[Лабораторная работа №1. Управление процессами обработки неструктурированных данных с помощью Cloudera Manager 3](#_Toc68627377)

[Цели и задачи лабораторной работы 3](#_Toc68627378)

[Сценарий лабораторной работы 3](#_Toc68627379)

[Инструмент Cloudera Manager 4](#_Toc68627380)

[Подготовка к лабораторной работе 6](#_Toc68627381)

[Запуск веб-консоли Cloudera Manager 9](#_Toc68627382)

[Работа со страницей приветствия 9](#_Toc68627383)

[Администрирование Cloudera Manager 12](#_Toc68627384)

[Запуск и остановка компонентов 12](#_Toc68627385)

[Работа с файлами 13](#_Toc68627386)

[Лабораторная работа №2. Управление большими данными при помощи Apache Hive 17](#_Toc68627387)

[Цели и задачи лабораторной работы 17](#_Toc68627388)

[Сценарий лабораторной работы 18](#_Toc68627389)

[Инструмент Apache Hive 19](#_Toc68627390)

[Импорт файла в рабочий каталог текущего пользователя 20](#_Toc68627391)

[Импорт данных формата JSON Array в хранилище Hive 22](#_Toc68627392)

[Экспорт данных из хранилища Hive в формате TSV 28](#_Toc68627393)

[Импорт данных из формата TSV в хранилище Hive 29](#_Toc68627394)

[Экспорт данных в формате JSON Array 30](#_Toc68627395)

[Лабораторная работа №3. Анализ структурированных и неструктурированных данных с использованием Apache Hue 32](#_Toc68627396)

[Цели и задачи лабораторной работы 32](#_Toc68627397)

[Сценарий лабораторной работы 32](#_Toc68627398)

[Инструмент Apache Hue 33](#_Toc68627399)

[Подготовка к лабораторной работе 34](#_Toc68627400)

[Преобразование данных 35](#_Toc68627401)

[Удаление ненужных данных 35](#_Toc68627402)

[Объединение данных 37](#_Toc68627403)

[Анализ данных 38](#_Toc68627404)

[Сортировка 38](#_Toc68627405)

[Построение диаграммы 38](#_Toc68627406)

[Очистка данных 41](#_Toc68627407)

[Выборка данных 43](#_Toc68627408)

[Выборка уникальных записей 45](#_Toc68627409)

[Выборка популярных записей 46](#_Toc68627410)

[Лабораторная работа №4. Анализ потоковой информации с использованием специализированных языков обработки потоков данных 50](#_Toc68627411)

[Цели и задачи лабораторной работы 50](#_Toc68627412)

[Сценарий лабораторной работы 50](#_Toc68627413)

[Инструменты IBM InfoSphere Streams 51](#_Toc68627414)

[Инструменты Kafka и Spark Streaming 52](#_Toc68627415)

[Работа с Kafka 54](#_Toc68627416)

[Подготовка к работе 54](#_Toc68627417)

[Kafka и Zookeeper 54](#_Toc68627418)

[Лабораторная работа №5. Исследование текстовой информации с помощью Apache Nutch и Apache Solr 59](#_Toc68627419)

[Цели и задачи лабораторной работы 59](#_Toc68627420)

[Сценарий лабораторной работы 59](#_Toc68627421)

[Инструмент Apache Nutch 60](#_Toc68627422)

[Инструмент Apache Solr 61](#_Toc68627423)

[Подготовка к лабораторной работе 63](#_Toc68627424)

[Сбор и индексация данных 64](#_Toc68627425)

[Текстовый анализ 70](#_Toc68627426)

# Лабораторная работа №1. Управление процессами обработки неструктурированных данных с помощью Cloudera Manager

## Цели и задачи лабораторной работы

Целью данной лабораторной работы является изучение процесса администрирования системы при помощи веб-консоли, что включает в себя запуск и остановку компонентов, отслеживание и анализ состояний выполняемых компонентов и некоторые другие функции.

В ходе выполнения лабораторной работы необходимо решить следующие задачи:

* запуск веб-консоли Cloudera Manager;
* работа со средствами, доступными со страницы приветствия (Welcome Page);
* администрирование системы (использование инструментов администрирования), проверка состояний кластеров, запуск и остановка компонентов;
* работа с распределенной файловой системой (работа с каталогами, создание подкаталогов, загрузка файлов в HDFS через веб-консоль Apache Hue).

## Сценарий лабораторной работы

1. Используя веб-консоль Apache Hue, загрузить текстовый документ в распределенную файловую систему HDFS.
2. Проверить состояние кластера, запустить и остановить компоненты кластера в Cloudera Manager.
3. Создать подкаталоги в HDFS и загрузить файлы в распределенную файловую систему через веб-консоль Apache Hue.

## Инструмент Cloudera Manager

Cloudera Manager — специализированный компонент Cloudera, который позволяет организовать создание и изменение Hadoop-сред, проводить анализ и отслеживать процесс выполнения команд на обработку, настраивать и получать уведомления о наступлении тех или иных событий, связанных с инфраструктурой организации. Другими словами, Cloudera Manager — это приложение для комплексного управления всеми кластерами и их компонентами, функционал которого аналогичен функционалу IBM InfoSphere BigInsights.

Главной частью Cloudera Manager является Cloudera Manager Server (Рис. 1). Cloudera Manager Server развернут на Admin Console Web Server и отвечает за установку программного обеспечения, настройку, запуск и остановку компонентов, а также за непосредственное управление кластером.

Cloudera Manager Server работает с несколькими другими компонентами:

* Агентами (Agent). Агенты установлены на каждом хосте. Агенты отвечают за запуск и остановку процессов, распаковку конфигураций, запуск установок и мониторинг хоста.
* Служба управления (Management Service). Служба управления состоит из набора ролей, которые выполняют различные функции мониторинга, оповещения, отчетности.
* База данных (DataBase). В базе данных хранится информация о конфигурации и мониторинге. Как правило, несколько баз данных работают на одном или нескольких серверах.
* Cloudera Repository. Хранилище программного обеспечения дистрибутива Cloudera Manager.
* Клиенты (Clients). Клиенты — это интерфейсы для взаимодействия с сервером.
* Консоль администратора (Admin Console). Консоль администратора — это веб-интерфейс, с помощью которого администраторы взаимодействуют с Cloudera Manager (управляют кластерами, запускают и останавливают компоненты и т.д.).
* API. С помощью API разработчики создают пользовательские приложения Cloudera Manager.

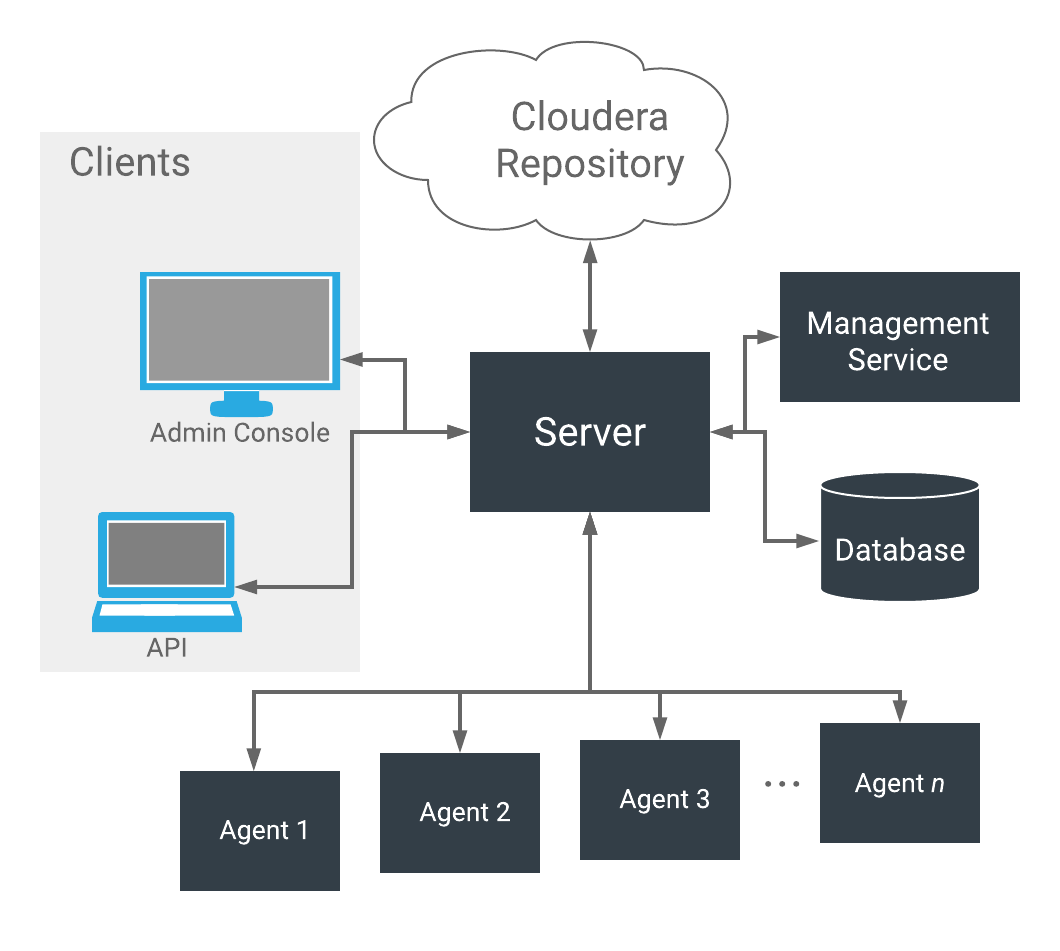


Рис. 1. Архитектура Cloudera Manager

## Подготовка к лабораторной работе

Перед началом выполнения лабораторной работы получите у преподавателя логин и пароль для входа на виртуальную машину.

Для выполнения лабораторной работы вам потребуется установить программы PuTTY и Xming. С помощью программы PuTTY вы сможете подключаться и управлять удаленным узлом через протокол SSH. Программа Xming необходима для отрисовки графических приложений удаленной машины на локальной машине.

Перед тем, как приступить к выполнению лабораторной работы, убедитесь, что:

* Вы корректно запустили Xming. Для этого посмотрите, появился ли в области уведомлений значок .
* Вы корректно запустили программу PuTTY (Рис. 2). Для этого сделайте следующее:
  1. Запустите PuTTY.
  2. Введите в поле **Host Name** *172.16.1.160* и выберите **SSH**.
  3. Откройте **Connection** **→ SSH**. Выберите **Enable compression**.
  4. Откройте **Connection** **→ SSH** **→ Cipher**. Выберите **Blowfish** и переместить его на первое место в списке. Для этого нажмите **Up**. Выберите **Arcfour (SSH-2 only)** и переместите его на второе место в списке. Для этого нажмите Up.
  5. Откройте **Connection** **→ SSH** **→ Auth → X11**. Выберите **Enable X11 forwarding**.
  6. Откройте **Connection** **→ SSH** **→ Auth → Tunnels**.Введите в поле **Source port** *5901*. Введите в поле **Destination** *127.0.0.1:5901*. Нажмите **Add**.
  7. Откройте **Session**, введите в поле **Saved Sessions** имя сессии и нажмите **Save**.
  8. Нажмите **Open**.

Чтобы загрузить сохраненную сессию, нажмите **Load**. Настройки загрузятся автоматически.

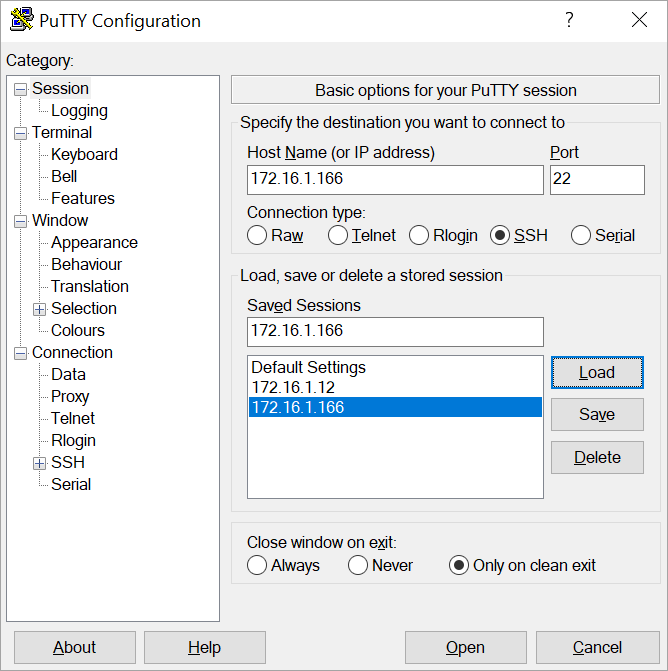


Рис. 2. Настройка PuTTY



Рис. 3. Подключение к удаленному терминалу через PuTTY

В открывшемся окне:

1. Введите логин (Рис. 3).
2. Нажмите **Enter**.
3. Введите пароль (Рис. 4).

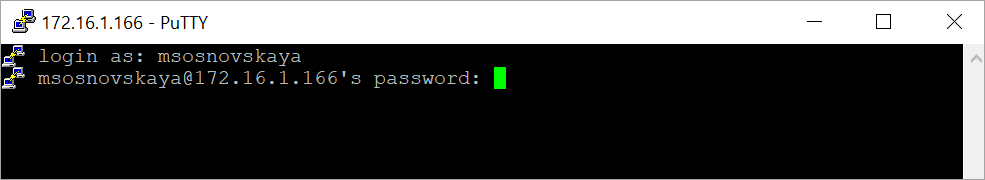


Рис. 4. Ввод логина и пароля

Запустите на удаленной машине веб-браузер, выполнив команду (Рис. 5):

firefox &

Если вы все сделали верно, у вас откроется окно браузера Firefox. Обратите внимание, что браузер запущен на удаленной машине. Окно на вашем экране отрисовывается с помощью программы Xming.

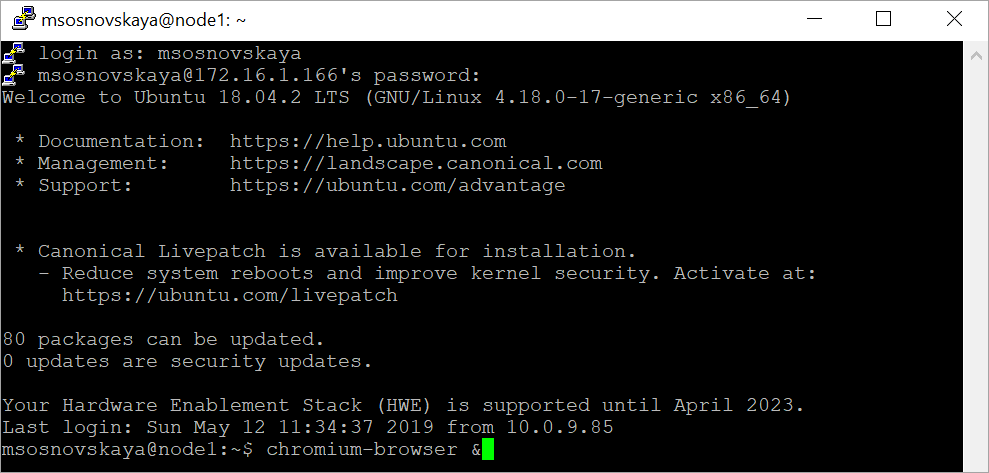


Рис. 5. Запуск веб-браузера

В большинстве лабораторных работ используется набор различных файлов с данными, вспомогательных скриптов и утилит. Для загрузки необходимых файлов требуется скачать репозиторий. Для этого введите команду в командную строку удаленной машины:

|  |
| --- |
| git **clone** https://github.com/alexcochnev/labs\_oss |

Для настройки окружения перед выполнением каждой лабораторной работы следует выполнить команды:

|  |
| --- |
| cd labs\_oss/  source envsetup.sh |

## Запуск веб-консоли Cloudera Manager

Чтобы запустить веб-консоль, введите в адресную строку веб-браузера <http://localhost:7180/>. Для входа в систему используйте логин и пароль, которые вам выдал преподаватель. У вас откроется стартовая страница Cloudera Manager (Рис. 6).

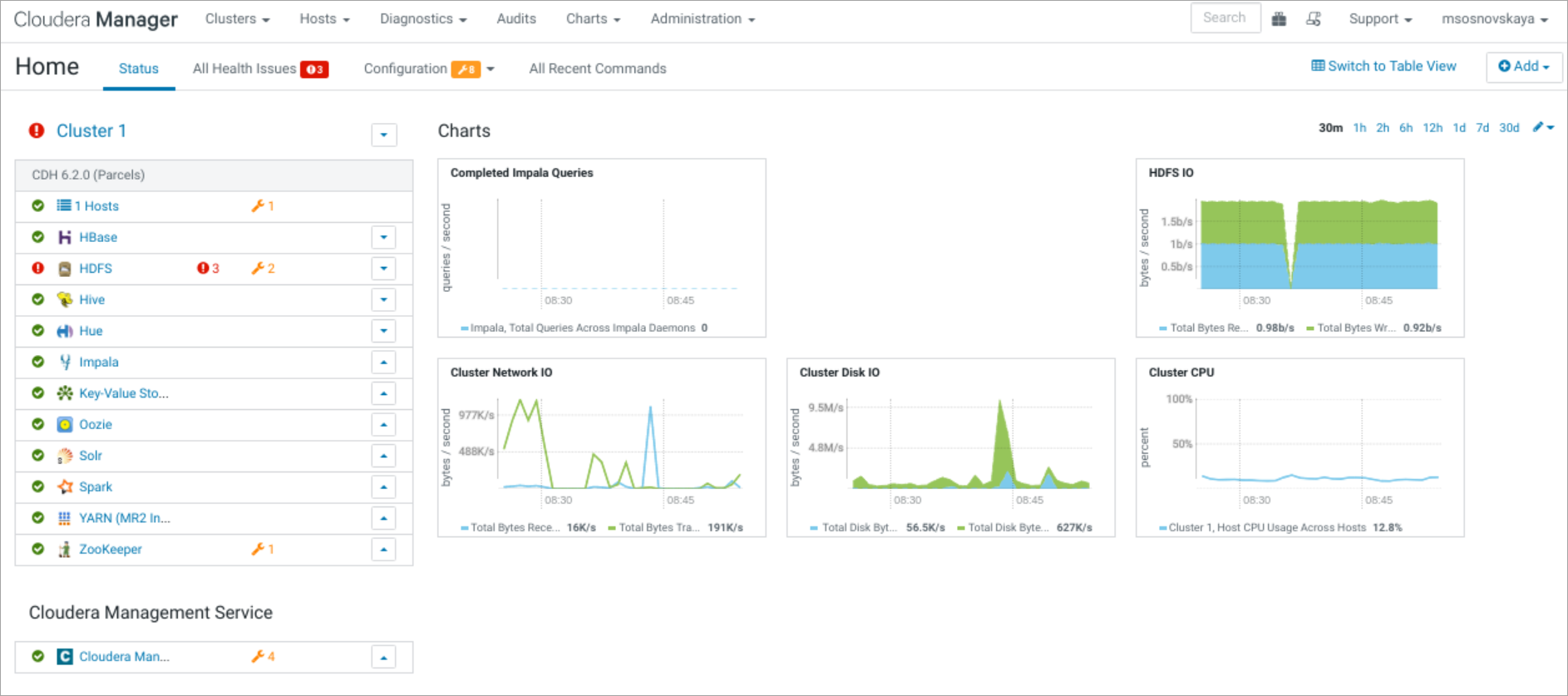


Рис. 6. Стартовая страница Cloudera Manager

## Работа со страницей приветствия

В данном разделе вы ознакомитесь с главной страницей Cloudera Manager. Главная страница содержит информацию о кластере и его компонентах и состоянии системы. Помимо этого на странице приветствия находятся ссылки на все основные ресурсы, с которыми работает администратор.

Убедитесь, что у вас активна вкладка **Status**, расположенная в верхнем левом углу консоли. Если необходимо, кликните по надписи *Cloudera Manager*, чтобы перейти на главную страницу (Рис. 7).

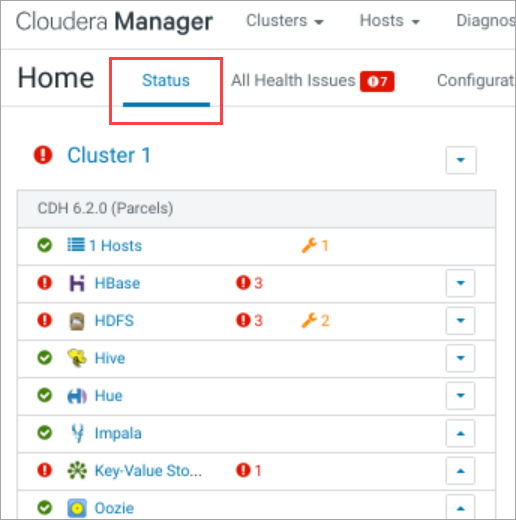


Рис. 7. Стартовая страница

Изучите панель **Cluster 1**, расположенную слева. В этом блоке отображаются все компоненты активного кластера и уведомления об ошибках в их работе. Чтобы просмотреть уведомления, нажмите на соответствующий значок рядом с названием инструмента (Рис. 8) или перейдите на вкладки **All Health Issues** и **Configuration** (Рис. 9), где собраны все ошибки и предупреждения с подробным описанием и логами. Красным обозначены критичные проблемы в работе инструмента, которые необходимо устранить, желтым — предупреждения.

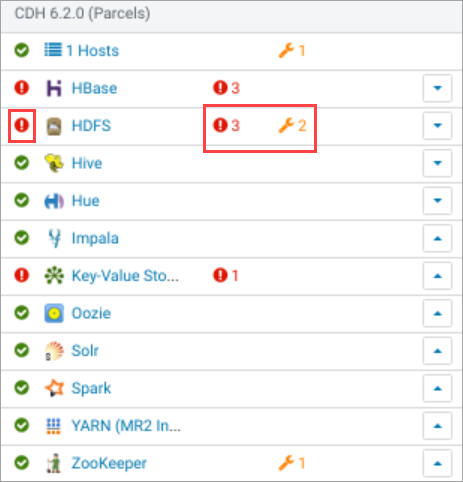


Рис. 8. Состояние служб

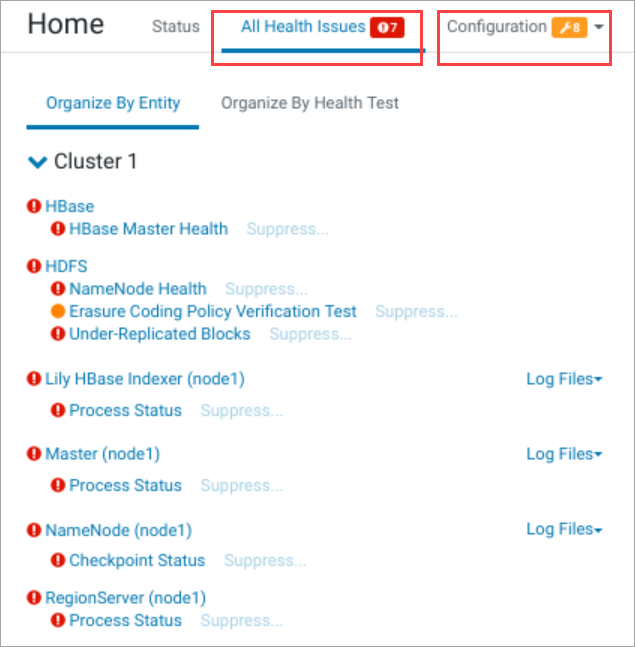


Рис. 9. Ошибки и предупреждения

В блоке **Charts** вы можете посмотреть загруженность вычислительных ядер процессора, состояние оперативной памяти (количество свободной и занятой), нагрузку на сеть, нагрузку на устройства постоянного хранения данных (устройства ввода-вывода). Вы можете выбрать, за какое время хотите получить данные. Для этого в верхнем левом углу экрана выберите количество часов или дней (Рис. 10).

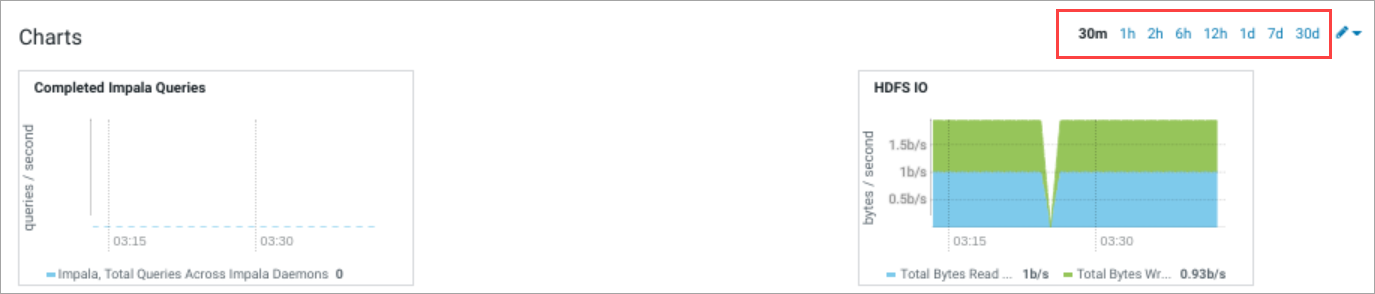


Рис. 10. Данные о системе

Чтобы посмотреть машины в кластере и их состояние, перейдите на вкладку **Hosts** и выберите **All hosts** (Рис. 11).

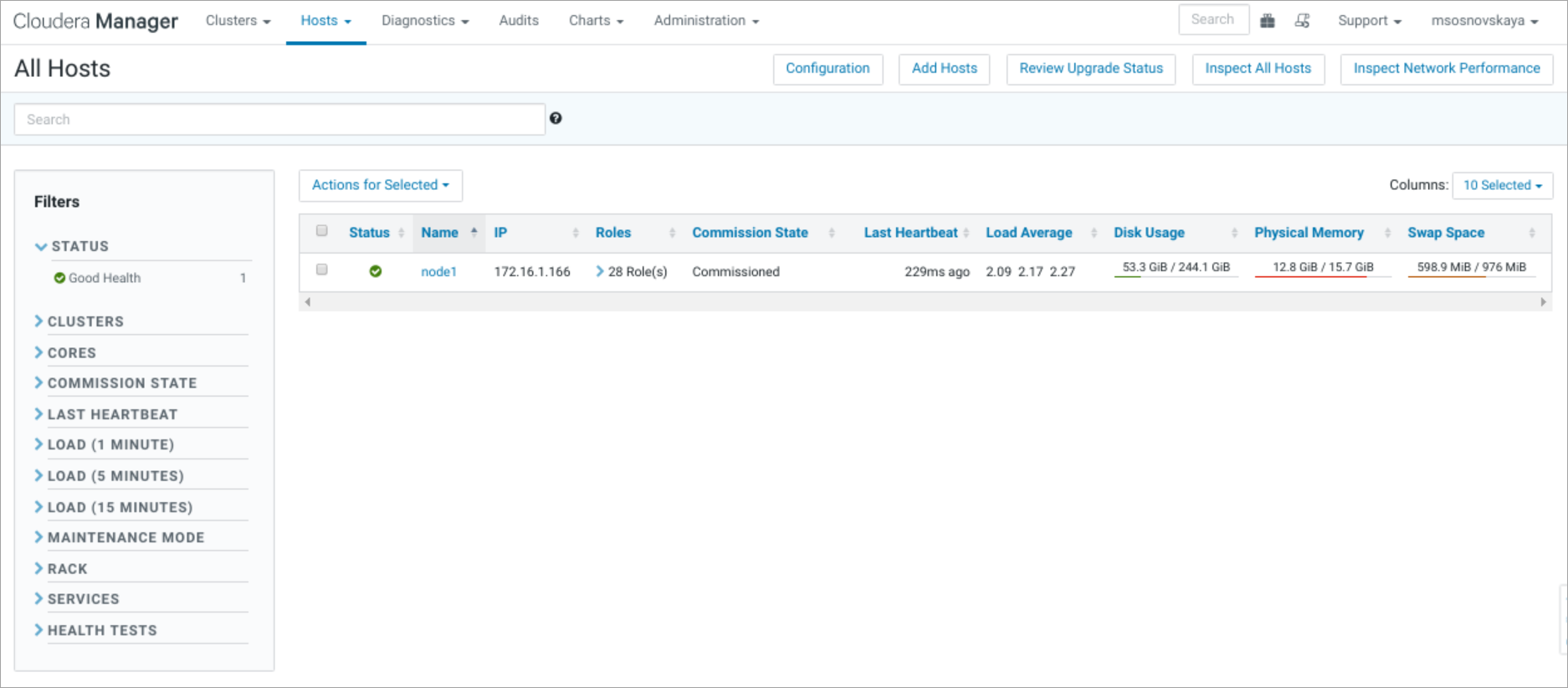


Рис. 11. Вкладка Hosts

## Администрирование Cloudera Manager

### **Запуск и остановка компонентов**

С помощью веб-консоли вы можете переключать статус компонентов.

**Внимание! Пожалуйста, изучите этот раздел, но не останавливайте компоненты системы, чтобы не помешать выполнению лабораторных работ другими участниками.**

В блоке **Cluster 1** выберите инструмент Hue, чтобы посмотреть его состояние (Рис. 12).

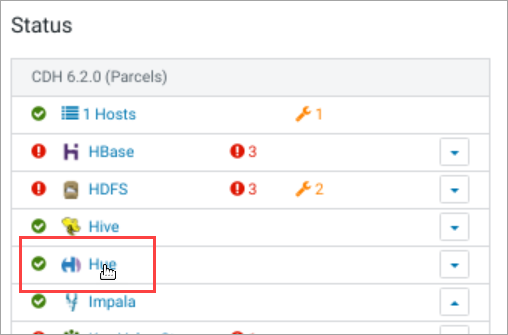


Рис. 12. Выбор службы Hue

На вкладке **Actions** вы увидите кнопки для запуска и остановки Apache Hue (Рис. 13).

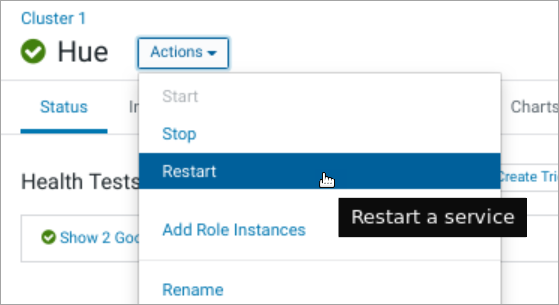


Рис. 13. Переключение статуса компонента

Чтобы посмотреть веб-интерфейс Apache Hue перейдите на вкладку **WEB UI** и выберите **Hue Load Balanced - recommended (node1)** (Рис. 14). У вас в браузере откроется новая страница.

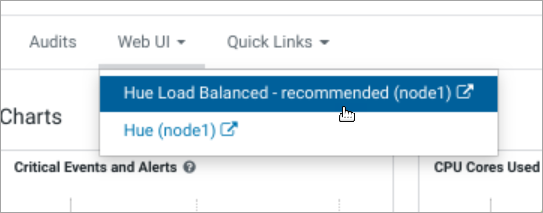


Рис. 14. Открытие веб-интерфейса

### **Работа с файлами**

В Apache Hue вы можете посмотреть содержание распределенной файловой системы, создать новые каталоги и подкаталоги и загрузить небольшие файлы. В данном разделе вы научитесь работать с распределенной файловой системой HDFS в Apache Hue.

Нажмите , чтобы перейти в распределенную файловую систему (Рис. 15).



Рис. 15. HDFS

Найдите вашу личную папку. Ее название совпадает с вашим логином. Чтобы посмотреть содержимое вашей личной папки, нажмите на нее правой кнопкой мыши и выберите **Open in Browser** (Рис. 16).

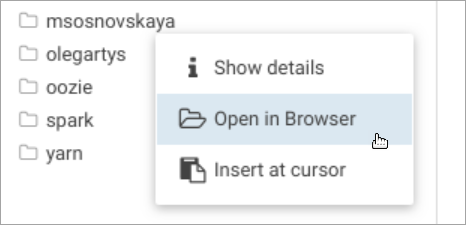


Рис. 16. Открытие папки

Чтобы создать новую директорию или новый файл, нажмите **New** **→ Directory** или **New** **→ File**, введите название и нажмите **Create** (Рис. 17).

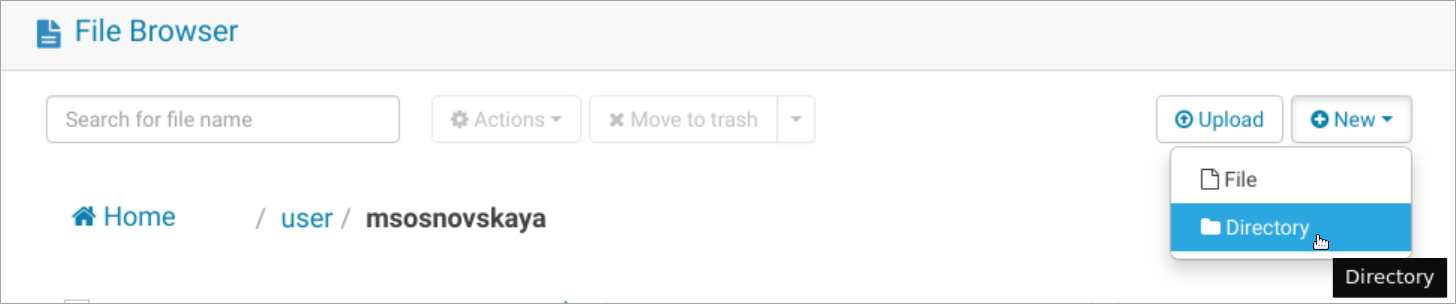


Рис. 17. Создание директории

Созданный файл или директория отобразятся на экране внизу, т.е. в содержимом вашей личной папки (Рис. 18). Обратите внимание, что система в соответствии с вашей ролью задает права доступа к файлу или директории. Например, если вы зайдете в личную папку другого студента, то увидите, что у вас будут права только на чтение его файлов.

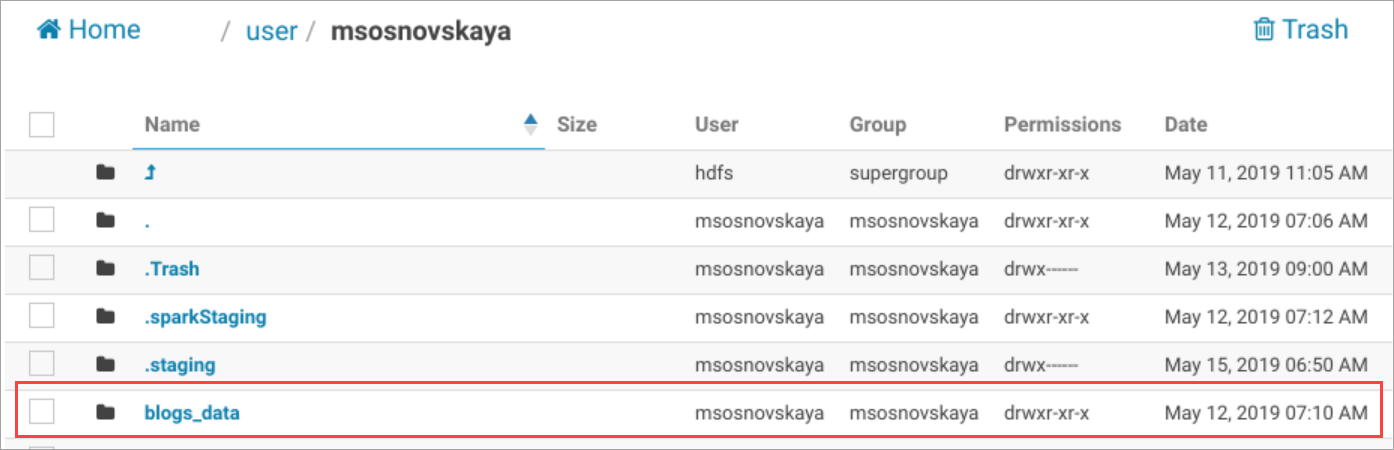


Рис. 18. Содержимое директории

Чтобы загрузить файл из вашей локальной системы, нажмите **Upload** (Рис. 19). В появившемся окне нажмите **Select files** и выберите файл. В окне загрузки должно появиться название файла и прогресс загрузки. После завершения загрузки проверьте, что выбранный вами файл появился в содержимом директории.

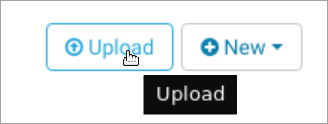


Рис. 19. Загрузка файла из локальной системы

Чтобы посмотреть, какие действия можно совершить над файлом или директорией, выберите директорию или файл (Рис. 20). Вы можете переименовать, переместить, скопировать файл или директорию, поменять владельца и отредактировать права доступа к файлу или директории, а также посмотреть краткую информацию о файле или директории и объединить несколько файлов или директорий в один архив. Для этого выберите нужное действие в выпадающем списке.

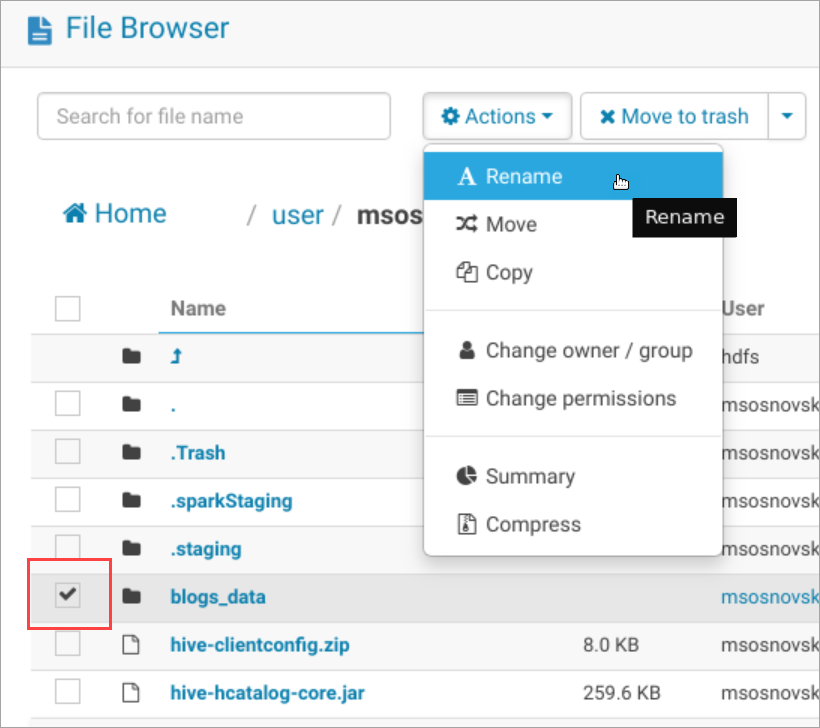


Рис. 20. Действия над файлом или директорией

Чтобы удалить файл или директорию (Рис. 21), нажмите **Move to trash**. Чтобы не помещать файл или директорию в корзину и удалить файл или директорию без возможности восстановления, нажмите **Move to trash** **→ Delete forever.**

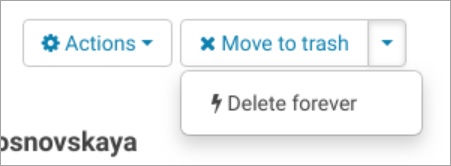


Рис. 21. Удаление файла

# 

# Лабораторная работа №2. Управление большими данными при помощи Apache Hive

## Цели и задачи лабораторной работы

Целью данной лабораторной работы является освоение основного способа работы с одним из главных компонентов экосистемы Hadoop — хранилища Hive. Получать данные из Hive можно таким же образом, как и из корпоративной базы данных. Для этого можно использовать JDBC и ODBC интерфейсы как в большинстве популярных продуктов, занимающихся хранением данных.

Все задания лабораторной работы выполняются из командной строки интерпретатора операционной системы GNU/Linux Bash. В результате выполнения лабораторной работы вы научитесь выполнять следующие задачи:

* работать с файловой HDFS из командной строки;
* подключаться к серверу хранилища Hive для работы с хранилищем;
* создавать и удалять базы данных в хранилище Hive;
* создавать и удалять таблицы в базах данных хранилища Hive;
* загружать данные, представленные как в текстовом формате с разметкой CSV/TSV, так и в формате JSON, в таблицы Hive;
* выполнять экспорт данных в форматах CSV/TSV и JSON.

В работе будут рассмотрены два способа создания и заполнения таблицы в хранилище Hive:

* с помощью средств Apache Spark с использованием оберток на языке Python;
* с помощью языка HiveSQL.

## Сценарий лабораторной работы

В данной лабораторной работе вы сделаете следующее: текстовый файл в формате JSON, содержащий записи из блогов, загрузите на файловую систему HDFS, а затем в хранилище Hive. Таким образом, данная лабораторная работа посвящена исследованию DDL (Data Definition Language) возможностей Apache Hive.

Для выполнения всех пунктов в репозитории подготовлен набор вспомогательных скриптов, которые решают исходные задачи лабораторной. Все скрипты используют набор констант, определенных в *defs.sh*. Вами будет выполнено следующее:

* импортирован файл из локальной файловой системы в свой рабочий каталог HDFS, а также созданы базы данных и таблицы, импортированы данные формата JSON в хранилище Hive (реализовано в скрипте *1\_upload\_data.sh*);
* данные из предыдущего шага экспортированы в формате TSV в файловую систему HDFS (реализовано в скрипте *2\_export\_data\_tsv.sh*);
* данные из предыдущего шага импортированы в Hive (реализовано в скрипте*3\_import\_data\_tsv.sh*);
* данные из таблицы Hive экспортированы в формате JSON Array (реализовано в скрипте *4\_export\_data\_json.sh*).

## Инструмент Apache Hive

Apache Hive — это хранилище данных для Apache Hadoop. Посредством Apache Hive можно обобщать, запрашивать и анализировать данные. Apache Hive дает возможность структурировать неструктурированные данные для проведения их дальнейшего анализа и обработки.

Взаимодействовать с хранилищем Hive можно посредством консольного интерфейса в интерактивном режиме. Консольный интерфейс Hive схож с интерфейс популярных СУБД, например, MySQL или PostgreSQL. Хранилище Hive реализует свой собственный диалект SQL — HiveSQL. Hive поддерживает запуск параллельных запросов посредством использования стандартного механизма Hadoop MapReduce.

Подготовка к лабораторной работе

Перед выполнением лабораторной работы необходимо выполнить все пункты из главы *Подготовка к лабораторной работе 1*. Убедитесь, что вы скачали репозиторий *labs\_oss*. Рассмотрим структуру каталогов, содержащихся в репозитории:

* bin — в данном каталоге содержатся вспомогательные исполняемые бинарные файлы. В данной лабораторной работе не используются.
* data — в данном каталоге содержатся данные, используемые для загрузки и анализа в лабораторных работах.
* docker — в данном каталоге содержатся вспомогательные скрипты для создания Docker окружения с Cloudera. В данной лабораторной работе не используются.
* labs — в данном каталоге содержатся специфичные для конкретной лабораторной работы файлы.
* other — в данном каталоге содержатся различные вспомогательные файлы для выполнения задач лабораторных работ.

Важно обратить внимание на скрипт *envsetup.sh*: его следует исполнять в каждой терминальной сессии перед выполнением любых других скриптов лабораторной работы. В нем содержится определение необходимых переменных и функций, на которых базируются все скрипты. Таким образом, скрипт *envsetup.sh* настраивает окружение.

Для экспорта его содержимого в текущую терминальную сессию скрипт следует исполнять командой:

|  |
| --- |
| **source** envsetup.**sh** |

## Импорт файла в рабочий каталог текущего пользователя

В экосистеме Hadoop для хранения файлов используется собственная файловая система пространства пользователя HDFS. Большинство компонентов Hadoop, осуществляющих работу с данными, используют HDFS в качестве хранилища.

Hive не является исключением: для загрузки данных в таблицы Hive они должны располагаться на файловой системе HDFS. Hive использует HDFS в том числе для хранения созданных баз данных.

В лабораторной работе 1 был рассмотрен способ взаимодействия с HDFS посредством графического интерфейса инструмента Apache Hue. В данной работе будет рассмотрен консольный интерфейс взаимодействия с HDFS.

Для работы с файловой системой HDFS используется команда **hdfs dfs.** Чтобы увидеть полный список возможных операций для взаимодействия с HDFS, нужно запустить команду:

|  |
| --- |
| hdfs dfs |

Будучи запущенной без аргументов, она выведет весь список возможных операций. Большинство команд схожи с классическими командами работы с файловой системой в окружениях UNIX.

Для вывода содержимого корневого каталога HDFS используйте команду:

|  |
| --- |
| hdfs dfs -**ls** / |

Для вывода содержимого рабочего каталога вашего пользователя введите команду:

|  |
| --- |
| hdfs dfs -ls /user/$USER |

Этот каталог будет использоваться в лабораторной работе для хранения ваших данных. Важно помнить, что файловая система HDFS в текущей настройке Cloudera на сервере является общей для всех пользователей системы. Будьте аккуратны при работе с операциями удаления или перемещения данных, чтобы не повредить данные других пользователей.

Основной целью данного пункта является подготовка данных для загрузки в Apache Hive. Как говорилось ранее, для загрузки данных в хранилище Hive необходимо, чтобы они располагались на файловой системе HDFS. Требуемые для загрузки данные располагаются в каталоге data/IBMWatson: *blogs-data-json-arr.txt*, *blogs-data-json-raw.txt*, *news-data.txt*.

Для загрузки файла blogs-data-json-arr.txt в рабочий каталог текущего пользователя следует выполнить команду:

|  |
| --- |
| **hdfs** dfs -put **data**/**IBMWatson**/blogs-**data**-json-arr.txt /user/${**USER**} |

Аргумент -put используется для копирования данных из локальной файловой системы в HDFS. Первым аргументом команда принимает путь до файла в локальной файловой системе (файл, который требуется загрузить), вторым — путь назначения (куда следует загрузить файл).

Для удобства загрузки файлов в файловую систему HDFS непосредственно в рабочий каталог текущего пользователя можно воспользоваться вспомогательной функцией **hdfs\_put\_home** из скрипта *envsetup.sh*. Загрузим с помощью неё два других файла с данными:

|  |
| --- |
| **hdfs\_put\_home** **data**/**IBMWatson**/blogs-**data**-json-raw.txt **hdfs\_put\_home** **data**/**IBMWatson**/news-**data**.txt |

После загрузки файлов с данными убедитесь в их наличии, выполнив команду вывода содержимого каталога из текста выше.

## 

## Импорт данных формата JSON Array в хранилище Hive

Исходные данные, загруженные в файловую систему HDFS в предыдущем пункте, содержат информацию об упоминаниях IBM Watson из блогов (*blogs-data-json-arr.txt* и *blogs-data-json-raw.txt*) и новостных источников (*news-data.txt*). Файлы *blogs-data-json-arr.txt* и *news-data.txt* используют формат JSON Array для хранения данных. Таким образом, данные в этих файлах представлены в виде:

|  |
| --- |
| [ {first\_entry}, {second\_entry}, ... ] |

Формат JSON Array является удобным и наиболее логичным для представления большого количество данных, имеющих единую структуру. Тем не менее, многие инструменты сбора данных формируют вывод в других форматах. Например, инструмент Apache Flume в результате сбора данных формирует выходной файл с записями в формате JSON, но в нестандартном формате:

|  |
| --- |
| {first\_entry} {second\_entry} ... |

В формате выше, т.е. набором строк с JSON записями, представлены данные в файле *blogs-data-json-raw.txt.*

Существует множество различных способов загрузить данные в хранилище Hive, но на момент написания данной работы инструмента, позволяющего загрузить в Hive данные в обоих форматах из представленных выше, разработано не было. По этой причине возникает необходимость использовать различные инструменты и средства. Рассмотрим два из них: с использованием языка HiveSQL и с использованием Apache Spark.

Произведем загрузку данных из файла *blogs-data-json-raw.txt*. Для этого следует перейти в директорию со скриптами лабораторной работы по пути *labs\_oss/labs/lab\_hive*. Создание базы данных, таблиц и загрузка данных реализованы в скрипте *1\_upload\_data.sh*.

Прежде чем запускать скрипт, рассмотрим подробнее его содержимое. В первую очередь происходит импорт необходимых переменных:

|  |
| --- |
| **source** defs.**sh** |

Прежде чем создавать базу данных, скрипт осуществляет загрузку данных на файловую систему HDFS в каталог текущего пользователя способом, рассмотренным в предыдущем разделе:

|  |
| --- |
| **log** "Create directories in HDFS for ${USER}" hdfs dfs -**mkdir** ${LAB\_HIVE\_HDFS\_FILES\_STORAGE\_PATH} hdfs dfs -**mkdir** ${LAB\_HIVE\_HDFS\_DB\_JSON\_ARRAY\_STORAGE\_PATH} hdfs dfs -**mkdir** ${LAB\_HIVE\_HDFS\_DB\_JSON\_RAW\_STORAGE\_PATH}  **log** "Upload blogs data into HDFS" hdfs dfs -put -f ${LABS\_ROOT\_DIR}/data/IBMWatson/blogs-data-json-arr.txt ${LAB\_HIVE\_HDFS\_FILES\_STORAGE\_PATH}/blogs-data-json-arr.txt hdfs dfs -put -f ${LABS\_ROOT\_DIR}/data/IBMWatson/blogs-data-json-raw.txt ${LAB\_HIVE\_HDFS\_FILES\_STORAGE\_PATH}/blogs-data-json-raw.txt |

После загрузки данных происходит подключение к серверу Hive и исполнение содержимого скрипта *create\_json\_tables.sql*:

|  |
| --- |
| log "Running Hive create\_tables.sql..." run\_hive -f ${SCRIPT\_DIR}/scripts/create\_json\_tables.sql |

Скрипт *create\_json\_tables.sql* написан с использованием языка HiveSQL. Задача скрипта состоит в создании базы данных для текущего пользователя и таблицы для хранения содержимого *blogs-data-json-raw.txt*:

|  |
| --- |
| -- HiveSQL script for creating Hive db and tables for storing data **CREATE** **DATABASE** **IF** **NOT** **EXISTS** ${env:LAB\_HIVE\_DB\_NAME}; **DROP** **TABLE** **IF** **EXISTS** ${env:LAB\_HIVE\_BLOGS\_TABLE\_NAME\_RAW}; -- Create table from set of separate JSON entries **CREATE** **EXTERNAL** **TABLE** ${env:LAB\_HIVE\_BLOGS\_TABLE\_NAME\_RAW} (  IsAdult INT,  PostSize BIGINT,  Crawled **STRING**,  Country **STRING**,  SubjectHtml **STRING**,  **Language** **STRING**,  Inserted **STRING**,  Tags **STRING**,  **Type** **STRING**,  Published **STRING**,  **Url** **STRING** ) **ROW** **FORMAT** SERDE 'org.apache.hive.hcatalog.data.JsonSerDe' LOCATION '${env:LAB\_HIVE\_HDFS\_DB\_JSON\_RAW\_STORAGE\_PATH}'; |

Все переменные из пространства env являются экспортированными переменными среды окружения из скрипта *defs.sh*.

Команда CREATE DATABASE служит для создания базы данных текущего пользователя. Имя базы данных формируется как конкатенация имени пользователя и строки “\_watson”. Эта база данных впредь будет использоваться для создания всех таблиц текущей лабораторной работы, а также при работе с данными в последующих работах.

CREATE EXTERNAL TABLE создает таблицу с именем из переменной LAB\_HIVE\_BLOGS\_TABLE\_NAME\_RAW. Директива EXTERNAL означает, что Hive не копирует данные во внутренние структуры и хранит только метаданные о них. Удаление таблицы, помеченной EXTERNAL, не влечет за собой удаление данных. Удалены будут только метаданные.

Далее определяется схема таблицы, состоящая из перечисления имен полей и их типов. Схема была составлена на основе анализа примера записи из файла *blogs-data-json-raw.txt*. Директива LOCATION указывает полный путь до файла *blogs-data-json-raw.txt* на файловой системе HDFS.

Формат представления данных JSON не является стандартным для Hive, поэтому для его интерпретации необходимо указать компонент, который сможет провести десериализацию данных в формат, понятный Hive. Для этого в Hive предусмотрен мощный механизм, позволяющий в самом тексте запроса указать тип формата строк ROW FORMAT SERDE, предоставив класс на языке Java, реализующий интерфейс сериализации-десериализации данных (SerDe от SerializationDeserialization). В качестве SerDe компонента указан org.apache.hive.hcatalog.data.JsonSerDe, являющийся частью дистрибутива Cloudera по умолчанию. Отметим, что благодаря механизму SerDe в Hive можно загрузить данные любого структурированного формата, написав собственный сериализатор-десериализатор для данного формата.

Следующим шагом скрипт *1\_upload\_data.sh* осуществляет исполнение скрипта Hive *load\_json\_data.sql*:

|  |
| --- |
| log "Running Hive load\_data.sql..." run\_hive -f ${SCRIPT\_DIR}/scripts/load\_json\_data.sql |

Задача скрипта *load\_json\_data.sql* состоит в загрузке данных из файла *blogs-data-json-raw.txt* в созданную на предыдущем шаге таблицу:

|  |
| --- |
| **LOAD** **DATA** INPATH '${env:LAB\_HIVE\_HDFS\_JSON\_RAW\_DATA\_PATH}' OVERWRITE **INTO** **TABLE** `${env:LAB\_HIVE\_BLOGS\_TABLE\_NAME\_RAW}`; |

Таким образом, на данном этапе в базе данных текущего пользователя будет создана таблица с загруженными данными из блогов из файла *blogs-data-json-raw.txt*.

Следующим шагом произведем загрузку данных из файла *blogs-data-json-arr.txt* и *news-data.txt*. Как было сказано ранее, файлы *blogs-data-json-raw.txt* и *blogs-data-json-arr.txt* содержат одинаковые данные, представленные в разных форматах. Загрузка данных из файла *blogs-data-json-arr.txt* будет с целью демонстрации возможностей Hive. Файлы *blogs-data-json-arr.txt* и *news-data.txt* хранят данные в одинаковом формате, поэтому процедура загрузки данных из них будет схожа.

Загрузка данных *blogs-data-json-arr.txt* и *news-data.txt* осуществляется также в скрипте *1\_upload\_data.sh*:

|  |
| --- |
| **log** "Running pyspark script spark\_data\_upload.py for uploading blogs-data" ${CLOUDERA\_ROOT\_DIR}/bin/spark-submit ${SCRIPT\_DIR}/scripts/load\_json\_data\_spark.py ${LABS\_ROOT\_DIR}/data/IBMWatson/blogs-data-json-arr.txt ${LAB\_HIVE\_BLOGS\_TABLE\_NAME}  **log** "Running pyspark scrpit spark\_data\_upload.py for uploading news-data" ${CLOUDERA\_ROOT\_DIR}/bin/spark-submit ${SCRIPT\_DIR}/scripts/load\_json\_data\_spark.py ${LABS\_ROOT\_DIR}/data/IBMWatson/**news**-data.txt ${LAB\_HIVE\_NEWS\_TABLE\_NAME} |

Из листинга выше видно, что для загрузки данных из двух файлов используются одинаковые команды с различием в передаваемых аргументах: пути до файла с данными и имени итоговой таблицы.

Загрузка данных *blogs-data-json-arr.txt* и *news-data.txt* осуществляется с помощью инструмента Apache Spark. Apache Spark представляет из себя мощный фреймворк для анализа больших данных. Для Spark существуют обертки на языке Python, благодаря которым можно посредством удобного API взаимодействовать с Hive. Ниже приведен фрагмент кода на языке Python с использованием средств Spark:

|  |
| --- |
| import json import sys import os  from pyspark.sql import HiveContext from pyspark import SparkContext  **if** len(sys.argv) != 3:  print('Usage: %s json\_file table\_name' % sys.argv[0])  sys.**exit**(-1)  # Context to interact with Hive storage spark = SparkContext() hiveContext = HiveContext(spark)  # Drop table if exists hiveContext.sql('DROP TABLE IF EXISTS %s' % (sys.argv[2]))  # Read given JSON jsonDF = hiveContext.read.json('file://' + sys.argv[1])  # Save parsed data in Hive storage jsonDF.write.format('orc').saveAsTable(sys.argv[2]) |

Скрипт принимает два аргумента: файл с исходными данными в формате JSON Array и имя таблицы в формате <db\_name.table\_name>, куда нужно загрузить данные. Сохранение данных происходит в таблицу в формате ORC, хранящем данные в сжатом виде.

После ознакомления с процессом загрузки данных, запустите сам скрипт:

**bash** 1\_upload\_data.sh

Таким образом, на данном этапе в базе данных текущего пользователя будут созданы три таблицы: две идентичные с данными из блогов и таблица с данными из новостных источников. Чтобы убедиться в наличии таблиц и посмотреть их содержимое, подключимся к серверу Hive и воспользуемся командным интерфейсов HiveCLI.

Исполните команду **run\_hive**, чтобы войти в интерфейс HiveCLI. Перед вводом запросов на получение данных необходимо выбрать базу данных текущего пользователя. Список существующих баз данных можно получить с помощью команды **SHOW SCHEMAS;**. С помощью команды **USE <db\_name>;** выберите базу данных текущего пользователя: как было сказано ранее, имя базы данных формируется конкатенацией имени пользователя и “\_watson”. Выведите список таблиц командой **SHOW TABLES;**. Для вывода содержимого таблицы blogs\_data используйте команду **SELECT \* FROM blogs\_data;**. Для ограничения количества выводимых строк необходимо в запросе добавить **LIMIT <count>;** после названия таблицы. Просмотрите содержимое двух других таблиц аналогичной командой, подставив имена таблиц для просмотра. Для выхода из Hive введите **QUIT;.**

Таким образом, в данном пункте были рассмотрены два способа загрузки данных формата JSON в зависимости от специфики представления формата:

* непосредственно через HiveSQL для набора JSON записей;
* с использованием средств Apache Spark для записей в формате JSON Array.

Первый способ является наиболее простым, но наименее функциональным. Второй способ предполагает написание кода на языке Python, но даёт больше гибкости при работе со входными данными. Так, перед вставкой в таблицу может быть проведена подготовка данных напрямую в скрипте загрузки.

## Экспорт данных из хранилища Hive в формате TSV

Hive предоставляет возможность произвести экспорт загруженных в таблицу данных в различных форматах. Отметим, что как и в предыдущем пункте работы для экспорта данных можно использовать Apache Spark, предоставляющий полную гибкость в работе с данными.

Тем не менее, в данном пункте будет рассмотрен более простой и быстрый способ экспорта данных в формате TSV посредством языка HiveSQL. Экспорт данных производится скриптом *2\_export\_data\_tsv.sh*:

|  |
| --- |
| log "Running Hive export\_tsv.sql..." run\_hive -f ${SCRIPT\_DIR}/scripts/export\_as\_tsv.sql |

Скрипт осуществляет вызов скрипта HiveSQL *export\_as\_tsv.sql*:

|  |
| --- |
| **INSERT** OVERWRITE **LOCAL** **DIRECTORY** '${env:LAB\_HIVE\_TSV\_EXPORT\_PATH}' **ROW** **FORMAT** **DELIMITED** **FIELDS** **TERMINATED** **BY** '\t' **SELECT** \* **FROM** ${env:LAB\_HIVE\_BLOGS\_TABLE\_NAME}; |

Экспорт данных происходит в директорию из переменной **LAB\_HIVE\_TSV\_EXPORT\_PATH**. **ROW FORMAT DELIMITED** означает, что каждая запись из таблицы выводится в отдельную строку в результирующем файле. **FIELDS TERMINATED BY '\t'** определяет символ-разделитель между полями каждой записи. Для формата TSV разделителем является Tab, или \t в терминах ASCII. Строка **SELECT \* FROM ${env:LAB\_HIVE\_BLOGS\_TABLE\_NAME};** определяет набор данных, которые необходимо экспортировать.

Для запуска скрипта экспорта введите в командную строку:

**bash** 2\_export\_data\_tsv.sh

## Импорт данных из формата TSV в хранилище Hive

На данном этапе будет произведена обратная операция: импорт данных в формате TSV из предыдущего. Импорт данных в формате TSV осуществляется в скрипте *3\_import\_data\_tsv.sh*:

|  |
| --- |
| log "Running Hive create\_tsv\_table.sql..." run\_hive -f ${SCRIPT\_DIR}/scripts/create\_tsv\_table.sql |

Скрипт осуществляет вызов скрипта HiveSQL *create\_csv\_table.sql*:

|  |
| --- |
| -- HiveSQL script for creating Hive db and tables for storing data  **CREATE** **DATABASE** **IF** **NOT** **EXISTS** ${env:LAB\_HIVE\_DB\_NAME};  **DROP** **TABLE** **IF** **EXISTS** ${env:LAB\_HIVE\_TSV\_TABLE\_NAME};  -- Create table from TSV file  **CREATE** **EXTERNAL** **TABLE** ${env:LAB\_HIVE\_TSV\_TABLE\_NAME} (  IsAdult INT,  PostSize BIGINT,  Crawled **STRING**,  Country **STRING**,  SubjectHtml **STRING**,  **Language** **STRING**,  Inserted **STRING**,  Tags **STRING**,  **Type** **STRING**,  Published **STRING**,  **Url** **STRING** ) **ROW** **FORMAT** **DELIMITED** **FIELDS** **TERMINATED** **BY** '\t' **STORED** **AS** TEXTFILE LOCATION '${env:LAB\_HIVE\_TSV\_EXPORT\_PATH}'; |

Сравните листинг выше с листингом из пункта 4.5. Форматы запросов отличаются в указании формата входных данных. Для импорта данных из файла в формате TSV тип содержимого описывается как **ROW FORMAT DELIMITED ; FIELDS TERMINATED BY ‘\t’**. Обратите внимание, что именно такой формат использовался при экспорте данных в предыдущем пункте.

Для запуска скрипта импорта из TSV введите в командную строку:

**bash** 3\_import\_data\_tsv.sh

## Экспорт данных в формате JSON Array

Экспорт данных в формате JSON Array, в отличие от TSV, невозможно выполнить стандартными средствами HiveSQL. Для решения задачи будет использована сторонняя библиотека Brickhouse, предоставляющая набор UDF (User Defined Functions) для Hive, в том числе функцию **to\_json**.

Экспорт данных осуществляется скриптом *4\_export\_data\_json.sh*:

|  |
| --- |
| log "Running Hive export\_as\_json.sql..." run\_hive -f ${SCRIPT\_DIR}/scripts/export\_as\_json.sql |

Скрипт осуществляет вызов скрипта HiveSQL *export\_as\_json.sql.* В первую очередь скрипт загружает в Hive внешнюю библиотеку и создает UDF на базе метода из этой библиотеки:

**ADD** JAR ${env:LAB\_HIVE\_BRICKHOUSE\_JAR\_PATH};  
CREATE TEMPORARY FUNCTION to\_json AS 'brickhouse.udf.json.ToJsonUDF';

Далее происходит экспорт данных с использованием функции **to\_json**. В нее посредством вызова **named\_struct** передается отображение схемы исходной таблицы на имена полей в записи объекта в формате JSON:

**INSERT** OVERWRITE **LOCAL** **DIRECTORY** '${env:LAB\_HIVE\_JSON\_EXPORT\_PATH}'   
**ROW** **FORMAT** **DELIMITED**   
**SELECT** to\_json(   
 named\_struct(   
 'IsAdult', t.isadult,   
 'PostSize', t.postsize,   
 'Crawled', t.crawled,   
 'Country', t.country,   
 'SubjectHtml', t.subjecthtml,   
 'Language', t.language,   
 'Inserted', t.inserted,   
 'Tags', t.tags,   
 'Type', t.type,   
 'Published', t.published,   
 'Url', t.url  
 )  
) **FROM** ${env:LAB\_HIVE\_BLOGS\_TABLE\_NAME} **as** t;

Для запуска скрипта экспорта введите в командную строку:

**bash** 4\_export\_data\_json.sh

# Лабораторная работа №3. Анализ структурированных и неструктурированных данных с использованием Apache Hue

## Цели и задачи лабораторной работы

Целью данной лабораторной работы является ознакомление с функционалом Apache Hue для осуществления работы с базами данных Apache Hive.

Для выполнения данной лабораторной работы будут использованы неструктурированные данные из общедоступных источников в Интернете об IBM Watson. В ходе работы будут производиться объединение этих данных и их анализ, цель которого — определить уровень популярности IBM Watson в СМИ и оценить эффективность проводимых рекламных кампаний. Другими словами, решить типичную задачу для бизнеса: понять, насколько популярен тот или иной известный бренд, проект, сервис или услуга.

В ходе выполнения данной лабораторной работы вы научитесь:

* работать с таблицами Apache Hive в инструменте Apache Hue;
* строить графики и агрегации;
* осуществлять анализ и обработку данных.

## Сценарий лабораторной работы

1. Преобразование данных. Часто перед анализом данных их необходимо обработать. На этом шаге вы удалите ненужные данные, а после объедините собранные из новостей и блогов данные в единую таблицу.
2. Оценка популярности IBM Watson. После завершения подготовки данных можно приступать к их анализу. Анализ данных на данном этапе заключается в оценке популярности IBM Watson в разных странах (будет осуществлен подсчет количества новостных сообщений, написанных на разных языках).
3. Оценка эффективности рекламных кампаний IBM Watson. На этом шаге вы узнаете, какие СМИ больше других размещают новостные записи об IBM Watson.

## Инструмент Apache Hue

Apache Hue — это интерактивный веб-редактор в стеке Hadoop, который позволяет работать с данными, делиться ими и визуализировать их. Apache Hue предоставляет облачный редактор SQL с открытым исходным кодом для просмотра, обработки и анализа данных, представленных в виде таблиц, путем написания к ним запросов.

Используя Apache Hue вы сможете:

* исследовать, просматривать и импортировать данные (просматривать каталоги HDFS из облачного хранилища, документы, базы данных, разворачивая их до конкретных таблиц);
* создать собственную пользовательскую панель мониторинга для анализа данных, сохранить повторяющиеся пользовательские сценарии, настроить рабочие процессы, задав их запуск на постоянной основе;
* получить советы экспертов и подсказки, работая с любым приложением на центральной панели главной страницы.

## Подготовка к лабораторной работе

В данной лабораторной работе вы будете работать с данными в Apache Hue. Данные собраны, обработаны и представлены в виде таблиц Apache Hive в ходе выполнения предыдущей лабораторной работы. Перед выполнением лабораторной работы убедитесь, что вы выполнили все, что описано в разделе *Подготовка к лабораторной работе 1*, и загрузили в Apache Hive в вашу личную базу данных таблицы *blogs\_data* и *news\_data*.

Для работы с данными в Apache Hue вам необходимо открыть **Query** **→ Editor → Hive** (Рис. 1).

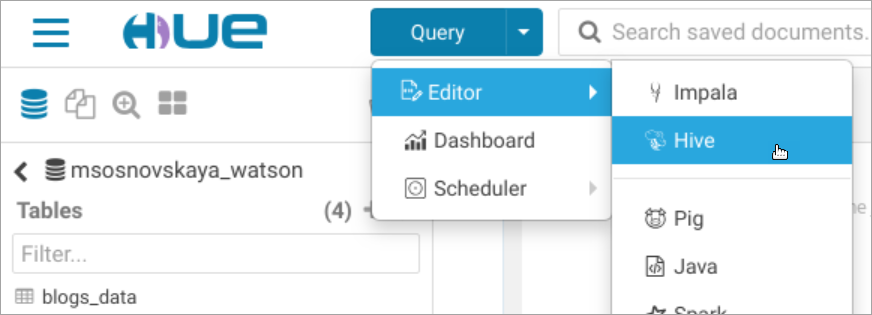


Рис. 1. Начало работы в Apache Hue

Убедитесь, что для дальнейшего анализа выбрана созданная вами база данных. Для этого в левом верхнем углу нажмите  **→ Hive** и выберите имя созданной вами базы данных.Имя созданной вами базы данных отобразится в центральной части экрана (Рис. 2).

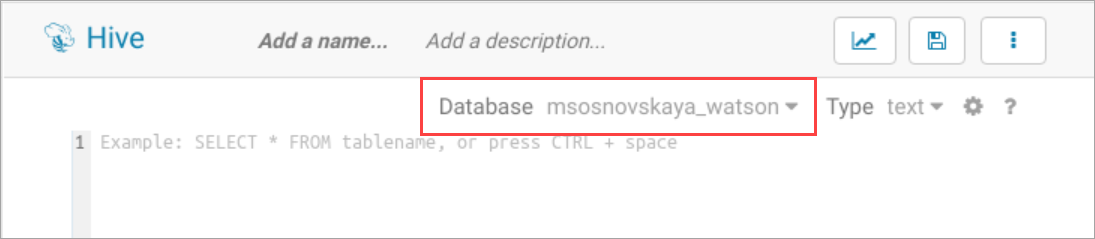


Рис. 2. Имя выбранной базы данных

## Преобразование данных

Нередко перед анализом данные необходимо дополнительно обработать. Например, преобразовать содержание и структуру таблиц, а только после приступать к исследованию. В данном разделе мы разберем удаление столбцов с лишними данными и объединение данных из двух таблиц.

### **Удаление ненужных данных**

В текущей лабораторной работе данные об IBM Watson представлены в виде таблиц, каждый из столбцов которых содержит ту или иную информацию о СМИ из новостей и блогов. Для дальнейшего анализа нам понадобится только часть столбцов. Следовательно, ненужные столбцы нужно удалить, создав новую таблицу.

В Apache Hue нет инструмента, который бы позволял удалять из таблиц ненужные столбцы. Тем не менее вы можете написать SQL запрос, который позволит вам выделить только нужные вам данные.

Чтобы посмотреть исходную таблицу, напишем запрос (Рис. 3). Введите в окно для ввода:

|  |
| --- |
| SELECT \* FROM news\_data; |

Чтобы выполнить запрос, нажмите  и дождитесь выполнения запроса.

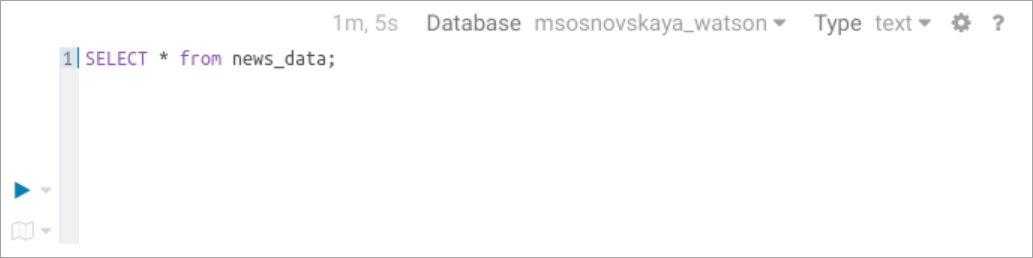


Рис. 3. Просмотр таблицы news\_data

Если запрос выполняется долго, вы увидите уведомление об этом в верхнем правом углу экрана на вкладке **Jobs** (Рис. 4). При нажатии на вкладку откроется список исполняемых, успешно и неуспешно завершившихся процессов.

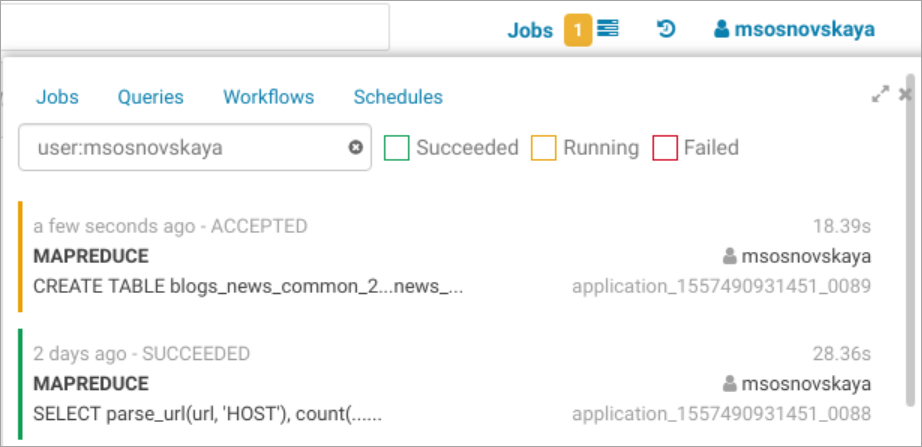


Рис. 4. Статусы запросов

Результат выполнения запроса можно посмотреть на вкладке **Results**. Чтобы развернуть таблицу на весь экран, наведите курсор на вкладку **Results** и нажмите . Чтобы посмотреть все столбцы и строки, воспользуйтесь скроллами. Чтобы закрыть таблицу, нажмите  (Рис. 5).

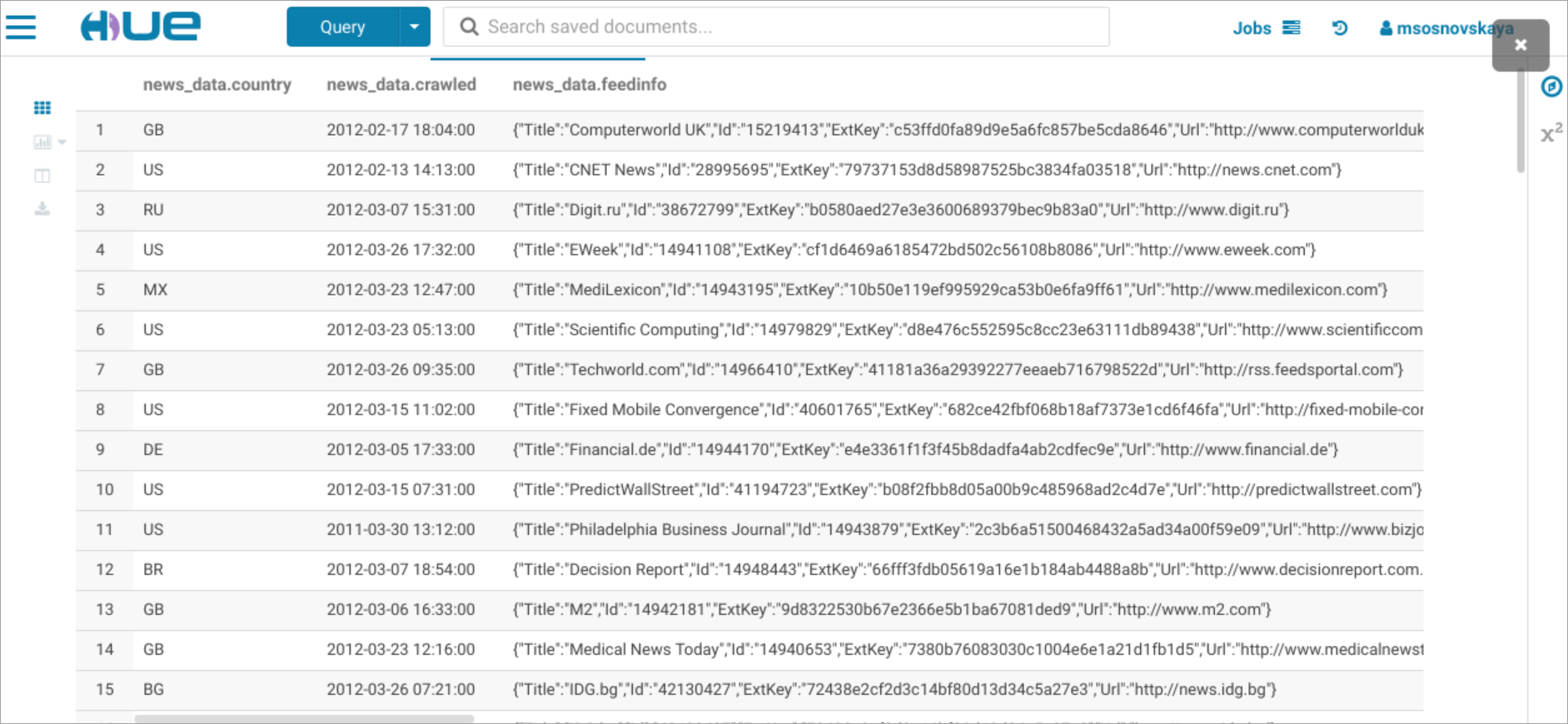


Рис. 5. Таблица news\_data

В новой таблице нам необходимо оставить столбцы Country, FeedInfo, Language, Published, SubjectHtml, Tags, Type и Url. Для этого напишем запрос:

|  |
| --- |
| SELECT country, feedinfo, language, published, subjecthtml, tags, type, url FROM news\_data; |

После выполнения запроса на вкладке Results отобразится таблица с только нужными вам столбцами (Рис. 6).

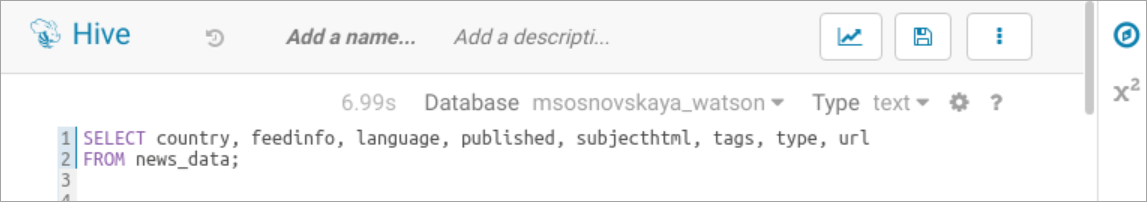


Рис. 6. Выбор нужных столбцов

### **Объединение данных**

Теперь объединим данные из новостей и блогов в одну таблицу. Чтобы при объединении не возникло ошибок, названия, порядок и количество столбцов первой таблицы должны совпадать со второй. Для объединения двух таблиц напишем запрос:

|  |
| --- |
| CREATE TABLE blogs\_news\_common STORED AS orc  AS SELECT \* FROM blogs\_data UNION ALL SELECT \* FROM news\_data; |

После выполнения запроса (Рис. 7) будет создана новая таблица blogs\_news\_common, содержащая данные из новостей и блогов.

UNION ALL — оператор, создающий таблицу без удаления повторяющихся записей. STORED AS orc — создание таблицы в специальном формате хранения данных в Apache Hive, оптимизированном по количеству занимаемого на месте диска.

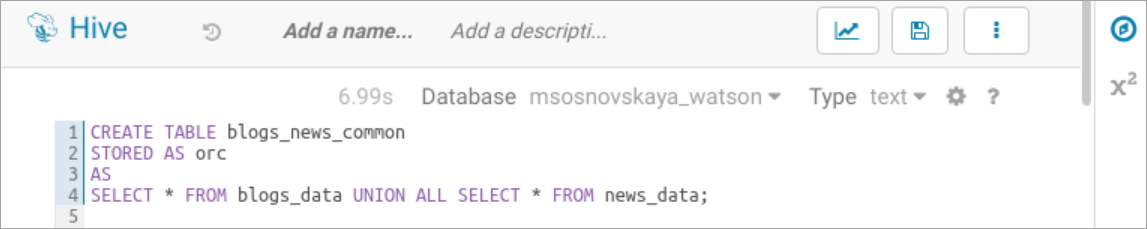


Рис. 7. Объединение таблиц

## Анализ данных

### **Сортировка**

Для того чтобы исследовать степень охвата IBM Watson, произведем сортировку записей по языку и типу, потому что большинство записей содержат данные о языке их источника. Чтобы отсортировать записи, напишем запрос:

|  |
| --- |
| SELECT \* FROM blogs\_news\_common ORDER BY language DESC, type ASC LIMIT 50; |

После выполнения такого запроса (Рис. 8) вы увидите первые 50 записей, отсортированные по языку и типу (Рис. 9). Порядок сортировки по языку убывающий, по типу — возрастающий.

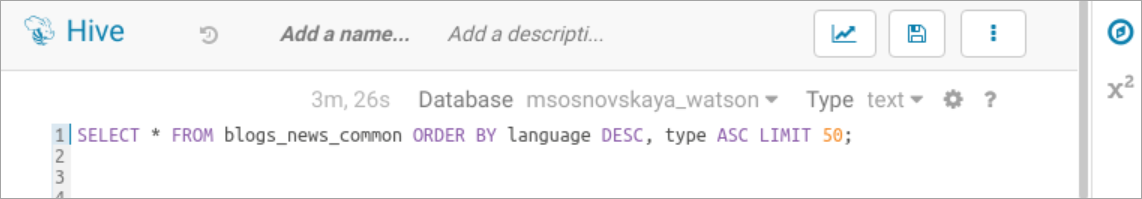


Рис. 8. Сортировка записей

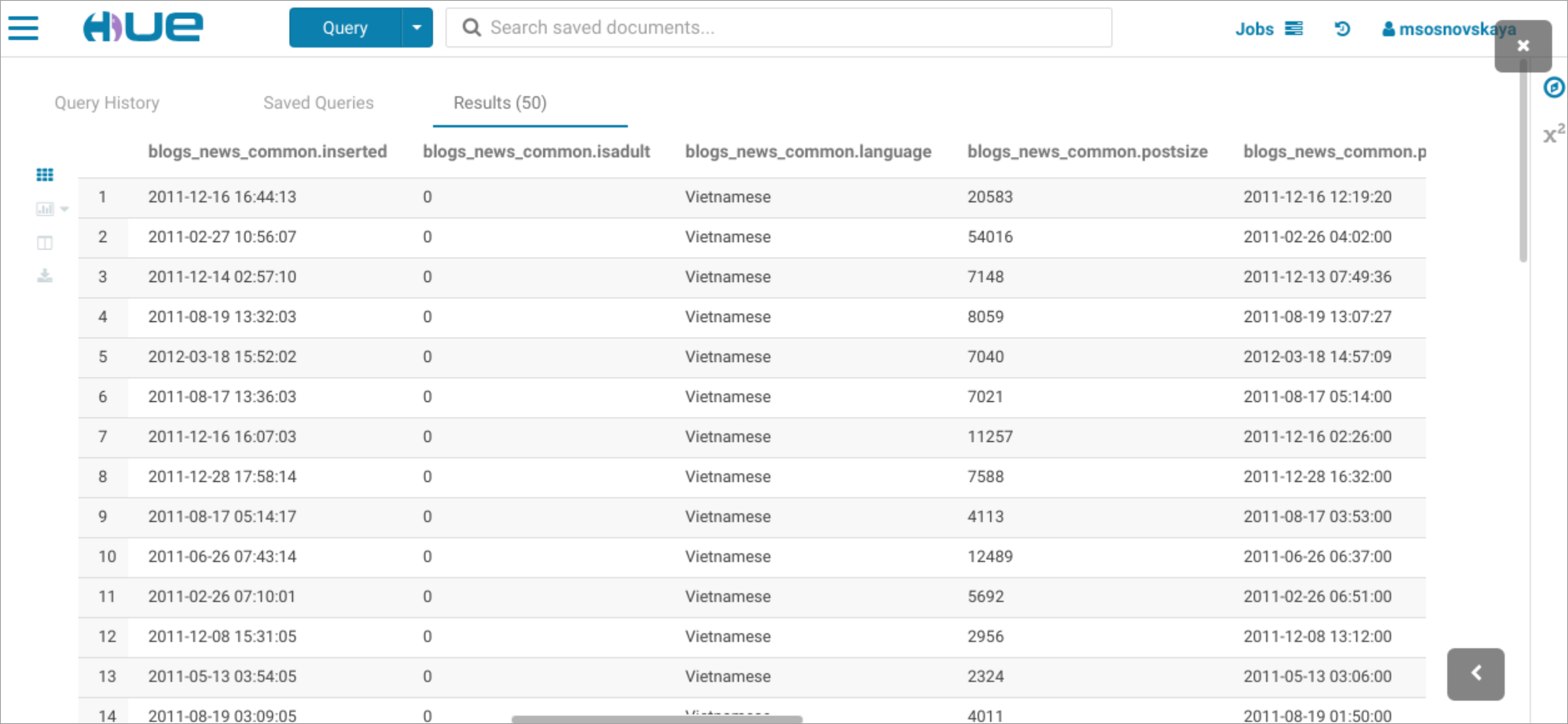


Рис. 9. Записи, отсортированные по языку и типу

### **Построение диаграммы**

Данные о степени известности IBM Watson на разных языках можно представить как в табличном виде, так и в виде диаграммы. В Apache Hue есть возможность представления результатов анализа в виде нескольких типов диаграмм. В данном разделе мы построим простую круговую диаграмму.

Для этого выполните запрос (Рис. 10):

|  |
| --- |
| SELECT language, COUNT(\*) FROM blogs\_news\_common GROUP BY language; |

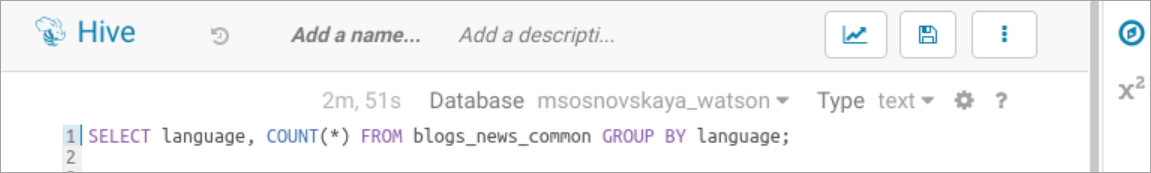


Рис. 10. Группировка по языку

После выполнения запроса вы увидите данные, сгруппированные по языку (Рис. 11). COUNT(\*) — оператор для подсчета количества строк. В данном случае он подсчитывает количество записей на разных языках. Обратите внимание, что не у всех записей указан язык. Так бывает при сборе данных из большого числа источников.

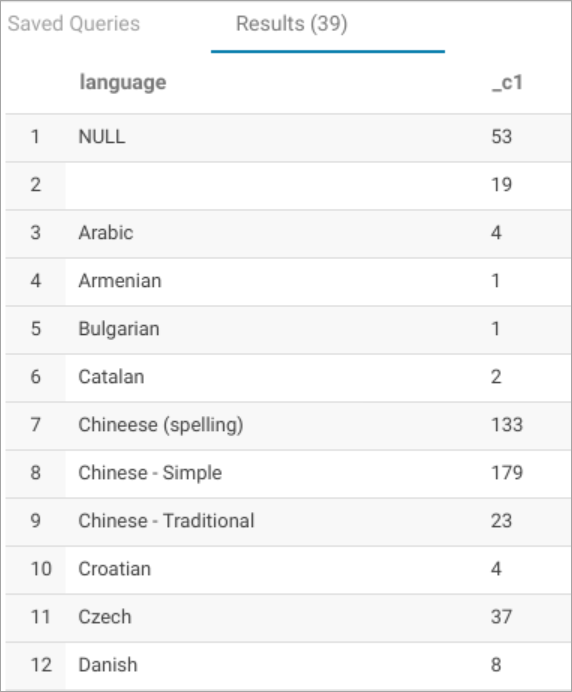


Рис. 11. Таблица с данными, сгруппированными по языку

Построим круговую диаграмму, отражающую степень известности IBM Watson на разных языках. Для этого нажмите  в нижней части экрана и выберите **Pie** (Рис. 12).

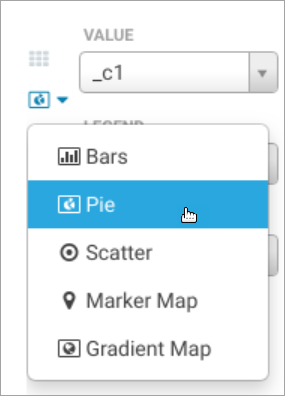


Рис. 12. Выбор круговой диаграммы

Обратите внимание, что на круговой диаграмме видно, что большинство записей написаны на английском языке (Рис. 13). Чтобы определить какому цвету соответствует какой язык, необходимо навести курсор на секцию круговой диаграммы.

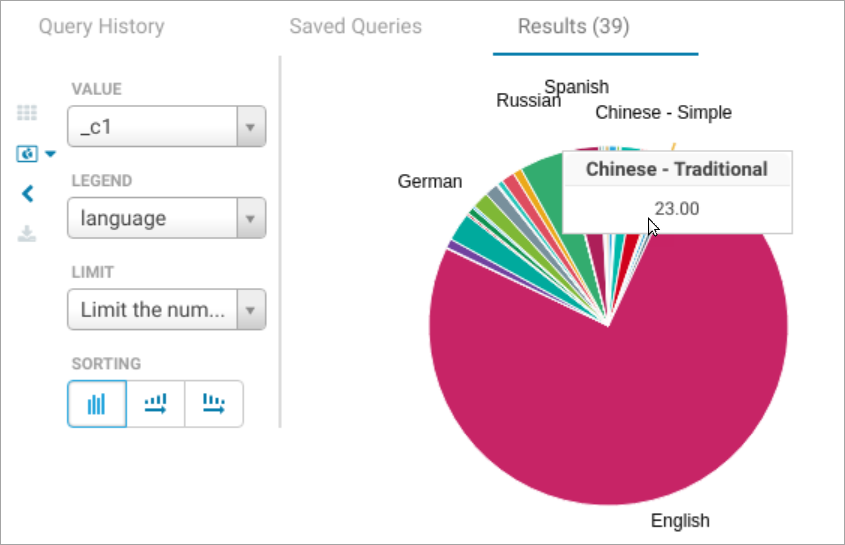


Рис. 13. Круговая диаграмма языков

### **Очистка данных**

Если вы внимательно посмотрите на круговую диаграмму, то заметите, что несколько секций на ней занимают разные варианты китайского языка. Это ситуация не является редкой при сборе данных из многих источников (в нашем случае различных новостных сайтов) — у нас есть по сути одинаковые данные, которые представляются как различные, что мешает нам корректно их обрабатывать.

Посмотрим, как в Apache Hue сделать различные варианты китайского одним языком, а после визуализировать данные о степени популярности IBM Watson на разных языках с учетом очистки данных.

Создадим новую таблицу blogs\_new\_common\_chinese из двух полей. Первое поле — название языка, второе — количество новостей, написанных на этом языке. Пока не будем включать в таблицу данные о различных вариантах китайского языка. Для этого выполним запрос:

|  |
| --- |
| CREATE TABLE blogs\_new\_common\_chinese AS SELECT language, count(\*) FROM blogs\_news\_common WHERE language NOT LIKE 'Chin%' GROUP BY language; |

После выполнения запроса (Рис. 14) вы получите таблицу с данными о количестве записей на всех языках, кроме вариантов китайского (Рис. 15).

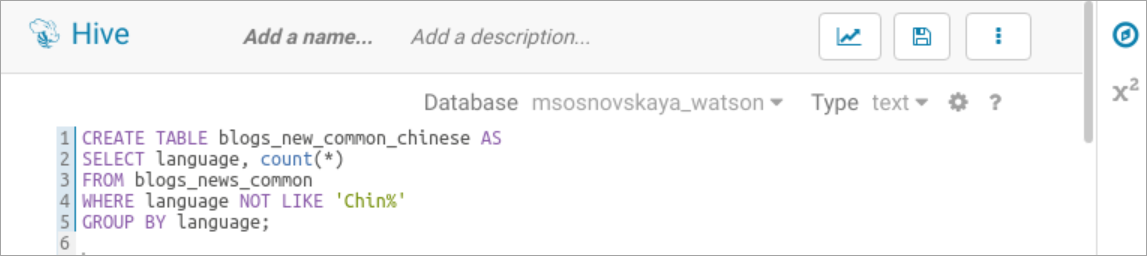


Рис. 14. Создание таблицы с данными всех языках, кроме китайского



Рис. 15. Таблица с данными о всех языках, кроме китайского

Теперь посчитаем, сколько записей написано на всех вариантах китайского, и добавим эти данные в созданную нами таблицу. Для этого введите в поле для ввода:

|  |
| --- |
| **INSERT** **INTO** blogs\_new\_common\_chinese **SELECT** 'Chinese', **count**(**language**) **FROM** blogs\_news\_common **WHERE** **language** **LIKE** 'Chin%'; |

После выполнения запроса (Рис. 16) в вашу таблицу добавится строка, содержащая общее количество записей на всех вариантах китайского языка.



Рис. 16. Добавление в таблицу данных о количестве записей на китайском

Теперь попробуйте построить круговую диаграмму (Рис. 17). Сравните результат с диаграммой из прошлого раздела. Обратите внимание, что после очистки данных китайский язык занимает второе место (до этого второе место занимал русский язык).

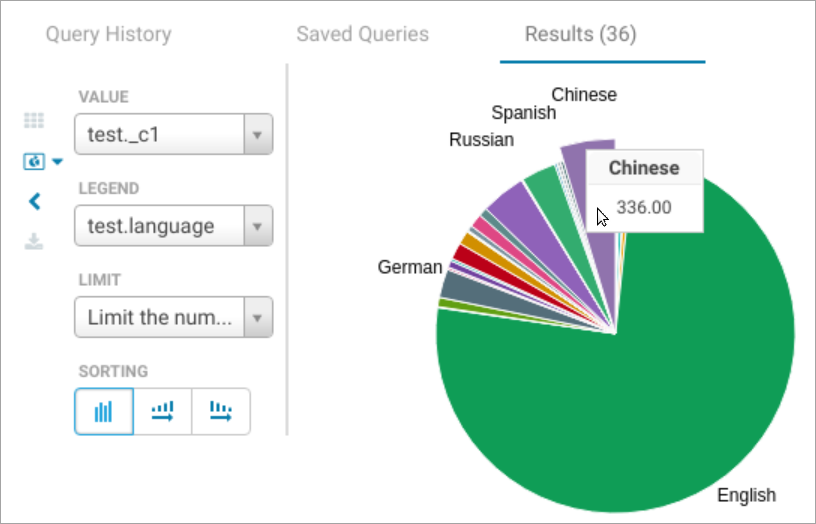


Рис. 17. Круговая диаграмма с объединенным китайским языком

### **Выборка данных**

Теперь детализируем степень охвата IBM Watson в блогах и новостях, выделив записи, опубликованные в Великобритании. Мы будем использовать довольно простой подход для исследования, но надо понимать, что подробный анализ больших объемов данных часто занимает много времени и проводится итеративно.

Выберем записи, URL-адрес которых заканчивается на .uk или в колонке Country которых значится GB. Для этого напишем запрос:

|  |
| --- |
| **SELECT** \* **from** blogs\_news\_common **WHERE** country = 'GB' **OR** **url** **LIKE** '%.uk%'; |

После выполнения запроса (Рис. 18) вы получите список записей о новостях, опубликованных в Великобритании (Рис. 19).

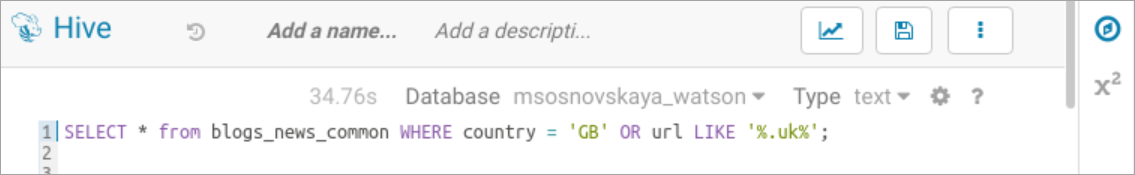
****

Рис. 18. Выделение записей о новостях в Великобритании

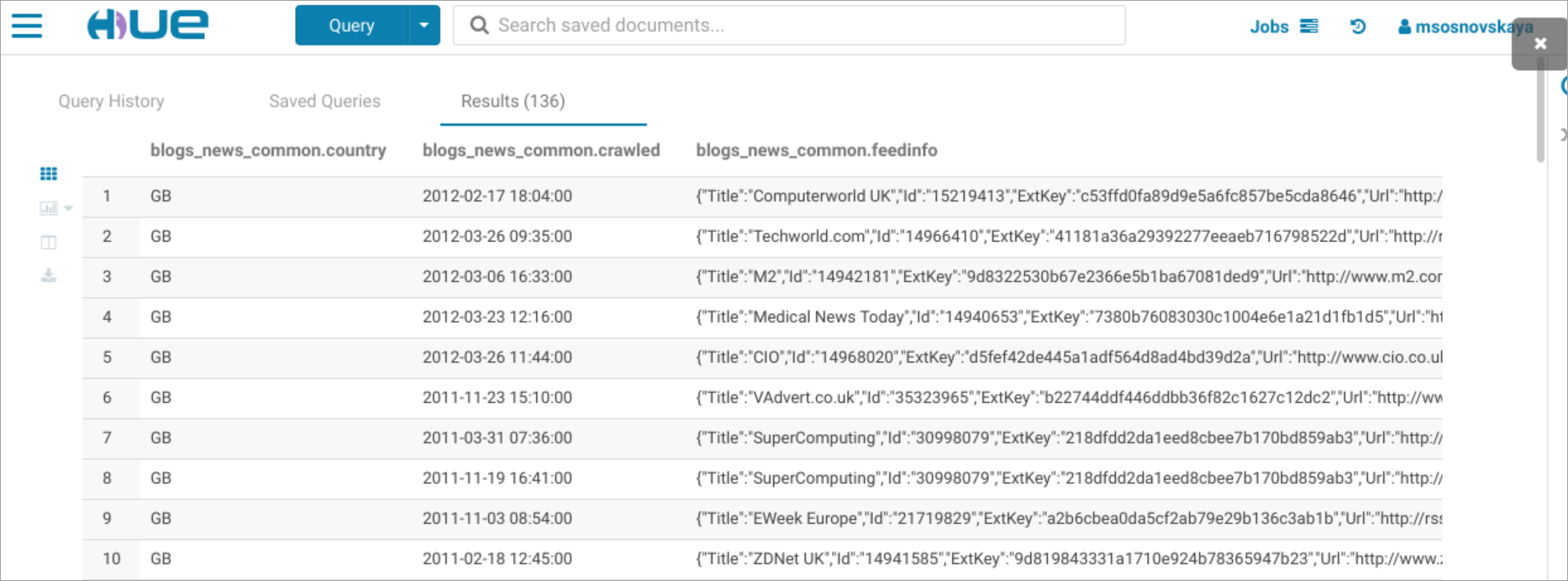
****

Рис. 19. Записи о новостях в Великобритании

Теперь нарисуем круговую диаграмму, на которой будут видны 10 сайтов, которые чаще всего публикуют новости об IBM Watson в Великобритании. Для этого выполним запрос:

|  |
| --- |
| **SELECT** parse\_url(**url**, 'HOST'), **count**(parse\_url(**url**, 'HOST')) **from** blogs\_news\_common **WHERE** country = 'GB' **OR** **url** **LIKE** '%.uk%' **GROUP** **BY** parse\_url(**url**, 'HOST'); |

parse\_url(url, ‘HOST’) — функция, которая парсит поле url и выделяет из него имя хоста.

При построении круговой диаграммы в поле Limit выберите 10 и нажмите , чтобы отсортировать данные по убыванию (Рис.20).

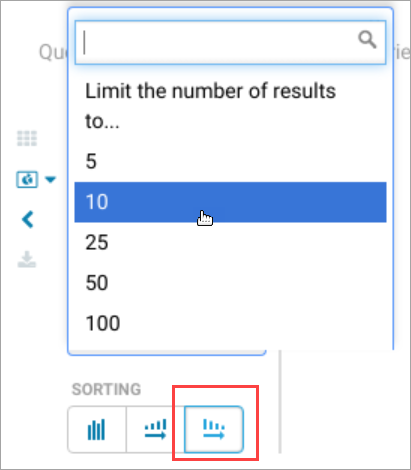


Рис. 20. Настройка круговой диаграммы

На круговой диаграмме видны сайты, публикующие новости об IBM Watson в Великобритании чаще других (Рис. 21).

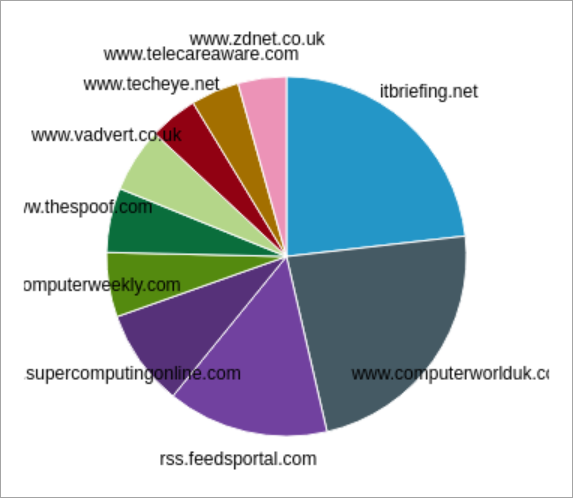


Рис. 21. 10 сайтов с наибольшим количеством записей об IBM Watson в Великобритании

### **Выборка уникальных записей**

Эффективность рекламной кампании является одним из показателей, позволяющим определить степень освещенности бренда, товара или услуги в СМИ. В данном разделе, используя Apache Hue, вы установите количество новостных сайтов и блогов, размещающих записи об IBM Watson, а также выделите сайты, содержащие наибольшее количество упоминаний.

Общее количество записей в новостях и блогах составляет 7273 (Рис. 22).

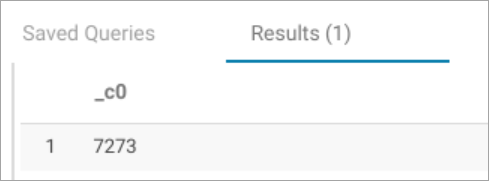


Рис. 22. Общее количество записей в новостях и блогах

Сгруппируем записи по сайтам. Для этого напишем запрос:

|  |
| --- |
| **SELECT** parse\_url(**url**, 'HOST'), **count**(parse\_url(**url**, 'HOST')) **FROM** blogs\_news\_common **GROUP** **BY** parse\_url(**url**, 'HOST'); |

После выполнения запроса (Рис. 23) количество записей в таблице уменьшилось и теперь составляет 2787, что значит, что некоторые сайты не один раз публиковали записи об IBM Watson (Рис. 24).

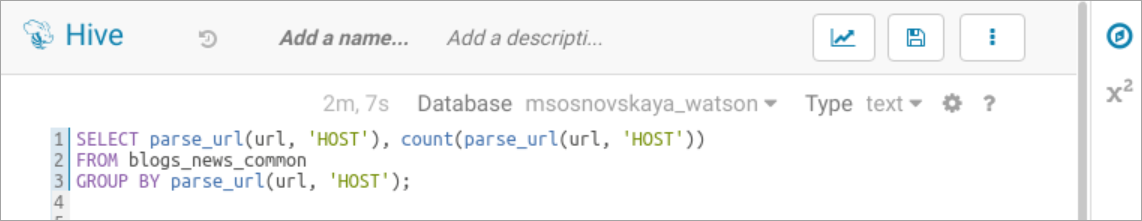


Рис. 23. Группировка записей по сайтам

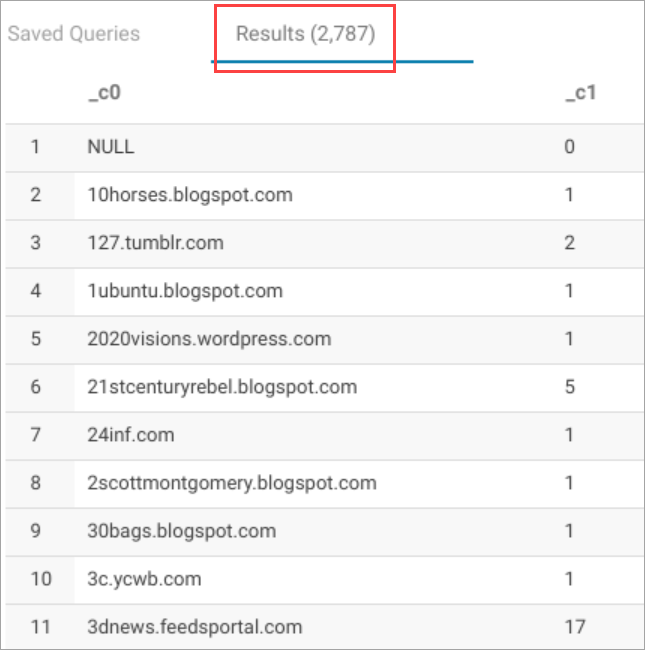


Рис. 24. Записи, сгруппированные по сайтам

### **Выборка популярных записей**

Теперь получим 10 сайтов с наибольшим числом записей об IBM Watson и представим результат в виде диаграммы. Для этого напишем запрос (Рис. 25):

|  |
| --- |
| **SELECT** parse\_url(**url**, 'HOST'), **count**(parse\_url(**url**, 'HOST')) **AS** t **from** blogs\_news\_common **GROUP** **BY** parse\_url(**url**, 'HOST') **ORDER** **BY** t **DESC**; |

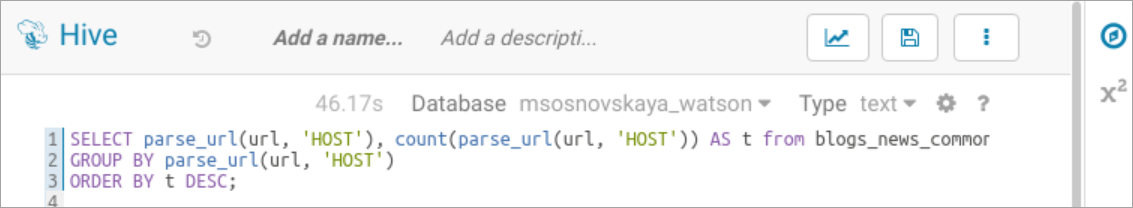


Рис. 25. Записи в порядке убывания популярности сайтов

После выполнения запроса вы получите список сайтов, отсортированный по убыванию опубликованного на них количества записей об IBM Watson. Постройте диаграмму, чтобы посмотреть первые 10 сайтов, выставив настройки, обозначенные ниже (Рис. 26).

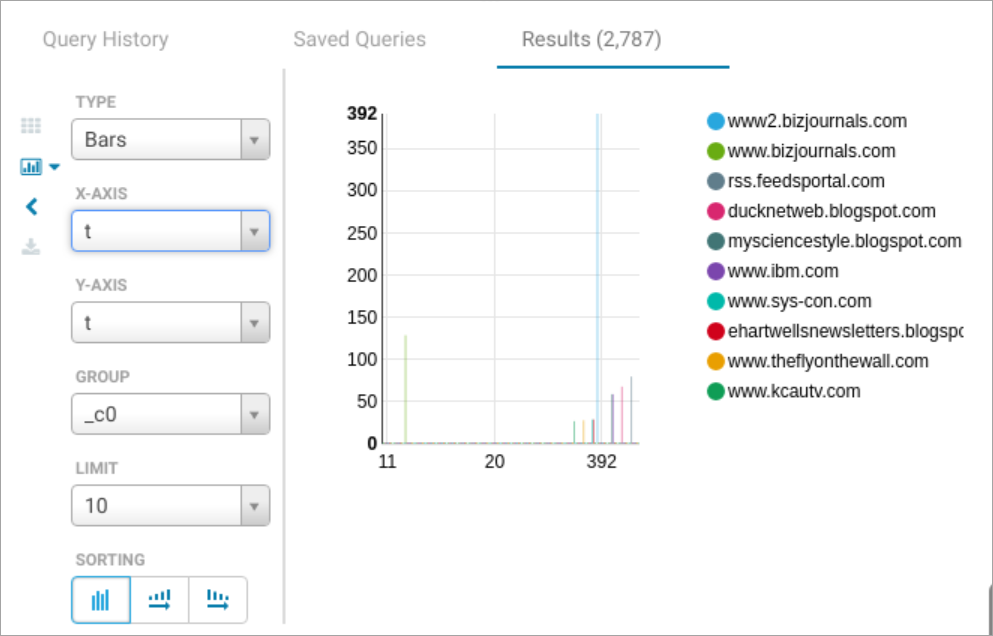


Рис. 26. 10 самых популярных сайтов

# 

# Лабораторная работа №4. Анализ потоковой информации с использованием специализированных языков обработки потоков данных

## Цели и задачи лабораторной работы

Целью данной лабораторной работы является ознакомление с функционалом Spark Streaming в связке с Apache Kafka для осуществления работы с данными в реальном времени, а также выполнение обучающей задачи по обработке и представлению данных в реальном времени.

Все задания лабораторной работы выполняются из командной строки интерпретатора операционной системы GNU/Linux Bash. В результате выполнения лабораторной работы вы научитесь выполнять следующие задачи:

● производить первичную настройку Apache Kafka;

● создавать и удалять топики Kafka;

● записывать и считывать данные из топиков Kafka;

● обрабатывать данные из топиков Kafka средствами Spark Streaming;

● выполнять экспорт данных в форматах CSV/TSV и JSON.

## Инструменты Kafka и Spark Streaming

Spark Streaming — это подпроект Apache Spark. Spark — это платформа пакетной обработки, аналогичная Apache Hadoop, а Spark Streaming — инструмент обработки в реальном времени, работающий поверх движка Spark, тем самым предлагая явное преимущество, заключающееся в сближении пакетной обработки и обработки в реальном времени, поскольку в идеале для обоих может использоваться одна и та же кодовая база.



Рис. 1. Spark Streaming

Spark Streaming — это так называемая микропакетная среда, которая использует временные интервалы. Он использует так называемые D-Stream (дискретизированный поток), которые структурируют вычисления в виде небольших наборов коротких, не имеющих состояния и детерминированных задач. Состояние распределяется и сохраняется в отказоустойчивых RDDs. D-Stream может быть создан из различных источников данных, таких как Kafka, Flume или HDFS, предлагая многие из тех же операций, которые доступны для RDD с дополнительными операциями, типичными для временных операций, такими как скользящие окна.

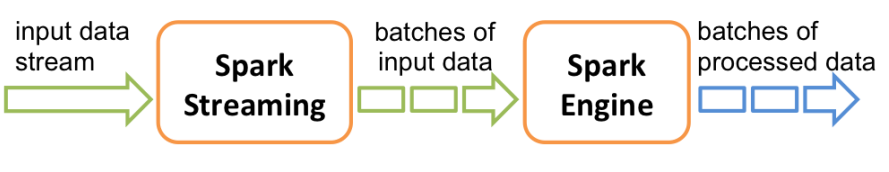


Рис. 2. Spark DStream

Apache Kafka — это распределенная система обмена сообщениями с публикацией и подпиской. Система Kafka построена поверх службы синхронизации ZooKeeper. По сути, это система, которая может очень быстро и эффективно передавать ваши сообщения. В качестве сообщений могут выступать любые типы данных, поскольку для Kafka это всего лишь последовательность байтов. Apache Kafka может работать как на одной машине, так и на нескольких, которые вместе образуют кластер и повышают эффективность всей системы.

Важно понять то, как работает Apache Kafka. Мы можем писать в нее сообщения, и можем читать из нее. Все сообщения в Kafka принадлежат к тому или иному топику (topic). Топик — это как заглавие, и он должен быть определен для каждого сообщения, которое мы хотим передать в Apache Kafka. Точно также, если мы собираемся читать из Kafka сообщения, мы должны указать, с каким топиком будут эти сообщения.

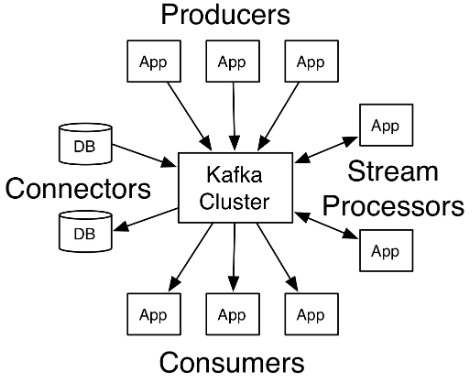


Рис. 3. Kafka core APIs

## Работа с Kafka

### Подготовка к работе

В данной лабораторной работе вы будете работать с данными в Kafka, а после и в Spark. Данные собраны, обработаны и представлены в виде таблиц Apache Hive в ходе выполнения предыдущей лабораторной работы. Перед выполнением лабораторной работы убедитесь, что вы выполнили все, что описано в разделе *Подготовка к лабораторной работе 1*.

Для уточнения работы Spark с локальными данными можно обратиться к лабораторной работе №2.

## Kafka и Zookeeper

Для начала проверим доступность Kafka и Zookeeper, для этого введем следующие команды в консоль:

|  |
| --- |
| **netcat -vz localhost 2181 netcat -vz localhost 9092** |

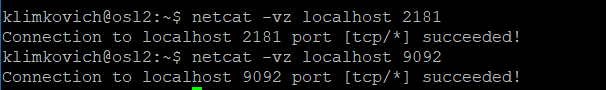


Рис. 4. Пример успешной проверки доступности сервисов.

При замене localhost на IP адрес удаленного сервера данные команды позволяют узнать их доступность.

При доступности служб мы можем начать создание темы(topic) для сбора данных. Команда для создания новой темы выглядит как:

**kafka-topics --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic topic\_<username>**

где localhost:2181 – адрес и порт запущенного Zookeper, --replication-factor 1 – количество реплик данных, которые будет хранить тема (число не может быть больше, чем количество запущенных сервисов Kafka в кластере), --partitions 1 – количество разделов и topic topic\_<username> – название новой темы, состоящее из имени пользователя.

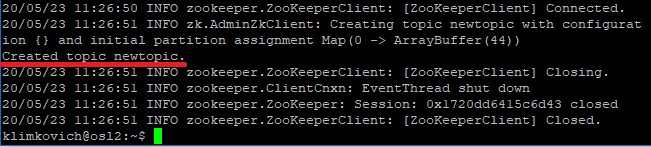


Рис. 5. Пример успешного создания темы.

Для того чтобы проверить весь список активных тем выполните следующее:

**kafka-topics --list --zookeeper localhost:2181**

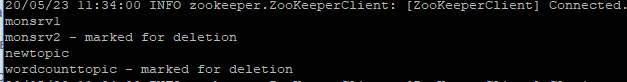


Рис. 6. Список активных тем.

Чтобы узнать больше информации о темах:

**kafka-topics --describe --zookeeper localhost:2181 --topic monsrv1**



Рис. 7. Информация о теме monsrv1.

Для того чтобы отметить тему для удаления:

**kafka-topics --zookeeper localhost:2181 --topic monsrv1 --delete**



Рис. 8. Удаление темы monsrv1.

Теперь, когда у нас уже создана тема, в которую мы можем передавать данные, то стоит создать первого Издателя (Producer). Издатель, это компонент Kafka предназначенный для передачи сообщений в темы или любые другие совместимые внешние компоненты. Так же он предоставляет возможность вручную вводить данные в тему через консоль, что бывает удобно для отладки и первичных тестов.

Чтобы создать такого издателя:

**kafka-console-producer --broker-list localhost:9092 --topic topic\_<username>**

после выполнения команды у вас появится возможность передавать сообщения в тему через консоль. Для дальнейшей наглядности советуется вносить упорядоченные данные, например слова с цифрой на конце.

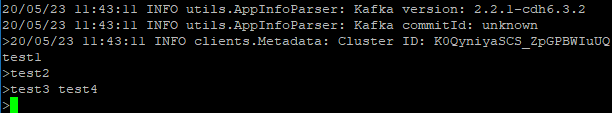


Рис. 9. Передача через консоль данных в Тему.

Для завершения работы издателя и дальнейшей возможности работы с консолью завершите процесс сочетанием Ctrl+C.

Стоит обратить особое внимание при написании целевой Темы, т.к. даже если вы ошибетесь в написании вы получите только одно предупреждение, после чего топик с ошибочным названием будет создан с параметрами по умолчанию.

На данном этапе мы создали тему, которая агрегирует и сохраняет полученные сообщения, создали издателя, который передает сообщения с данными в эту тему, теперь надо получить эти сообщения из темы. Для этого нам нужно создать Подписчика (Сonsumer).

Перед выполнением следующих шагов откроем еще одну консоль запустив ещё один экземпляр Putty. В новой консоли создадим подписчика для нашей темы:

**kafka-console-consumer --bootstrap-server localhost:9092 --topic topic\_<username>**

После выполнения команды может показаться что ничего не работает, но это не так, просто по умолчанию подписчик прослушивает тему только с момента своего запуска. В итоге мы видим успешную подписку, но отсутствие данных (рис. 10):

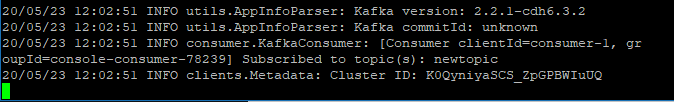


Рис. 10. Успешный запуск подписчика.

Вернемся к предыдущей консоли, не закрывая экземпляр с подписчиком, и повторим запуск издателя. Теперь, при передаче данных издателем мы можем почти моментально увидеть их полученными подписчиком:



Рис. 11. Передача и получение данных.

Если же нам надо увидеть все хранимые в теме сообщения, нужно начать считывать тему с самого начала:

**kafka-console-consumer --bootstrap-server localhost:9092 --topic topic\_<username> --from-beginning**

## Apache Spark

При упоминании об альтернативном IBM InfoSphere Streams инструменте обработки потоковых данных, по тексту работы, зачастую упоминается Apache Spark. Это действительно мощный инструмент обработки данных, но в ходе лабораторного практикума о нем не получится сказать многого по нескольким причинам. Во первых, Spark поддерживает несколько языков написание обработчиков, это Scala, Python и Java, и каждый из них во многом отличается и требует своего подхода. Во вторых, если организация конвейеров получения данных, их первичной сортировки и распределения поддается следованию за best practices, то конечная обработка и анализ данных это задачи почти индивидуальные для каждого случая. Если единообразие продуктов IBM позволяет сказать что для обработки данных географического положения стоит использовать конкретную библиотеку и функции, то в Spark этих вариантов несколько для каждого языка.

В ходе предыдущих лабораторных работ вы уже анализировали данные из файла news-data.txt, который расположен в вашей личной папке в hdfs. Проведем повторный анализ этого файла средствами Apache Spark. Поскольку примеры приведены для разных языков необходимо запускать соответствующую консоль Spark.

Для запуска Scala консоли Apache Spark введите:

**/usr/bin/spark-shell**

после успешного выполнения вы должны увидеть следующее уведомление

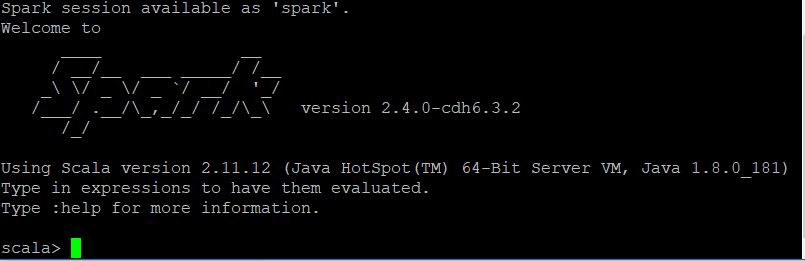


Рис. 12. Успешно запущенная консоль Spark Scala

Для запуска Python консоли Apache Spark введите:

**/usr/bin/pyspark**

После успешного выполнения вы должны увидеть следующее уведомление:



Рис. 13. Успешно запущенная консоль Spark Python

В зависимости от выбранного языка, выполните по очереди следующие команды для того чтобы считать данные, провести подсчет количества различных слов в файле и сохранить полученный результат.

Scala

**scala> val file = sc.textFile("/user/<username>/news-data.txt")**

**scala> val counts = file.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(\_ + \_)**

**scala> counts.saveAsTextFile("/user/<username>/spark\_output")**

Python

**>>> myfile = sc.textFile("hdfs://localhost:8020/user/<username>/news-data.txt")**

**>>> counts = myfile.flatMap(lambda line: line.split(" ")).map(lambda word: (word, 1)).reduceByKey(lambda v1,v2: v1 + v2)**

**>>> counts.saveAsTextFile("hdfs://localhost:8020/user//<username>/spark\_output")**

В результате выполнения вы должны получить набор из 3 файлов расположенный по пути /user/<username>/spark\_output.

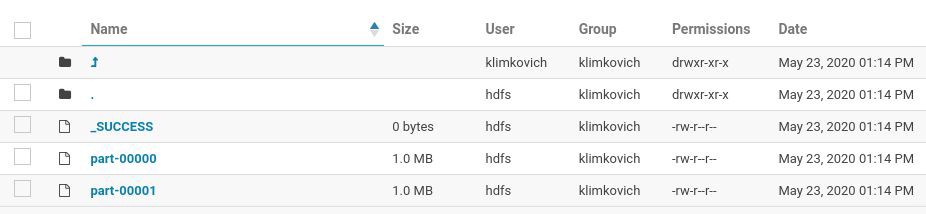


Рис. 14. Файлы записанные в результате выполнения команд Для вывода количества разных слов в консольном окне введите:

**counts.count()**

В результате изучения материала по работе с потоковыми данными средствами Apache Kafka и Apache Spark вы сможете создавать темы Kafka, получать и отправлять данные из этих тем, а также выполнять обработку данных средствами Apache Spark при чтении из файла или потока.

# Лабораторная работа №5. Исследование текстовой информации с помощью Apache Nutch и Apache Solr

## Цели и задачи лабораторной работы

Целью данной лабораторной работы является ознакомление с функционалом инструмента для сбора данных Apache Nutch и инструмента полнотекстового поиска Apache Solr для решения задачи исследования текстовой информации.

В ходе выполнения лабораторной работы обучающийся выполнит следующие задачи:

* настройка Nutch для сбора данных посредством работы с командной строкой;
* сбор данных с веб-сайтов с помощью Nutch;
* создание базы данных для хранения собранных данных, её обновление;
* индексация данных;
* импортирование данных в Apache Solr;
* ознакомление с интерфейсом Apache Solr:
* просмотр результатов в Apache Solr;
* выполнение текстового анализа.

## Сценарий лабораторной работы

1. Предварительная настройка инструмента Apache Nutch. На этом шаге будут созданы необходимые каталоги и файлы для дальнейшей работы с инструментом Apache Nutch, а также создан список источников для сбора данных и база данных для хранения собранных данных.
2. Создание выборки всех страниц, проведение синтаксического анализа собранных данных и импорт в базу данных.
3. Создание выборки топ-1000 страниц, проведение синтаксического анализа собранных данных и обновление базы данных.
4. Индексация и инвертирование данных.
5. Импортирование данных в Apache Solr и просмотр результатов.
6. Проведение анализа данных.

## Инструмент Apache Nutch

Apache Nutch — подключаемый модульный фреймворк для построения поисковых систем, веб-crawler. Роль Nutch заключается в сборе и индексации данных из сети Интернет с помощью алгоритмов веб-кроулинга.

Под веб-краулером понимается программное обеспечение, способное просматривать огромное количество веб-страниц, “сканировать” их содержимое, анализировать метаданные и собирать информацию по установленным фильтрам. Индексация собранных данных позволяет более быстро обращаться к ним повторно в дальнейшем.

Apache Nutch не имеет пользовательского интерфейса, и работа с ним возможна только посредством консоли, параметры и фильтры для сбора данных по URL-адресам настраиваются в конфигурационном файле. Дальнейшие обработка и анализ собранных данных, как правило, производятся при помощи других инструментов, в частности, Apache Solr, который входит в дистрибутив Cloudera. Помимо Apache Solr, Nutch поддерживает интеграцию с Elastic Search и SolrCloud.

Непосредственно сам фреймворк Nutch не входит в дистрибутив Cloudera, однако также является продуктом с открытым исходным кодом, лицензированным организацией Apache Software Foundation, и по умолчанию поддерживает работу с Apache Solr, что существенно упрощает их интеграцию и способствует только наращиванию функциональности дистрибутива Cloudera.

Nutch содержит следующие ключевые составляющие:

1. База данных сканирования (crawldb). В ней содержится информация о всех URL, известных Nutch, в том числе о том, была ли проведена выборка по этим адресам и когда;
2. База данных ссылок (linkdb). Она содержит список известных ссылок на каждый URL, исходный URL и его метки (anchor text);
3. Набор сегментов. Каждый сегмент представляет собой набор URL, которые выбираются как единое целое. Сегменты - это каталоги со следующими подкаталогами:
   * crawl\_generate - набор URL для сканирования;
   * crawl\_fetch содержит статус выборки каждого URL;
   * content содержит необработанные данные, полученные из каждого URL;
   * parse\_text содержит проанализированный текст каждого URL;
   * parse\_data содержит исходящие ссылки и метаданные, проанализированные с каждого URL;
   * crawl\_parse содержит URL-адреса исходящих ссылок, используемые для обновления crawldb.

## Инструмент Apache Solr

Apache Solr — инструмент полнотекстового поиска с широким спектром возможностей, таких как возможность интеграции с базами данных, поиска в источниках различного формата, таких как doc, docx, ppt, pptx, xls, xlsx, pdf, XML, JSON, CSV, подсвечивания результатов, фасетного поиска и динамической кластеризации. Apache Solr является мощным инструментом, способным обрабатывать большие объёмы данных и производить индексацию практически в режиме реального времени. Solr обладает довольно простым графическим интерфейсом и позволяет составлять запросы сразу на его странице. Главная страница Solr показана на Рис. 1.

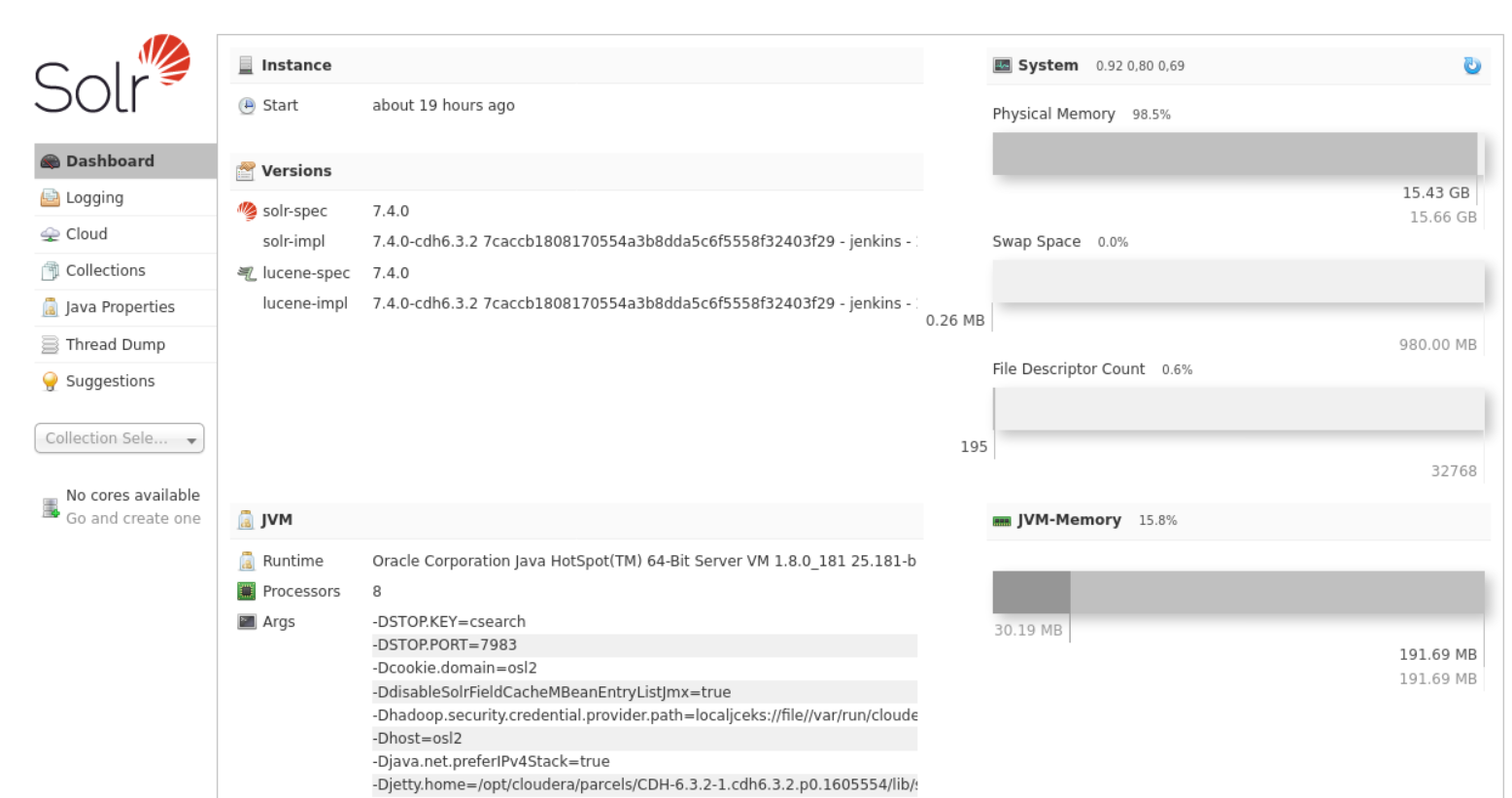


Рис. 1. Главная страница Apache Solr

Основная панель инструментов расположена слева, справа расположена служебная информация, такая как установленная версия и состояние системы.

Также возможности Solr могут быть расширены при помощи плагинов, что даёт возможность, к примеру, обрабатывать данные практически на любом языке и из источников практически любого формата.

Apache Nutch в связке с Apache Solr представляют собой мощный инструмент для полнотекстового поиска и анализа структурированных и неструктурированных данных большого объёма.

## Подготовка к лабораторной работе

Перед началом выполнения лабораторной работы убедитесь, что инструмент Apache Solr присутствует на главной странице Cloudera Manager:

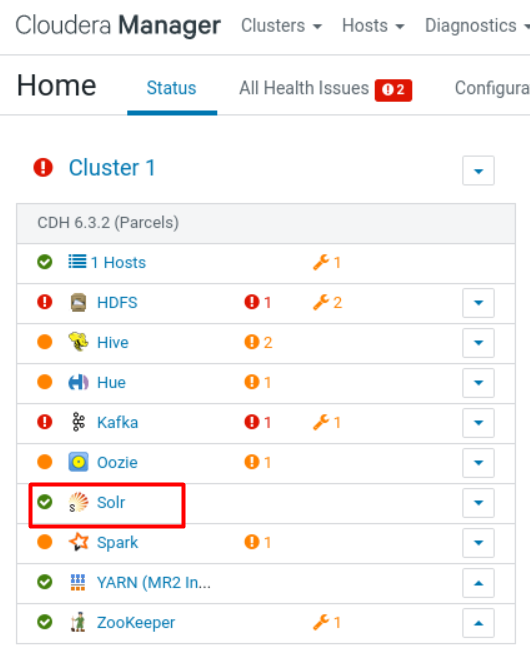


Рис. 2. Список сервисов Cloudera Manager

Если Apache Solr отсутствует в списке сервисов, добавьте его, нажав на выпадающий список рядом с заголовком **Cluster 1**, выберите **Add Service**, затем выберите сервис Apache Solr из открывшегося списка (Рис. 3):

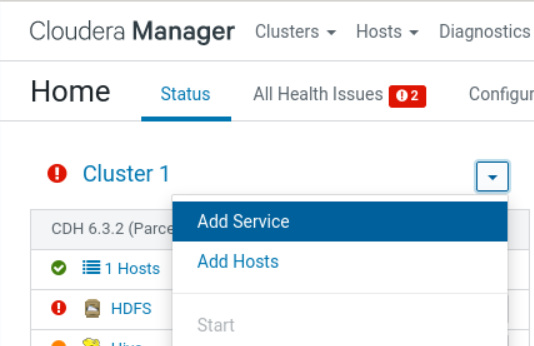


Рис. 3. Добавление нового сервиса

## Сбор и индексация данных

Задача сбора данных с веб-сайтов (веб-краулинга) решается при помощи инструмента Apache Nutch. Поскольку инструмент не имеет графического интерфейса, взаимодействие с ним осуществляется посредством командной строки.

Перед началом сбора данных следует корректно задать параметры Nutch в файле nutch-site.xml:

1. Перейдите в каталог **conf**:

cd /opt/miem\_nutch/1.16/conf

1. Откройте файл **nutch-site.xml** для редактирования:

nano nutch-site.xml

1. Добавьте следующие строки так, чтобы файл конфигурации имел вид как на Рис. 4. При желании вы можете поменять имя агента в атрибуте **value** параметра **http.agent.name**. В примере заданное имя - MIEM Nutch Spider.

<property>

<name>plugin.includes</name>

<value>protocol-http|urlfilter-(regex|validator)|parse-(html|tika)|index-(basic|anchor|more)|indexer-solr|scoring-opic|urlnormalizer-(pass|regex|basic)</value>

</property>



Рис. 4. Настройка файла конфигурации

1. фСохраните изменения сочетанием клавиш **ctrl-O** и нажмите **Enter** для подтверждения изменений.
2. Нажмите **ctrl-X**, чтобы выйти из режима редактирования.

Следующим шагом необходимо непосредственно задать список url-адресов, которые будет изучать Nutch. Для этого выполните следующие шаги:

1. Перейдите в директорию, где установлен Nutch, создайте папку **urls\_<username>** и файл **seed.txt**, выполнив следующие команды:

cd /opt/miem\_nutch/1.16

sudo mkdir urls\_<username>

cd urls\_<username>

sudo nano seed.txt

1. В открывшемся редакторе введите url-адреса, при этом каждый адрес должен находиться на новой строке. В примере для простоты будет использоваться только один адрес (Рис. 5):

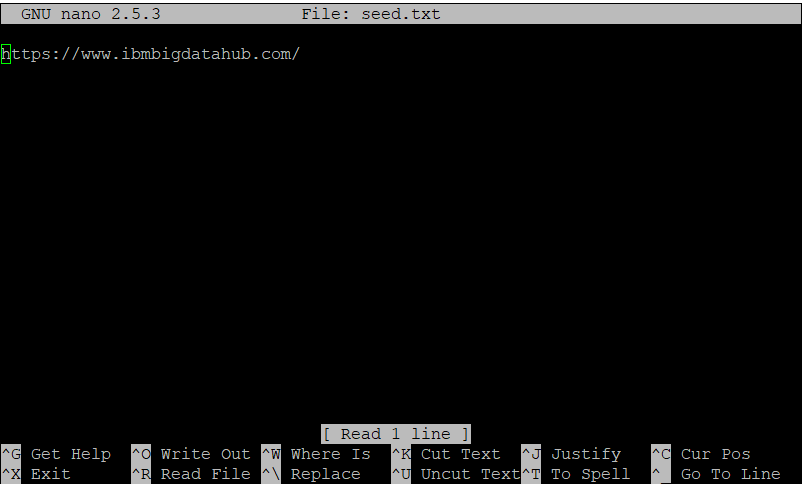


Рис. 5. Файл со списком URL

1. Сохраните изменения, нажав **ctrl-O**, затем нажмите **Enter** для подтверждения изменений.
2. Нажмите **ctrl-X**, чтобы выйти из режима редактирования.

В результате этих шагов был создан файл **seed.txt**, из которого Nutch будет считывать url-адреса для изучения.

Далее необходимо создать базу данных, в которой Nutch будет хранить известные ему адреса, и связать её с источником данных - ранее созданной папкой **urls\_<username>**:

cd /opt/miem\_nutch/1.16

bin/nutch inject crawl/crawldb\_<username> urls\_<username>

Для инициализации выборки данных заданных источников, выполните:

1. Сгенерируйте список выборки из созданной базы данных:

bin/nutch generate crawl/crawldb\_<username> crawl/segments\_<username>

1. В результате выполнения команды на предыдущем шаге был создан список выборки для всех страниц. Список выборок помещается в каталог **segments\_<username>**. Папки с выборками по умолчанию имеют имя по дате их создания. Для удобства обращения к ним сохраните имя папки в переменной s1:

s1=`ls -d crawl/segments\_<username>/2\* | tail -1`

echo $s1

1. Запустите процесс выборки, обращаясь к созданной переменной:

bin/nutch fetch $s1

1. Выполните синтаксический анализ полученных данных:

bin/nutch parse $s1

1. Обновите базу данных:

bin/nutch updatedb crawl/crawldb\_<username> $s1

Теперь база данных содержит как обновленные записи для всех начальных страниц, так и новые записи, которые соответствуют обнаруженным страницам, связанным с начальным набором.

Шаги выше являются типовыми и выполняются каждый раз, когда необходимо осуществить сбор данных.

Выборку возможно проводить повторно для уже изученных страниц:

1. Сгенерируйте список выборки для топ-1000 страниц:

bin/nutch generate crawl/crawldb\_<username> crawl/segments\_<username> -topN 1000

1. Сохраните имя новой папки с выборкой в переменную s2:

s2=`ls -d crawl/segments/2\* | tail -1`

echo $s2

1. Запустите процесс выборки, обращаясь к созданной переменной:

bin/nutch fetch $s2

1. Выполните синтаксический анализ полученных данных:

bin/nutch parse $s2

1. Обновите базу данных:

bin/nutch updatedb crawl/crawldb\_<username> $s2

В результате выполнения каждого “цикла” Nutch изучает данные на уровень глубже предыдущего, основываясь на уже имеющихся в его базе адресах, и пополняет свою базу данных новыми адресами.

Инвертируйте полученные данные, чтобы их можно было индексировать:

bin/nutch invertlinks crawl/linkdb\_<username> -dir crawl/segments\_<username>

Проиндексируйте данные:

bin/nutch index crawl/crawldb\_<username>/ -linkdb crawl/linkdb\_<username>/ crawl/segments\_<username>/\* -filter -normalize -deleteGone

Для того, чтобы данные импортировались в Solr, выполните следующие команды:

cp/home/lyovin/nutch/src/apache-nutch-1.16/src/plugin/indexer-solr/schema.xml $HOME/my\_nutch\_solr\_configs/conf

rm $HOME/my\_nutch\_solr\_configs/conf/managed-schema

solrctl instancedir —create

my\_nutch\_solr\_core $HOME/my\_nutch\_solr\_configs

solrctl collection —create miem\_nutch\_solr -s 1

solrctl core —create miem\_nutch\_solr\_core -p collection=miem\_nutch\_solr

Проиндексированные данные можно увидеть в Apache Solr.

Попасть в Solr можно двумя методами:

1. Через Cloudera Manager:
   1. Выберите Solr из списка элементов на стартовой странице Cloudera Manager (Рис. 6).

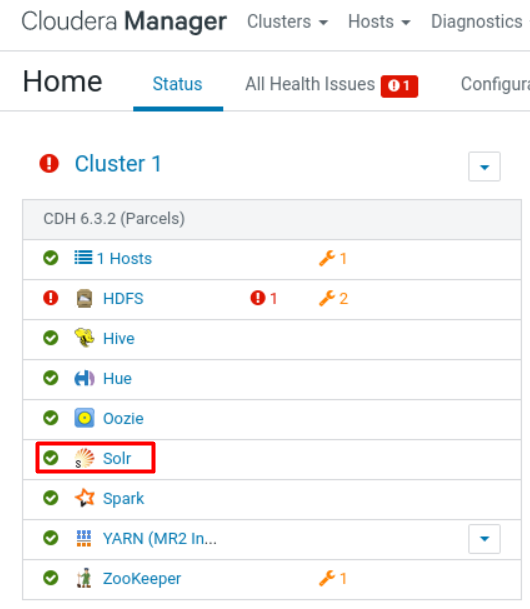


Рис. 6. Выбор Solr из списка элементов

* 1. Перейдите на вкладку **Instances** → **Solr Server** (Рис. 7).

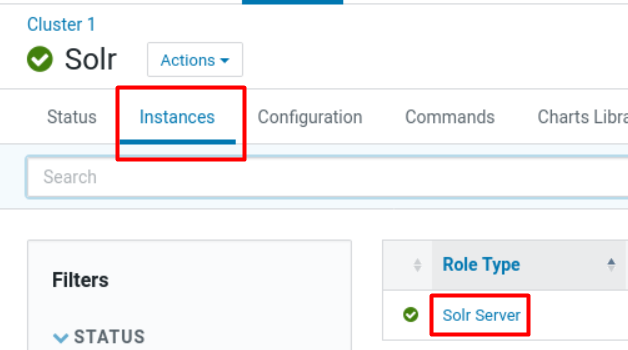


Рис. 7. Процесс перехода на домашнюю страницу Solr из Cloudera Manager

* 1. На открывшейся странице раскройте список **Actions** и выберите **Solr Server Web UI** (Рис. 8).

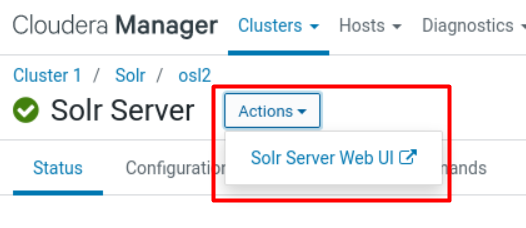


Рис. 8. Процесс перехода на домашнюю страницу Solr из Cloudera Manager

1. Путём ввода **localhost:8983/solr** в адресной строке браузера.

Выберите слева в выпадающем списке **Collection** коллекцию, перейдите в раздел **Query** и нажмите **Execute Query** внизу страницы, оставив предложенные параметры выборки без изменений. Справа отобразятся все проиндексированные страницы: их URL-адреса, дата выборки, заголовок, метаданные (Рис. 9).

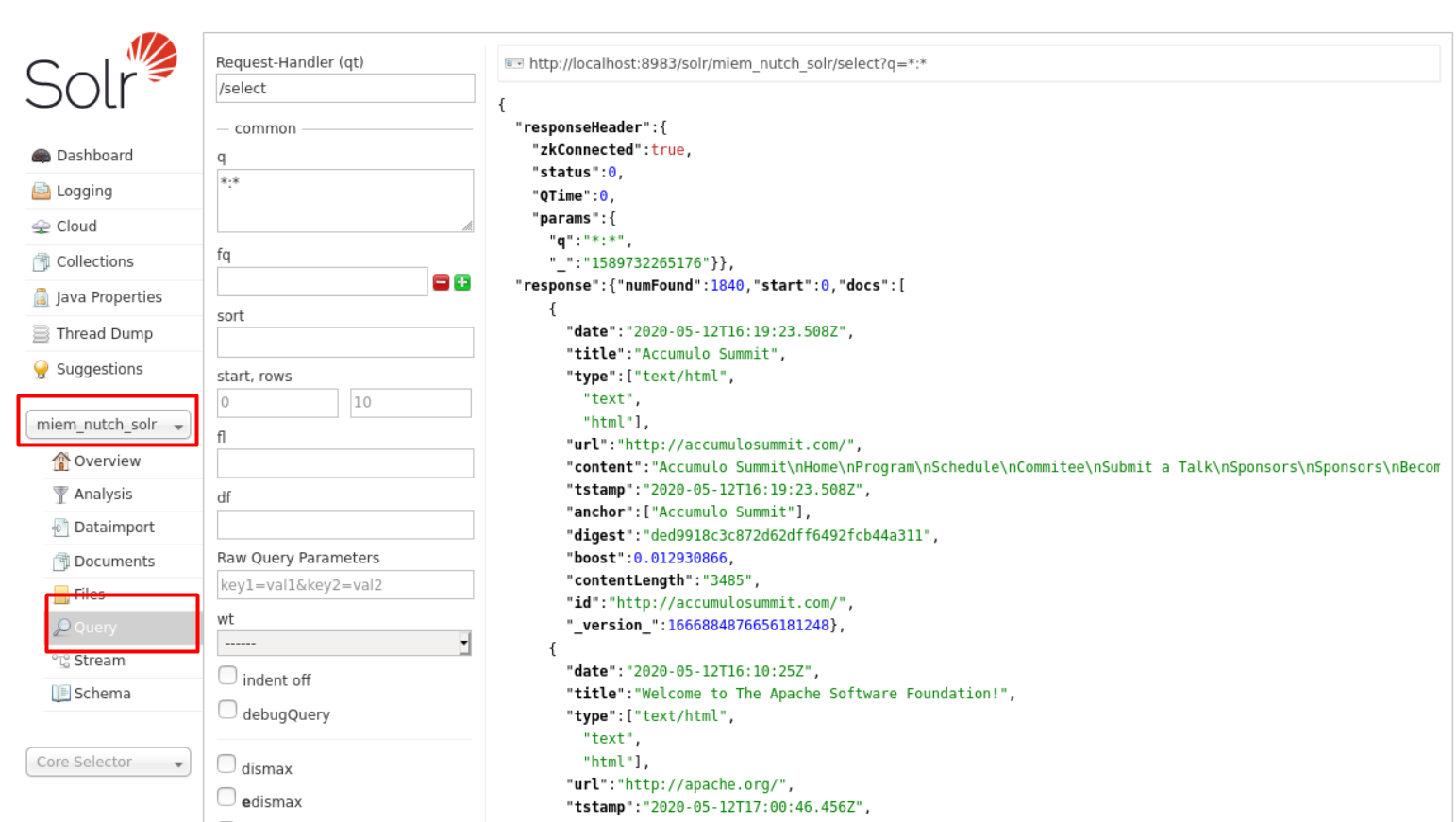


Рис. 9. Отображение данных в Apache Solr

Поиск и формирование условий поиска осуществляется путем составления запросов в форме слева. Попробуйте разные параметры и изучите выведенные результаты.

## Текстовый анализ

Простейший сравнительный анализ текста можно провести в Apache Solr без выполнения предварительных настроек. Для этого выполните следующие шаги:

1. Перейдите во вкладку **Analysis** (Рис. 10):

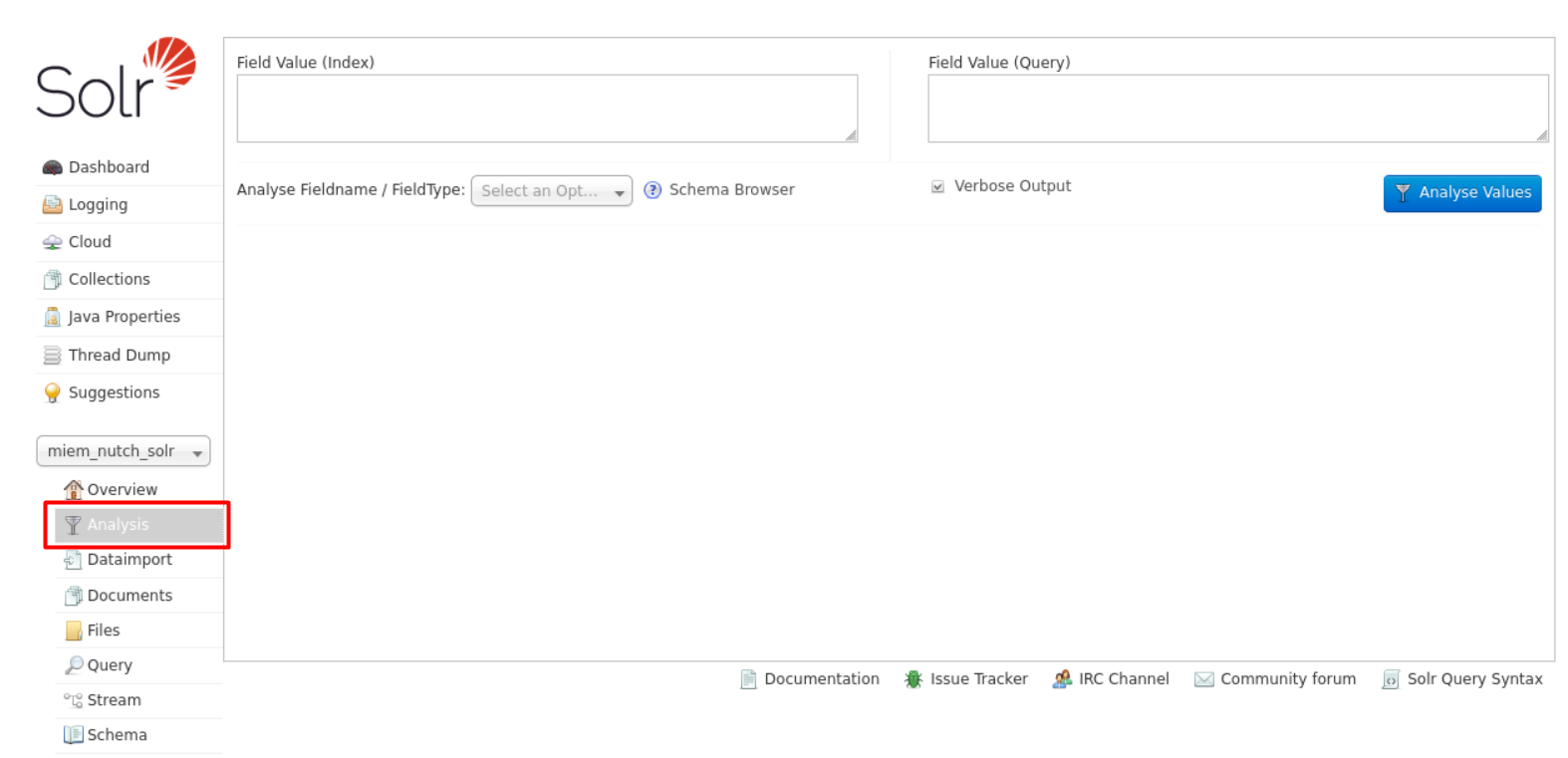


Рис. 10. Просмотр встроенного инструмента анализа текста

1. Введите в поля слева и справа текст, который будет проанализирован (Рис. 11):

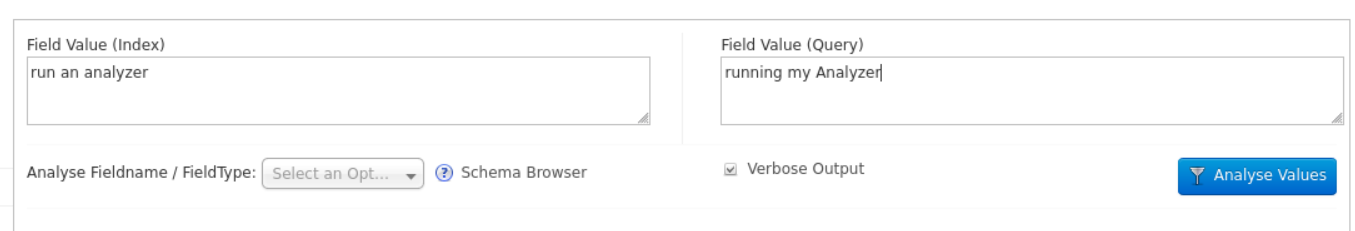


Рис. 11. Ввод данных для анализа

1. В **Analyse Fieldname / FieldType** выберите тип поля **text\_en** и нажмите **Analyse Values** (Рис. 12):

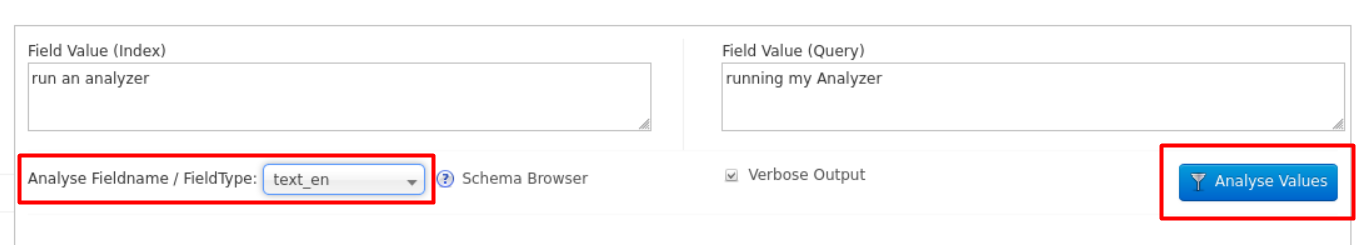


Рис. 12. Запуск анализатора

Изучив результат, можно увидеть, что алгоритмы Solr сделали вывод о том, что для входных данных из левой формы: исходное слово для analyzer - analyz, артикль an несущественен для анализа, для входных данных из правой формы: исходное слово для running - run, Analyzer - analyz. Меняя типы полей, можно получить разные результаты анализа. Попробуйте другие варианты входных данных и пронаблюдайте за результатом.