах ФНС России», он и описание которого доступно по ссылке https://www.nalog.ru/opendata/7707329152-job/. Фрагмент данных из указанного набора представлен на рис. 3.16., а фрагмент описания структуры – на рис. 3.17.

```
GA,GB,G1,G2,G3,G4,G5,G6,G7,G8,G9,G10,G11,G12,G13,G14,G15,G16,G17,G18,G19,G20,G21,G22,G23,G24,G25,G26,G27,,
1,12.10.2018,,Федеральная налоговая служба,Консультант отдел анализа и прогнозирования налоговых доходов Аналитическое управление,Москва,Москва,Централь
2,12.10.2018,,Федеральная налоговая служба,Ведущий специалист-эксперт отдела анализа и прогнозирования налоговых доходов Аналитического управления,Москв
3,12.10.2018,,Федеральная налоговая служба,Консультант отдела анализа деятельности налоговых органов Аналитического управления,Москва,Москва,Центральный
4,12.10.2018,,Федеральная налоговая служба,Консультант Контрольного управления,Москва,Москва,Центральный округ,27000,30000,,,Высшее,,не имеет значения,н
5,12.10.2018,,Федеральная налоговая служба,Советник отдела анализа контрольной работы и мониторинга крупнейших налогоплательщиков Контрольного управлени
6,12.10.2018,,Федеральная налоговая служба,Консультант отдела анализа контрольной работы и мониторинга крупнейших налогоплательщиков Контрольного управл
7,12.10.2018,,Федеральная налоговая служба,Консультант отдела анализа налоговых рисков и развития технологий планирования и проведения налоговых провероо
8,12.10.2018,,Федеральная налоговая служба,Консультант контрольно-аналитического отдела Контрольного управления,Москва,Москва,Центральный округ,27000,30
9,12.10.2018,,Федеральная налоговая служба,Советник отдела специальных налоговых режимов Управления налогообложения юридических лиц,Москва,Москва,Центра
10,12.10.2018,,Федеральная налоговая служба,Советник отдела акцизов и ресурсных налогов Управления налогообложения юридических лиц,Москва,Москва,Централ
11,12.10.2018,,Федеральная налоговая служба,Консультант отдела учета и отчетности по центральному аппарату Финансового управления,Москва,Москва,Централь
12,12.10.2018,,Федеральная налоговая служба,Консультант отдела учета и отчетности по центральному аппарату Финансового управления,Москва,Москва,Централь
13,12.10.2018,,Федеральная налоговая служба,Консультант отдела прикладного программного обеспечения Управления информационных технологий,Москва,Москва,Ц
14,12.10.2018,,Федеральная налоговая служба,Ведущий специалист-эксперт отдела управления проектами Управления информационных технологий,Москва,Москва,Це
15,12.10.2018,,Федеральная налоговая служба,Консультант отдела рассмотрения судебных споров с крупнейшими налогоплательщиками сферы добычи и переработки
16,12.10.2018,,Федеральная налоговая служба,Консультант отдела информационных ресурсов и отчетности Управления по работе с задолженностью,Москва,Москва,
17,12.10.2018,,Федеральная налоговая служба,Консультант отдела взаимодействия с правоохранительными органами Управления по работе с задолженностью,Москв
18,12.10.2018,,Федеральная налоговая служба,Главный специалист-эксперт отдела рассмотрения споров с юридическими лицами по решениям налоговых органов фе
19,12.10.2018,,Федеральная налоговая служба,Главный специалист-эксперт отдела методологии и стратегического развития Управления досудебного урегулирован
20,12.10.2018,,Федеральная налоговая служба,Консультант отдела материально-технического обеспечения Административно-контрольного управления,Москва,Москв
21,12.10.2018,,Федеральная налоговая служба,Главный специалист-эксперт отдела материально-технического обеспечения Административно-контрольного управлен
22,12.10.2018,,Федеральная налоговая служба,Ведущий специалист-эксперт отдела социального обеспечения Административно-контрольного управления,Москва,Мос
23,12.10.2018,,Федеральная налоговая служба,Советник отдела контроля выполнения технологических процессов Управления модернизации налоговых органов,Моск
24,12.10.2018,,Федеральная налоговая служба,Главный специалист-эксперт отдела контроля выполнения технологических процессов Управления модернизации нало
25,12.10.2018,,Федеральная налоговая служба,Главный специалист-эксперт отдела налогового контроля цен № 1 Управления трансфертного ценообразования,Москв
26,12.10.2018,,Федеральная налоговая служба,Заместитель начальника отдела отдела налогового контроля цен № 2 Управления трансфертного ценообразования,Мо
27,12.10.2018,,Федеральная налоговая служба,Консультант отдела налогового контроля цен № 2 Управления трансфертного ценообразования,Москва,Москва,Центра
28,12.10.2018,,Федеральная налоговая служба,Советник отдела обмена информацией Управления трансфертного ценообразования,Москва,Москва,Центральный округ,
29,12.10.2018,,Федеральная налоговая служба,Консультант отдела обмена информацией Управления трансфертного ценообразования,Москва,Москва,Центральный окр
30,12.10.2018,,Федеральная налоговая служба,Консультант отдела обмена информацией Управления трансфертного ценообразования,Москва,Москва,Центральный окр
31,12.10.2018,,Федеральная налоговая служба,Заместитель начальника отдела отдела методологии, автоматизации и договорной работы Управления трансфертного
32,12.10.2018,,Федеральная налоговая служба,Ведущий специалист-эксперт отдела методологии, автоматизации и договорной работы Управления трансфертного це
33,12.10.2018,,Федеральная налоговая служба,Консультант отдела организации учёта юридических и физических лиц Управления регистрации и учёта налогоплате
34,12.10.2018,,Федеральная налоговая служба,Ведущий специалист-эксперт отдела организации учёта юридических и физических лиц Управления регистрации и уч
35,12.10.2018,,Федеральная налоговая служба,Главный специалист-эксперт отдела методологии ведения, мониторинга государственных реестров и предоставления
```

Рисунок 3.16 – Фрагмент данных с сайта ФНС России

Требуется проверить набор на пригодность его загрузки в систему хранения и анализа больших данных, а при наличии ошибок – вывести отчет об ошибках.



```
Наименование таблицы;;Field name;"English
name";description;field Type;unique;obligatory;length;Number of characters after the decimal point;"Communication with
another
table"
;GA;Номер вакансии;Vacancy number;Номер вакансии;Number;No;Yes;256;;
Перечень вакантных должностей государственной гражданской службы в территориальных отделениях ФНС России;GB;Дата создания;date of creation;Дата создания
;G1;Число прямых откликов;Number of direct responses;Число прямых откликов;Number;Yes;Yes;256;;
;G2;Название компании;Company name;Название компании;Text;No;Yes;256;;
;G3;Должность;Position;Должность;Text;No;No;60;;
;G4;Город;City;Город;Texe;Yes;No;60;;
;G5;Область;Region;Область;Text;No;No;60;;
;G6;Регион;Region;Регион;Texe;Yes;No;60;;
;G7;Уровень дохода, от;The level of income, from;Уровень дохода, от;Number;No;No;60;;
;G8;Уровень дохода, до;Income level, up to;Уровень дохода, до;Number;Yes;No;60;;
;G9;Возраст, от;Age, from;Возраст, от;Number;No;No;60;;
;G10;Возраст, до;Age, up to;Возраст, до;Number;Yes;No;60;;
;G11;Уровень образования;The level of education;Уровень образования;Text;No;No;60;;
;G12;Семейное положение;Family status;Семейное положение;Texe;Yes;No;60;;
;G13;Пол;Floor;Пол;Text;No;No;60;;
;G14;Наличие детей;Children;Наличие детей;Texe;Yes;No;60;;
;G15;Соискатели из других городов;Competitors from other cities;Соискатели из других городов;Text;No;No;60;;
;G16;Владение английским языком;Knowledge of English;Владение английским языком;Texe;Yes;No;60;;
;G17;Источник вакансии;Job Source;Источник вакансии;Text;No;No;60;;
;G18;Тип работы;Kind of work;Тип работы;Texe;Yes;No;60;;
;G19;Права категории A;Category A rights;Права категории A;Text;No;No;60;;
;G20;Права категории B;Category B rights;Права категории B;Texe;Yes;No;60;;
;G21;Права категории C;Category C rights;Права категории C;Text;No;No;60;;
;G22;Права категории D;Category D Rights;Права категории D;Texe;Yes;No;60;;
;G23;Права категории E;Rights of category E;Права категории E;Text;No;No;60;;
;G24;Численность сотрудников в компании;Number of employees in the company;Number;Texe;Yes;No;60;;
;G25;Отрасль клиента;Customer sector;Отрасль клиента;Text;No;No;60;;
;G26;Опыт работы;experience;Number;Texe;Yes;No;60;;
;G27;Требования к квалификации;Qualification requirements;Требования к квалификации;Text;No;No;60;;
```

Рисунок 3.17 – Описание структуры данных в формате CSV

Несмотря на то, что парсинг CSV файлов осуществим с использованием стандартных библиотек, таких как библиотеки ввода-вывода, включая NIO2, а

также средства разбивки строк на лексемы и стандартные преобразования типов и использованием классов-оберток, по мнению автора, наиболее приемлемым способом является использование сторонних библиотек. В рассматриваемой задаче наиболее популярной библиотекой является OpenCSV [44]. Приведем пример кода, анализирующего алгоритм проверки набора на корректность данных. На рисунке 3.18 приведен фрагмент кода приложения. Сначала на основе файлового буферизированного потока создается экземпляр класса CsvToBean, представляющего собой реализацию отображения читаемых данных из CSV файла на объекты класса Vacancy. Для этого используется экземпляр класса-построителя CsvToBeanBuilder, который, используя потоковые методы, проводит дополнительные настройки, такие как выбор разделителя, указание удалять конечные пробелы и т.п.

```
public class VacancyReader {
    static final String FILE_NAME = "data.csv";

    public static void main(String[] args) {
            try (   Reader reader =
Files.newBufferedReader(Paths.get(FILE_NAME))   ) {
                CsvToBean<Vacancy> csvToBean =
                        new CsvToBeanBuilder<Vacancy> (reader).
                            withType(Vacancy.class).
                            withSeparator(';').
                            withIgnoreLeadingWhiteSpace(true).
                            build();
                Iterator<Vacancy> it = csvToBean.iterator();
                while (it.hasNext()) {
                    Vacancy vacancy = it.next();
                    System.out.println(vacancy);
                }
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
    }
}
```

Рисунок 3.18 – Фрагмент кода приложения для чтения CSV файла и отображения данных в объекты Java

После инициализации потока инициализируется итератор, который проходит по всему набору. При этом каждая строчка CSV файла отображается в соответствии с заглавиями столбцов (первая строка) в экземпляры класса Vacancy (рис. 3.19).

```
public class Vacancy {
    //номер вакансии
    @CsvBindByName (column = "GA", required = true)
    @CsvNumber("##0")
    private int number;

    //город
    @CsvBindByName (column = "G4", required = true)
    @Size(max = 60, message = "Длина имени города не может быть
больше 60")

    private String city;

    //Уровень дохода от
    @CsvBindByName (column = "G7", required = true)
    @CsvNumber("####0")
    private int incomeLevelMin;

    //Уровень дохода до
    @CsvBindByName (column = "G8", required = true)
    @CsvNumber("####0")
    private int incomeLevelMax;
```

Рисунок 3.19 – Класс-прототип CSV данных набора

Класс Vacancy представляет собой компонент Java, поля которого аннотированы в соответствии с библиотекой OpenCSV и стандартной технологией Bean Validation. Основу отображения создают аннотации @CsvBindByName, в которых можно указать данные какого столбца связываются с конкретным полем, а также указать обязательность поля, правила форматирования CSV для отображения и т.д.

Результат запуска приложения, приведенный на рисунке 3.20, позволяет сделать следующие выводы:

- первые 60 строк являются корректными;

- на 61-й строке произошла ошибка отображения в объект класса Vacancy.

Для дальнейшего анализа необходимо просканировать вест набор данных и вывести отчет об ошибках. Для этой цели было разработано дополнительное приложение, код которого представлен на рисунке 3.21. Аналогично ранее приведенному примеру, на основе фалового потока и парсера CSV создается входной поток для чтения CSV в формате массива строк. Обращаясь к конкретному элементу массива (переменная rec) можно получить строковое представление данных конкретного столбца. Зная требуемую структуру данных можно проконтролировать ошибки и вывести их описание для дальнейшего анализа.

```
number: 1| city :Москва| LevelMin :27000| LevelMax :30000
number: 2| city :Москва| LevelMin :21000| LevelMax :23000
number: 3| city :Москва| LevelMin :27000| LevelMax :30000
number: 4| city :Москва| LevelMin :27000| LevelMax :30000
number: 5| city :Москва| LevelMin :28000| LevelMax :31000
number: 6| city :Москва| LevelMin :27000| LevelMax :30000
number: 7| city :Москва| LevelMin :27000| LevelMax :30000
number: 8| city :Москва| LevelMin :27000| LevelMax :30000
number: 9| city :Москва| LevelMin :28000| LevelMax :31000
…
number: 58| city :Москва| LevelMin :30000| LevelMax :34000
number: 59| city :Москва| LevelMin :29000| LevelMax :32000
number: 60| city :Москва| LevelMin :28000| LevelMax :31000
Exception in thread "main" java.lang.RuntimeException: com.opencsv.exceptions.CsvRequiredFieldEmptyException: Field 'number' is mandatory but no value was provided.
        at com.opencsv.bean.concurrent.ProcessCsvLine.run(ProcessCsvLine.java:101)
        at com.opencsv.bean.CsvToBean$CsvToBeanIterator.readLineWithPossibleError(CsvToBean.java:551)
        at com.opencsv.bean.CsvToBean$CsvToBeanIterator.readSingleLine(CsvToBean.java:571)
        at com.opencsv.bean.CsvToBean$CsvToBeanIterator.next(CsvToBean.java:591)
        at edu.bigdata.VacancyReader.main(VacancyReader.java:25)
Caused by: com.opencsv.exceptions.CsvRequiredFieldEmptyException: Field 'number' is mandatory but no value was provided.
        at com.opencsv.bean.AbstractBeanField.setFieldValue(AbstractBeanField.java:163)
        at com.opencsv.bean.AbstractMappingStrategy.setFieldValue(AbstractMappingStrategy.java:449)
        at com.opencsv.bean.AbstractMappingStrategy.populateNewBean(AbstractMappingStrategy.java:317)
        at com.opencsv.bean.concurrent.ProcessCsvLine.processLine(ProcessCsvLine.java:132)
        at com.opencsv.bean.concurrent.ProcessCsvLine.run(ProcessCsvLine.java:85)
        ... 4 more
```

Рисунок 3.20 – Результат преобразования набора данных

Результат работы приложения, выводящего отчет об ошибках в CSV файле приведен на рисунке 3.22, из которого следует вывод, что с 61 строки идут строки, в которых отсутствуют данные (рис. 3.23). Таким образом, необходима ручная или автоматическая корректировка набора.

```
try (Reader in = Files.newBufferedReader(Paths.get(FILE_NAME))) {
    CSVParser csvParser =
            new CSVParserBuilder().
                withSeparator(';').
                withIgnoreLeadingWhiteSpace(true).
                build();
    CSVReader reader =
            new CSVReaderBuilder(in).
                withCSVParser(csvParser).
                withSkipLines(1). //pass headers line
                build() ;

    int linesReaded = 1;
    int vacancyNum;
    String vacancyCity;
    String[] rec = reader.readNext();
    while (rec != null) {
        linesReaded++;
        try {
            vacancyNum = Integer.parseInt(rec[0]);
            System.out.println("line " +linesReaded +
                        "; vacancyNum " + vacancyNum);
        } catch (NumberFormatException e) {
            System.out.println("line " +linesReaded +
                            ": vacancyNum err");
        }
        vacancyCity = rec[5];
        if ((vacancyCity == null) |
            (vacancyCity.isEmpty()) |
            (vacancyCity.length() > 60)) {
                System.out.println("line " +linesReaded +
                        "; vacancyCity err");
        } else {
            System.out.println("line " +linesReaded +
                        "; vacancyCity " + vacancyCity);
        }
        rec = reader.readNext();
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

Рисунок 3.21 – Фрагмент кода для логирования ошибок CSV файла

```
line 2; vacancyNum 1
line 2; vacancyCity Москва
line 3; vacancyNum 2
line 3; vacancyCity Москва
…
line 60; vacancyNum 59
line 60; vacancyCity Москва
line 61; vacancyNum 60
line 61; vacancyCity Москва
line 62: vacancyNum err
line 62; vacancyCity err
…
line 314: vacancyNum err
line 314; vacancyCity err
line 315: vacancyNum err
line 315; vacancyCity err
```

Рисунок 3.22 – Отчет об ошибках в CSV наборе

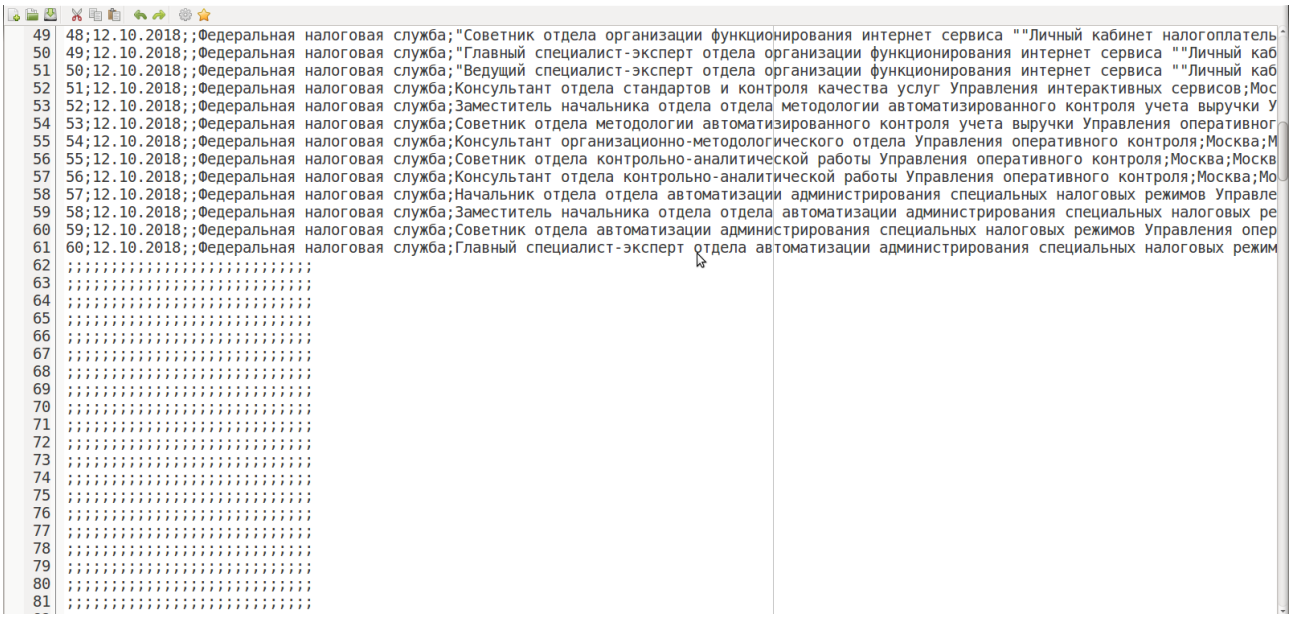


Рисунок 3.23 – Некорректные данные в наборе

Код рассмотренных файлов приведен в приложении Д.

Необходимо отметить, что очистка и консолидация данных может выполняться и тогда, когда данные загружены в распределенную файловую систему. Например, указанные процессы могут выполняться с использованием команд Spark. Однако, данные технологии выходят за рамки исследования.

3.4 Загрузка данных в систему хранения и анализа больших данных

Загрузка данных в распределённую файловую систему или базу данных NoSQL, также, может быть осуществлена различными способами, например:

- с использованием утилит, например hdfs, hdfs shell [30];
- с использованием GUI интерфейса системы хранения и анализа больших данных, например Ambari;
- с использованием специально разработанных приложений;
- с использованием Yarn, Spark.

В ходе исследования были исследованы первые два способа загрузки данных.

Загрузка данных с использованием команды hdfs аналогична работе с файлами в операционной системе класса Unix. Создание каталога загрузки в hdfs выполняется командой

```
hdfs dfs -mkdir /usr/data,
```

которая создает в распределенной файловой системе каталог data внутри каталога usr.

Копирование файла data.csv с локального хоста в распределенную файловую систему Hadoop осуществляется командой

```
hdfs dfs -put data.csv /usr/data.
```

В дополнение можно использовать следующие команды hdfs:

- ls – чтение содержимого каталога;
- rm – удаление файла;
- du -h – вывод размера файла;
- cmod – смена атрибутов доступа;
- cat – чтение содержимого файла;
- mv – перемещение файла на новое место.

Пример выполнения загрузки файла в систему хранения и анализа больших данных представлен на рис. 3.24.

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -mkdir data
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls
Found 1 items
drwxr-xr-x   - maria_dev hdfs          0 2019-06-13 06:35 data
[maria_dev@sandbox-hdp ~]$ hdfs dfs -put "/home/user/data-20181205-
structure-20181205.csv" data
```

Рисунок 3.24 – Загрузка файла в систему хранения и анализа больших данных утилитой hdfs

Второй способ загрузки данных – использование вэб-интерфейса системы Ambari сборки Hortonworks системы хранения и анализа больших данных. После входа в систему (рис. 3.25) возможно осуществить навигацию по HDFS и выбрать требуемый каталог для загрузки файла (рис. 3.26).

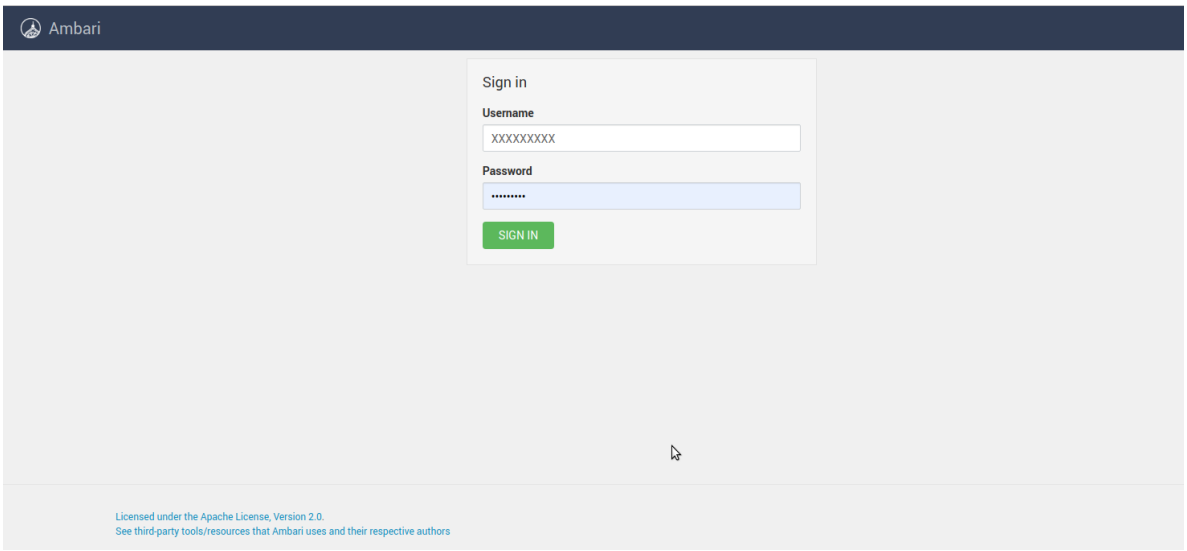


Рисунок 3.25 – Вход в систему хранения и анализа данных от Hortinworks

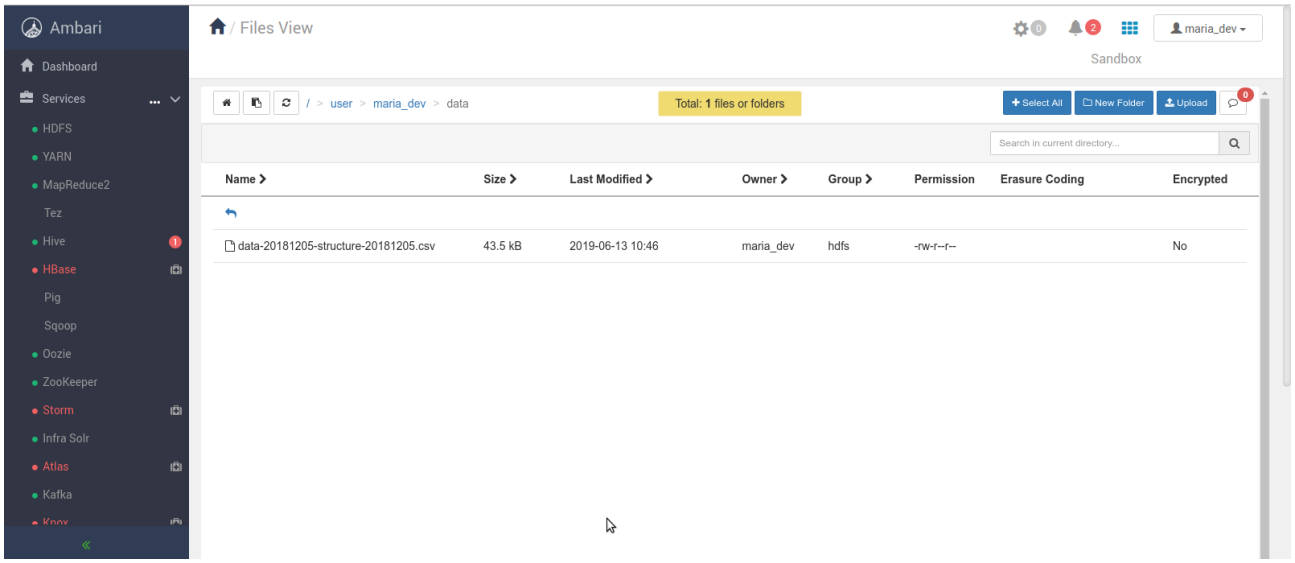


Рисунок 3.26 – Навигация по HDFS и загрузка файла

После загрузки файла доступно изменение прав доступа к файлам и каталогам. Например, на рисунке 3.27 показано изменение атрибутов каталога, для возможности использования загруженных файлов в системе Hive.

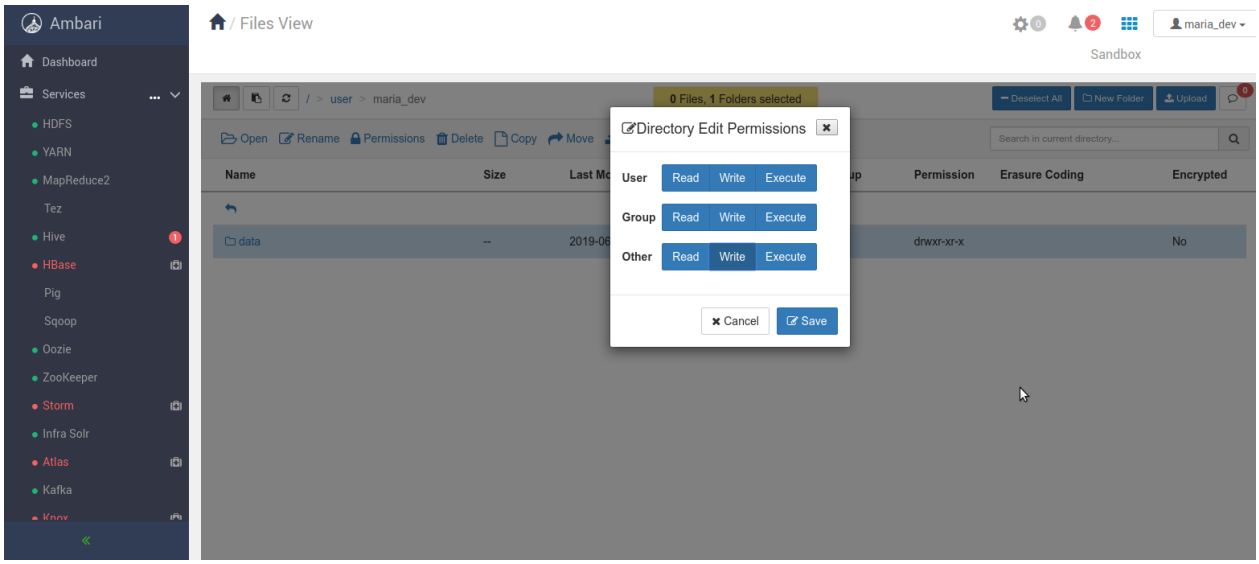


Рисунок 3.27 – Изменение атрибутов доступа к каталогу для возможности обработки в Hive

Выводы по третьему разделу

- проанализированы способы доступа к внешним данным, предоставляемыми интернет ресурсами. Апробированы технологии

веб-сканирования, веб-скрепинга и доступа к внешним API интернет ресурсов, используя стандартные и сторонние библиотеки Java;

- показано использование стандартных библиотек Java и открытой библиотеки OpenCSV для очистки данных средствами языка Java. Отмечено, что даже при использовании автоматически сгенерированных данных может понадобиться очистка;
- апробированы способы загрузки данных в систему хранения данных. В качестве реализации использована сборка от Hortonworks. Наиболее простыми способами загрузки являются использование веб-интерфейса системы Ambari или использование команд hdfs для загрузки данных с локальной системы в HDFS Hadoop;

Таким образом, предложена и апробирована технология создания системы хранения и анализа больших данных, а также апробированы и описаны решения по выполнению первых этапов процесса Data Science: получение, очистка и загрузка данных. Результаты проведенного исследования позволяют в дальнейшем продвинуться к выполнению дальнейших шагов по обработке больших данных.

ЗАКЛЮЧЕНИЕ

Целью магистерской диссертации является разработка практических рекомендаций по построению систем хранения и анализа больших данных, а также по выполнению первых этапов процесса Data Science: доступа, очистки и загрузки.

Выполненные в работе научные исследования представлены следующими основными результатами:

1. Проведен анализ архитектур построения систем хранения и анализа больших данных. Показано, что на начальных этапах целесообразно использовать дистрибутивы для развертывания рассматриваемых систем с учетом пакетной или потоковой обработки данных.
2. Произведен обзор применимости четырех видов NoSQL баз данных. Рассмотрены области их применимости.
3. С использованием стандартных и сторонних Java библиотек апробированы методы веб-сканирования и веб-скрепинга, как дополнительные методы получения внешних данных.
4. Апробирован доступ к российским Интернет источникам открытых данных с использованием JSON и CSV технологий. Разработаны алгоритмы получения информации об открытых наборах данных, скачивания наборов и их очистки.
5. Апробированы два способа загрузки данных в распределенную файловую систему Hadoop системы хранения и анализа больших данных, развернутой на основе Hortonworks HDP.

Исследованный комплекс технологий, алгоритмов и их реализаций на языке Java показал свою работоспособность и возможность его применения при дальнейших исследованиях в области больших данных.

Таким образом, в представленной магистерской диссертации решена актуальная научно-практическая проблема построения системы хранения и

анализа больших данных, а также решены практические вопросы работы по технологии больших данных на первых этапах процесса Data Science.

Результаты проведенного исследования имеют значимый практический интерес и могут быть использованы при проведении научных и прикладных исследований в области больших данных в Центре IT Student, в частности, и в Тольяттинском государственном опорном университете в целом.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Нормативно-правовые акты

1. Паспорт национальной программы «Цифровая экономика Российской Федерации» [Электронный ресурс] – Режим доступа: http://static.government.ru/media/files/urKHm0gTPPnzJlaKw3M5cNLo6gczMkPF.pdf (дата обращения 27.06.2019).

2. Информационные материалы о национальной программе «Цифровая экономика Российской Федерации» [Электронный ресурс] – Режим доступа: http://static.government.ru/media/files/3b1AsVA1v3VziZip5VzAY8RTcLEbdCct.pdf (дата обращения 27.06.2019).

Научная и методическая литература

3. Кукарцев, В. В. Теория баз данных [Электронный ресурс] : учебник / В. В. Кукарцев, Р. Ю. Царев, О. А. Антамошкин. — Электрон. текстовые данные. — Красноярск : Сибирский федеральный университет, 2017. — 180 с. — 978-5-7638-3621-9. — Режим доступа: http://www.iprbookshop.ru/84153.html

4. Киселева Т. В. Программная инженерия. Часть 1 [Электронный ресурс] : учебное пособие / Т. В. Киселева. — Ставрополь : Северо-Кавказский федеральный университет, 2017. — 137 с. — Режим доступа: http://www.iprbookshop.ru/69425.html (дата обращения 15.03.2019 г.).

Электронные ресурсы

5. Accenture. Россия. Инновации в действии. [Электронный ресурс] – Режим доступа: https://www.accenture.com/ru-ru (дата обращения 27.05.2019 г.).

6. Amazon DynamoDB [Электронный ресурс] – Режим доступа: https://aws.amazon.com/ru/dynamodb/ (дата обращения 27.05.2019 г.).

7. Analytics Comes of Age. [Электронный ресурс] – Режим доступа: https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Analytics/Our%20Insights/Analytics%20comes%20of%20age/Analytics-comes-of-age.ashx (дата обращения 27.05.2019 г.).

8. Apache Cassandra. [Электронный ресурс] – Режим доступа: http://cassandra.apache.org/ (дата обращения 27.05.2019 г.).

9. Apache CouchDB. [Электронный ресурс] – Режим доступа: http://couchdb.apache.org/ (дата обращения 27.05.2019 г.).

10. Apache HBase – Apache HBase™ Home. [Электронный ресурс] – Режим доступа: https://hbase.apache.org/ (дата обращения 27.05.2019 г.).

11. ArangoDB: Multi-model highly available NoSQL database. [Электронный ресурс] – Режим доступа: https://www.arangodb.com/ (дата обращения 27.05.2019 г.).

12. Azure Cosmos DB. Мультимодельная, глобально распределенная служба базы данных для любого масштаба [Электронный ресурс] – Режим доступа: https://azure.microsoft.com/ru-ru/services/cosmos-db/ (дата обращения 27.05.2019 г.).

13. Biehn Neil. The Missing V's in Big Data: Viability and Value / Neil Biehn. [Электронный ресурс] – Режим доступа: https://www.wired.com/insights/2013/05/the-missing-vs-in-big-data-viability-and-value/ (дата обращения 27.05.2019 г.).

14. Big Data Analytics Landscape 2019. [Электронный ресурс] – Режим доступа: https://www.learnbigdatatools.com/big-data-analytics-landscape-2019 (дата обращения 27.05.2019 г.).

15. Big Data Executive Survey 2017. Executive Summary of Findings. [Электронный ресурс] – Режим доступа: http://newvantage.com/wp-content/uploads/2017/01/Big-Data-Executive-Survey-2017-Executive-Summary.pdf (дата обращения 27.05.2019 г.).

16. Bigtable - Scalable NoSQL Database Service. [Электронный ресурс] – Режим доступа: https://cloud.google.com/bigtable/ (дата обращения 27.05.2019 г.).

17. ClickHouse. [Электронный ресурс] – Режим доступа: https://github.com/yandex/ClickHouse/releases (дата обращения 27.05.2019 г.).

18. Cloudera CDH. [Электронный ресурс] – Режим доступа: https://www.cloudera.com/products/open-source/apache-hadoop/key-cdh-components.html (дата обращения 27.05.2019 г.).

19. Columbus L. 10 Charts That Will Change Your Perspective Of Big Data's Growth / Louis Columbus. [Электронный ресурс] – Режим доступа: https://www.forbes.com/sites/louiscolumbus/2018/05/23/10-charts-that-will-change-your-perspective-of-big-datas-growth/#432b11a29268 (дата обращения 27.05.2019 г.).

20. Data sets – Data.gov [Электронный ресурс] – Режим доступа: http://catalog.data.gov/dataset (дата обращения 27.05.2019 г.).

21. Data.gov.ru | открытые данные России [Электронный ресурс] – Режим доступа: https://data.gov.ru/ (дата обращения 27.05.2019 г.).

22. DB-Engines Ranking [Электронный ресурс] – Режим доступа: https://db-engines.com/en/ranking (дата обращения 27.05.2019 г.).

23. Elasticity (cloud computing). From Wikipedia, the free encyclopedia. [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Elasticity_(cloud_computing) (дата обращения 27.05.2019 г.).

24. European Data Portal [Электронный ресурс] – Режим доступа: https://www.europeandataportal.eu/ (дата обращения 27.05.2019 г.).

25. Generate data. [Электронный ресурс] – Режим доступа: http://www.generatedata.com/ (дата обращения 27.05.2019 г.).

26. Get Started with Hortonworks Sandbox [Электронный ресурс] – Режим доступа: https://www.cloudera.com/downloads/hortonworks-sandbox.html#install (дата обращения 27.05.2019 г.).

27. Global Market Research Reports Company. Statistic MRC. [Электронный ресурс] – Режим доступа: https://www.strategymrc.com (дата обращения 27.05.2019 г.).

28. Greenfield D. How Technology and People Are Steering Manufacturing's Future David Greenfield / David Greenfield [Электронный ресурс] – Режим

доступа: https://www.automationworld.com/how-technology-and-people-are-steering-manufacturings-future (дата обращения 27.05.2019 г.).

29. Hadoop. [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Hadoop (дата обращения 27.05.2019 г.).

30. HDFS shell [Электронный ресурс] – Режим доступа: https://github.com/avast/hdfs-shell (дата обращения 27.05.2019 г.).

31. HDInsight. Простая экономичная служба аналитики корпоративного уровня с открытым кодом. [Электронный ресурс] – Режим доступа: https://azure.microsoft.com/ru-ru/services/hdinsight/ (дата обращения 27.05.2019 г.).

32. Home – Wikibon research. [Электронный ресурс] – Режим доступа: https://wikibon.com/ (дата обращения 27.05.2019 г.).

33. Hypertable [Электронный ресурс] – Режим доступа: http://www.hypertable.org/ (дата обращения 27.05.2019 г.).

34. IMF DATA. [Электронный ресурс] – Режим доступа: https://www.imf.org/en/Data (дата обращения 27.05.2019 г.).

35. Lambda architecture. From Wikipedia, the free encyclopedia. [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Lambda_architecture (дата обращения 27.05.2019 г.).

36. MapR Converged Data Platform. [Электронный ресурс] – Режим доступа: https://mapr.com/datasheets/mapr-converged-data-platform/ (дата обращения 27.05.2019 г.).

37. Marr Bernard. Big Data And AI: 30 Amazing (And Free) Public Data Sources For. / Bernard Marr 2018 [Электронный ресурс] – Режим доступа: https://www.forbes.com/sites/bernardmarr/2018/02/26/big-data-and-ai-30-amazing-and-free-public-data-sources-for-2018/#57d218e05f8a (дата обращения 27.05.2019 г.).

38. McNulty Eileen. Understanding Big Data: The Seven V’s / Eileen McNulty. [Электронный ресурс] – Режим доступа: http://dataconomy.com/2014/05/seven-vs-big-data/ (дата обращения 27.05.2019 г.).

39. Memcached - a distributed memory object caching system. [Электронный ресурс] – Режим доступа: https://memcached.org/ (дата обращения 27.05.2019 г.).

40. Min Chen, Shiwen Mao, Yin Zhang, Victor C.M. Leung. Big Data. Related Technologies, Challenges, and Future Prospects. – Spinger, 2014. – [Электронный ресурс] – Режим доступа: http://lab.ilkom.unila.ac.id/ebook/e-books%20Big%20Data/2014%20Big%20Data%20Related%20Technologies,%20Challenges%20and%20Future%20Prospects.pdf (дата обращения 27.05.2019 г.).

41. MongoDB: The most popular database for modern apps. [Электронный ресурс] – Режим доступа: https://www.mongodb.com/ (дата обращения 27.05.2019 г.).

42. Neo4j Graph Platform – The Leader in Graph Databases. [Электронный ресурс] – Режим доступа: https://neo4j.com/ (дата обращения 27.05.2019 г.).

43. Open Source Search & Analytics Elasticsearch. [Электронный ресурс] – Режим доступа: https://www.elastic.co/ (дата обращения 27.05.2019 г.).

44. OpenCSV. [Электронный ресурс] – Режим доступа: http://opencsv.sourceforge.net (дата обращения 27.05.2019 г.).

45. OpenLink Virtuoso Universal Server. [Электронный ресурс] – Режим доступа: https://virtuoso.openlinksw.com/ (дата обращения 27.05.2019 г.).

46. OrientDB: Graph Database | Multi-Model Database. [Электронный ресурс] – Режим доступа: https://orientdb.com/ (дата обращения 27.05.2019 г.).

47. Reddis. [Электронный ресурс] – Режим доступа: https://redis.io/ (дата обращения 27.05.2019 г.).

48. Tech pundits' tenuous but intriguing prognostications about 2016 and beyond. The Economist. [Электронный ресурс] – Режим доступа: https://www.economist.com/business/2015/12/31/tech-pundits-tenuous-but-intriguing-prognostications-about-2016-and-beyond (дата обращения 11.05.2019 г.).

49. Third platform. From Wikipedia, the free encyclopedia. [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Third_platform (дата обращения 27.05.2019 г.).

50. Vaseekaran Gowthamy. Big Data Battle: Batch Processing vs Stream Processing / Gowthamy Vaseekaran [Электронный ресурс] – Режим доступа: https://medium.com/@gowthamy/big-data-battle-batch-processing-vs-stream-processing-5d94600d8103 (дата обращения 27.05.2019 г.).

51. World Bank Open Data. Free and open access to global development data [Электронный ресурс] – Режим доступа: https://data.worldbank.org/ (дата обращения 27.05.2019 г.).

52. Большие данные – Википедиа. [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5 (дата обращения 27.05.2019 г.).

53. Большие данные (Big Data). [Электронный ресурс] – Режим доступа: http://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_(Big_Data) (дата обращения 27.05.2019 г.).

54. Бородаенко В., Ермаков А. Универсальная платформа обработки больших данных / Виктор Бородаенко, Александр Ермаков // «Открытые системы. СУБД» 2017, № 03 [Электронный ресурс] – Режим доступа: https://www.osp.ru/os/2017/03/13052699/ (дата обращения 27.05.2019 г.).

55. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс] / Б. Мейер. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 2019. — 285 с. — Режим доступа: http://www.iprbookshop.ru/79706.html (дата обращения 15.03.2019 г.).

56. Мэрия Москвы купила у сотовых операторов данные о перемещении жителей. [Электронный ресурс] – Режим доступа: https://sobesednik.ru/tehnologii/20190304-meriya-moskvy-kupila-u-sotovyh-operatorov-dannye-o-peremeshenii-zhitelej (дата обращения 27.05.2019 г.).

57. Открытые данные Росстата [Электронный ресурс] – Режим доступа: http://www.gks.ru/opendata/ (дата обращения 27.05.2019 г.).

58. Открытые данные ФНС России [Электронный ресурс] – Режим доступа: https://www.nalog.ru/opendata/ (дата обращения 27.05.2019 г.).

59. Портал открытых данных правительства Москвы [Электронный ресурс] – Режим доступа: https://data.mos.ru/ (дата обращения 27.05.2019 г.).

60. Правила и рекомендации. Открытые данные России. [Электронный ресурс] – Режим доступа: https://data.gov.ru/pravila-i-rekomendacii (дата обращения 27.05.2019 г.).

61. Сайт аналитической компании IDC [Электронный ресурс] – Режим доступа: http://idc.com (дата обращения 15.03.2019 г.).

62. Тольяттинский государственный опорный университет. Официальный сайт. [Электронный ресурс] – Режим доступа: https://www.tltsu.ru (дата обращения 27.05.2019 г.).

Литература на иностранном языке

63. Cielen D., Meysman A., Ali M. Introducing data science: big data, machine learning, and more, using Python tools. – Manning Publications Co., 2016.

64. Erik Meijer. The world according to linq. Communications of the ACM, 54(10):45–51, 2011.

65. Larman C. “Applying UML and patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development”, (New Jersey: Prentice Hall), 2004, p. 736.

66. Mehta R. Big Data Analytics with Java / R. Mehta. Packt Publishing, 2017. – 418 C.

67. Olive A. “Conceptual Modeling of Information Systems”, 2007.

68. Paul Zikopoulos, Chris Eaton, et al. Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media, 2011.

69. Sawant N., Shah H. Big Data Application Architecture //Big data Application Architecture Q & A. – Apress, Berkeley, CA, 2013. – C. 9-28.

ПРИЛОЖЕНИЕ А
КОД ВЕБ-СКРЕПИНГА

```
import java.io.IOException;
import java.util.List;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.select.Elements;

public class JsoupParser {

    //https://ru.wikipedia.org/wiki/Большие_данные
    private static String url =
"https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B
8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5";

    public static void main(String[] args) {
        try {
            Document document = Jsoup.connect(url).get();
            //get a title
            String title = document.title();
            System.out.println("Название статьи: " + title);
            //get a body
            Elements body = document.select("body");
            System.out.println(" Содержание: " + body.text());
            //get links
            Elements links = document.select("a[href]");
            //System.out.println(links.text());
            List<String> linksList = links.eachText();
            int linksNum = linksList.size();
            System.out.println("Число гиперссылок: " + linksNum +
"\n Первые пять:");
            for(int i=0; (i < 5) && (i < linksNum); i++) {
                System.out.println(linksList.get(i));
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

}
```

ПРИЛОЖЕНИЕ Б
КОД ВЕБ-СКАНЕРА

Класс CrawlerController

```
package edu.bigdata;

import java.time.LocalTime;
import java.time.format.DateTimeFormatter;

import edu.uci.ics.crawler4j.crawler.CrawlConfig;
import edu.uci.ics.crawler4j.crawler.CrawlController;
import edu.uci.ics.crawler4j.fetcher.PageFetcher;
import edu.uci.ics.crawler4j.robotstxt.RobotstxtConfig;
import edu.uci.ics.crawler4j.robotstxt.RobotstxtServer;

public class CrawlerController {
    public static final String SEED_URL = "http://www.tltsu.ru";
    static final int NUM_THREADS = 2; //number of threads used for
the crawl
    static final String CRAWL_STORAGE = "./data/crawl/root";
    static final int MAX_CRAWL_DEPTH = 3;
    static final int POLITENESS_DELAY = 500; //delay between
sending two requests to the same host
    static final int MAX_PAGES_TO_FETCH = 50;

    public static void main(String[] args) throws Exception {
        /*
         * Instantiate crawl config
         */
        CrawlConfig cfg = new CrawlConfig();
        cfg.setCrawlStorageFolder(CRAWL_STORAGE);
        cfg.setPolitenessDelay(POLITENESS_DELAY);
        cfg.setMaxDepthOfCrawling(MAX_CRAWL_DEPTH);
        cfg.setMaxPagesToFetch(MAX_PAGES_TO_FETCH);
        cfg.setIncludeBinaryContentInCrawling(false);

        /*
         * Instantiate controller for this crawl.
        */
        PageFetcher pageFetcher = new PageFetcher(cfg);
        RobotstxtConfig robotstxtConfig = new RobotstxtConfig();
        RobotstxtServer robotstxtServer = new
RobotstxtServer(robotstxtConfig, pageFetcher);
        CrawlController controller = new CrawlController(cfg,
pageFetcher, robotstxtServer);
        /*
         * Add seed URLs
         */
        controller.addSeed(SEED_URL);
```

```
        /*
         * Start the crawl.
         */
        controller.start(TltsuCrawler.class, NUM_THREADS);


System.out.println(LocalTime.now().format(DateTimeFormatter.ISO_LO
CAL_TIME) + " application stopped");
        }
    }
```

Класс TltsuCrawler

```
package edu.bigdata;

//import static edu.bigdata.CrawlerController.SEED_URL;
import edu.uci.ics.crawler4j.crawler.Page;
import edu.uci.ics.crawler4j.crawler.WebCrawler;
import edu.uci.ics.crawler4j.parser.HtmlParseData;
import edu.uci.ics.crawler4j.url.WebURL;
import java.util.Set;
import java.util.regex.*;


public class TltsuCrawler extends WebCrawler {
    private final static Pattern FILTERS =
Pattern.compile(".*(\\.(css|js|gif|jpg|png|mp3|mp4|zip|gz))$");

    /**
     * Specify whether the given url should be crawled
     */

    @Override
    public boolean shouldVisit(Page referringPage, WebURL url) {
        System.out.println("shouldVisit: " +
url.getURL().toLowerCase());

        String href = url.getURL().toLowerCase();
        boolean result = !FILTERS.matcher(href).matches(); //&&
href.startsWith(SEED_URL);
        if(result) {
            System.out.println("Visited " +
url.getURL().toLowerCase());
        } else {
            System.out.println("Not visited " +
url.getURL().toLowerCase());
        }
        return result;
    }

    /**
```

```
     * This function is called when a page is fetched and ready
     * to be processed by the program.
     */
    @Override
    public void visit(Page page) {
        String url = page.getWebURL().getURL();
        System.out.println("URL: " + url);

        if (page.getParseData() instanceof HtmlParseData) {
            HtmlParseData htmlParseData = (HtmlParseData)
page.getParseData();
            String text = htmlParseData.getText(); //extract text
from page
            String html = htmlParseData.getHtml(); //extract html
from page
            Set<WebURL> links = htmlParseData.getOutgoingUrls();

            System.out.println("----------------------------------
-----------------------");
            System.out.println("Page URL: " + url);
            System.out.println("Text length: " + text.length());
            System.out.println("Html length: " + html.length());
            System.out.println("Number of outgoing links: " +
links.size());
            System.out.println("----------------------------------
-----------------------");
        }
    }

}
```

ПРИЛОЖЕНИЕ В
КОД ПРОСМОТРА НАБОРА ДАННЫХ ПОРТАЛА ОТКРЫТЫХ ДАННЫХ РФ

```
import java.io.IOException;
import java.net.*;
import java.nio.charset.Charset;
import org.apache.commons.io.IOUtils;
import org.json.JSONArray;
import org.json.JSONObject;


/*
 * Перечень наборов открытых данных -
 * запрос предназначен для получения перечня наборов открытых
данных.
 * Структура запроса: «/api/<format>/dataset»
 */
public class Main {

    public static void main(String[] args) {
        try {
            URL url = new URL (
                "https://data.gov.ru/api/dataset/?" +
                "topic=Education&" +
                "access_token=XXX");

            JSONArray datasets = new JSONArray(
                IOUtils.toString(url, Charset.forName("UTF-8")));
            int numRec = datasets.length();
            System.out.println(numRec);
            JSONObject e;
            for (int i=0; (i < 5) && (i < numRec); i++) {
                e = (JSONObject) datasets.get(i);
                System.out.println(e.toString());
            }
        } catch (MalformedURLException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }

    }
}
```

ПРИЛОЖЕНИЕ Г
КОД ЗАГРУЗКИ НАБОРА ДАННЫХ С ПОРТАЛА ОТКРЫТЫХ ДАННЫХ РФ

```
package edu.bigdata;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.nio.channels.Channels;
import java.nio.channels.FileChannel;
import java.nio.channels.ReadableByteChannel;
import java.nio.charset.Charset;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardCopyOption;

import org.apache.commons.io.IOUtils;
import org.json.JSONArray;
import org.json.JSONObject;

public class Main {

    static final String ACCESS_TOKEN = "/?access_token=XXX";
    static final String ACCESS_URL =
                "https://data.gov.ru/api/json/dataset" +
                "/7710539135-trud" + //Трудоустройство выпускников
учреждений профессионального образования
                "/version";

    public static void main(String[] args) {
        try {
            //получаем набор версий
            URL url = new URL (ACCESS_URL + ACCESS_TOKEN);
            JSONArray versions = new JSONArray(
                IOUtils.toString(url, Charset.forName("UTF-8")));
            int numRec = versions.length();
            System.out.println(numRec);
            JSONObject lastVersion = (JSONObject) versions.get(0);
            System.out.println(lastVersion.toString());
            String created = lastVersion.getString("created");
            System.out.println(created);

            //получаем адрес последней версии набора данных
            url = new URL (ACCESS_URL +
                    "/" + created +
                    ACCESS_TOKEN);
            //url для получения информации о наборе данных
```

```
            JSONArray dataset = new JSONArray(
                IOUtils.toString(url, Charset.forName("UTF-8")));
            JSONObject data = (JSONObject) dataset.get(0);
            System.out.println(data.toString());
            String dataURI = data.getString("source");
            System.out.println(dataURI);
            //выделяем имя файла из url
            String fname = dataURI.substring(
                dataURI.lastIndexOf('/')+1, dataURI.length());
            System.out.println("file name: " + fname);

//сохраняем файл набора
//transfer bytes between 2 Channels
//without buffering them into the application memory
//read the file from our URL
            ReadableByteChannel readableByteChannel =
                Channels.newChannel(new URL(dataURI).openStream());
//The bytes read from the ReadableByteChannel
//will be transferred to a FileChannel corresponding
//to the file that will be downloaded
            try (FileOutputStream fileOutputStream = new
                    FileOutputStream(fname);
                FileChannel fileChannel =
                    fileOutputStream.getChannel()) {
//use the transferFrom() method from the ReadableByteChannel class
//to download the bytes from the given URL to our FileChannel
                fileOutputStream.getChannel()
                .transferFrom(readableByteChannel,
                                0,
                                Long.MAX_VALUE);
            }
        } catch (MalformedURLException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

ПРИЛОЖЕНИЕ Д
КОД ПРИЛОЖЕНИЙ ДЛЯ АНАЛИЗА И ЧТЕНИЯ CSV ФАЙЛОВ

Класс `Vacancy`

```
package edu.bigdata;

import javax.validation.constraints.Size;
import com.opencsv.bean.CsvBindByName;
import com.opencsv.bean.CsvNumber;

public class Vacancy {
    //номер вакансии
    @CsvBindByName (column = "GA", required = true)
    @CsvNumber("##0")
    private int number;

    //город
    @CsvBindByName (column = "G4", required = true)
    @Size(max = 60, message = "Длина имени города не может быть
больше 60")

    private String city;

    //Уровень дохода от
    @CsvBindByName (column = "G7", required = true)
    @CsvNumber("####0")
    private int incomeLevelMin;

    //Уровень дохода до
    @CsvBindByName (column = "G8", required = true)
    @CsvNumber("####0")
    private int incomeLevelMax;

    public int getNumber() {
        return number;
    }

    public void setNumber(int number) {
        this.number = number;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
```

```
        this.city = city;
    }

    public int getIncomeLevelMin() {
        return incomeLevelMin;
    }

    public void setIncomeLevelMin(int incomeLevelMin) {
        this.incomeLevelMin = incomeLevelMin;
    }

    public int getIncomeLevelMax() {
        return incomeLevelMax;
    }

    public void setIncomeLevelMax(int incomeLevelMax) {
        this.incomeLevelMax = incomeLevelMax;
    }

    @Override
    public String toString() {
        return "number: " + this.number +
                        "| city :" + this.city +
                        "| LevelMin :" + this.incomeLevelMin +
                        "| LevelMax :" + this.incomeLevelMax;
    }

}
```

Класс VacancyReader

```
package edu.bigdata;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.Reader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Iterator;
import com.opencsv.bean.CsvToBean;
import com.opencsv.bean.CsvToBeanBuilder;

public class VacancyReader {
    static final String FILE_NAME = "data.csv";

    public static void main(String[] args) {
            try (   Reader reader =
Files.newBufferedReader(Paths.get(FILE_NAME))   ) {
                CsvToBean<Vacancy> csvToBean =
                        new CsvToBeanBuilder<Vacancy> (reader).
                            withType(Vacancy.class).
                            withSeparator(';').
                            withIgnoreLeadingWhiteSpace(true).
                            build();
                Iterator<Vacancy> it = csvToBean.iterator();
                while (it.hasNext()) {
                    Vacancy vacancy = it.next();
                    System.out.println(vacancy);
                }
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
    }
}
```

Класс `ProblemReporter`

```
package edu.bigdata;

import java.io.IOException;
import java.io.Reader;
import java.nio.file.Files;
import java.nio.file.Paths;
import com.opencsv.CSVParser;
import com.opencsv.CSVParserBuilder;
import com.opencsv.CSVReader;
import com.opencsv.CSVReaderBuilder;

public class ProblemReporter {
    static final String FILE_NAME = "data.csv";

    public static void main(String[] args) {
        try (   Reader in =
Files.newBufferedReader(Paths.get(FILE_NAME))   ) {
            CSVParser csvParser =
                    new CSVParserBuilder().
                    withSeparator(';').
                    withIgnoreLeadingWhiteSpace(true).
                    build();
            CSVReader reader =
                    new CSVReaderBuilder(in).
                    withCSVParser(csvParser).
                    withSkipLines(1). //pass headers line
                    build() ;

            int linesReaded = 1;
            int vacancyNum;
            String vacancyCity;

            String[] rec = reader.readNext();
            while (rec != null) {
                linesReaded++;
                //test vacancy number (int)
                try {
                    vacancyNum = Integer.parseInt(rec[0]);
                    System.out.println("line " +linesReaded + ";
vacancyNum " + vacancyNum);
                } catch (NumberFormatException e) {
                    System.out.println("line " +linesReaded + ":
vacancyNum err");
                }
                //test vacancy City (String[60])
                vacancyCity = rec[5];
                if ((vacancyCity == null) |
                        (vacancyCity.isEmpty()) |
                        (vacancyCity.length() > 60)) {
```

```
                    System.out.println("line " +linesReaded + "; vacancyCity err");
                } else {
                    System.out.println("line " +linesReaded + "; vacancyCity " + vacancyCity);
                }

                rec = reader.readNext();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```