**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

по дисциплине **«ИТРОД**»

на тему: «**Анализ проведенной работы»**

Выполнил: студенты гр. ИП-41

Сачек Д.Н

Маланичева А.С.

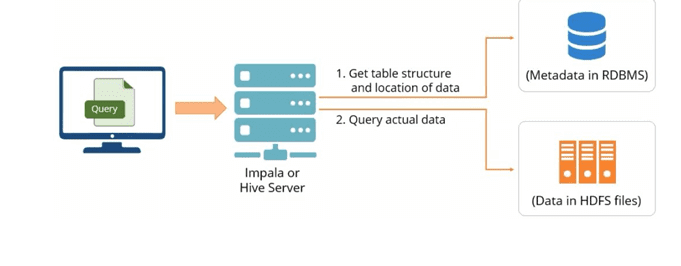
Принял: преподаватель

Шибеко В.Н.

Гомель 2020

1. Обеспечение масштабируемости и живучести, согласованности.

В Hive масштабируемость обеспечивается благодаря кластеру, состоящему из множества Data node, и курирующего их Name node.

Сам Hive Server использует отдельный Node для хранения метаданным о структуре БД. А сами данные хранятся на Data Node. ****

Живучесть обеспечивается механизмом репликации. Name Node постоянно следит за состоянием Data Node и в случае выхода из строя некоторой машины запускает механизм репликации данных.

Из вышесказанного следует и что за согласованность данных отвечает также Name Node.

1. Менеджмент CRUD операций.

Hive определяет простой [SQL](https://ru.bmstu.wiki/SQL_(Structured_Query_Language))-подобный язык запросов, называемый HiveQL (Hive Query Language), что позволяет пользователям, знакомым с [SQL](https://ru.bmstu.wiki/SQL_(Structured_Query_Language)) легко работать с запросами к данным. HiveQL имеет функции для работы с форматами XML и JSON, поддержку нескалярных типов данных, таких как массивы, структуры, ассоциативные массивы, поддерживает широкий набор агрегирующих функций, определяемые пользователем функции (User Defined Functions), блокировки. В то же время, этот язык позволяет программистам, которые знакомы со структурой MapReduce, подключить свои пользовательских карты для выполнения более сложного анализа, который может быть не поддерживаемым встроенными возможностями языка. HiveQL может быть расширен с пользовательскими скалярных функций (UDF), агрегаций (UDAF годов), и табличных функций (UDTF).

Однако, по-умолчанию, для поддержки операций обновления и удаления необходима дополнительная настройка и определенная структура таблиц.

1. Применение hadoop для big data

Используется для реализации поисковых и контекстных механизмов многих высоконагруженных веб-сайтов, в том числе, для [Yahoo!](https://ru.wikipedia.org/wiki/Yahoo!) и [Facebook](https://ru.wikipedia.org/wiki/Facebook)[[3]](https://ru.wikipedia.org/wiki/Hadoop#cite_note-_a27d67d55a679a1c-3). Разработан на [Java](https://ru.wikipedia.org/wiki/Java) в рамках вычислительной парадигмы [MapReduce](https://ru.wikipedia.org/wiki/MapReduce), согласно которой приложение разделяется на большое количество одинаковых элементарных заданий, выполнимых на узлах кластера и естественным образом сводимых в конечный результат.

По состоянию на 2014 год, проект состоит из четырёх модулей — Hadoop Common[[⇨]](https://ru.wikipedia.org/wiki/Hadoop#Hadoop_Common) ([связующее программное обеспечение](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D1%83%D1%8E%D1%89%D0%B5%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) — набор инфраструктурных программных библиотек и утилит, используемых для других модулей и родственных проектов), HDFS[[⇨]](https://ru.wikipedia.org/wiki/Hadoop#HDFS) ([распределённая файловая система](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D0%B0%D1%8F_%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)), YARN[[⇨]](https://ru.wikipedia.org/wiki/Hadoop#YARN) (система для планирования заданий и управления кластером) и Hadoop MapReduce[[⇨]](https://ru.wikipedia.org/wiki/Hadoop#Hadoop_MapReduce) (платформа программирования и выполнения распределённых MapReduce-вычислений), ранее в Hadoop входил целый ряд других проектов, ставших самостоятельными в рамках системы проектов Apache Software Foundation.

1. Размещение в Docker

Кластер развертывается с помощью docker-compose, начальные настройки хранятся в hadoop-hive.env. После развертывания расширение кластера возможно с помощью изменения docker-compose файла.

Достоинства:

Достоинства этого подхода, этой платформы. Очень хорошая и гладкая масштабируемость. Т.е. если нам нужно обработать в два раза больше данных, или хранить в два раза больше данных, нам достаточно добавить в два раза больше машин в кластер. Т.е. не совсем ровно в два раза, но почти в два раза.

Нулевая стоимость software. Можно взять кластер и использовать open-source Hadoop, как систему анализа и хранения данных.

Недостатки:

Во-первых, это довольно высокая стоимость поддержки. Если у вас есть Hadoop-кластер, из многих машин, вам нужно найти умного системного администратора, который разберется в архитектуре Hadoop-а, в том, как это работает, и будет все это поддерживать. Т.е. это действительно непросто, это действительно занимает много времени.

Вам нужны будут люди, которые будут придумывать бизнес-часть, какие именно данные им нужны, и вам будет нужна команда Java-разработчиков, которые будут писать эти map-reduce job.

Проблема с real-time-ом. Hadoop это не real-time система. Если вы хотите получать какие-то данные, у вас не получится так, что вы будете запускать map-reduce job-ы, когда пользователь заходит на сайт. Вам нужно обновлять данные, хоть раз час, в background-е, а пользователю показывать уже рассчитанные данные.

**Подробности реализации Hive**

[Apache Hive](https://hive.apache.org/) был придуман для того чтобы объединить два достоинства:

* Масштабируемость MapReduce
* Удобство использования SQL для выборок из данных.

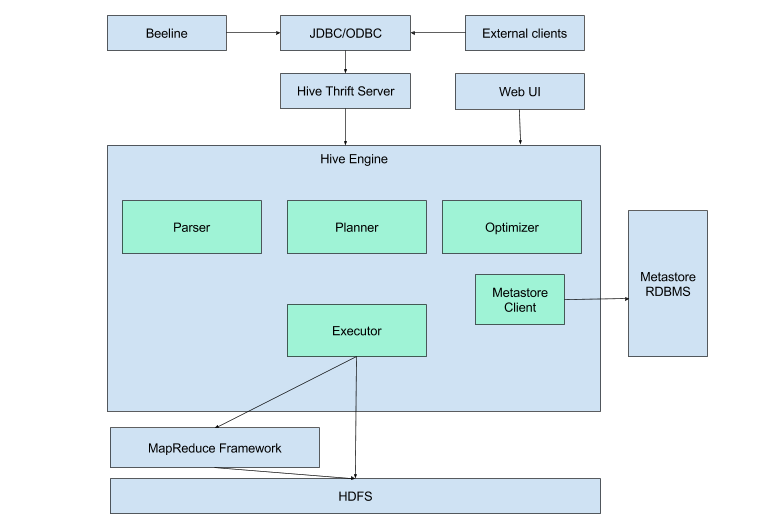
## Общее описание

Hive появился в недрах компании Facebook в 2007 году, а через год исходники hive были открыты и переданы под управление apache software foundation. Изначально hive представлял собой набор скриптов поверх hadoop streaming позже развился в полноценный фреймворк для выполнения запросов к данным поверх MapReduce.

Актуальная версия apache hive представляет собой продвинутый фреймворк, который может работать не только поверх фреймворка Map/Reduce, но и поверх [Spark](http://spark.apache.org/), а также [Apache Tez](https://tez.apache.org/).

Apache hive используют в production такие компании как Facebook, Grooveshark, Last.Fm и многие другие.

## Архитектура

****

Hive представляет из себя движок,  который превращает SQL-запросы в цепочки map-reduce задач.  Движок включает в себя такие компоненты, как Parser(разбирает входящие SQL-запрсоы), Optimimer(оптимизирует запрос для достижения большей эффективности), Planner (планирует задачи на выполнение) Executor(запускает задачи на фреймворке MapReduce.

Для работы hive также необходимо хранилище метаданных. Дело в том что SQL предполагает работу с такими объектами как база данных, таблица, колонки, строчки, ячейки и тд. Поскольку сами данные, которые использует hive хранятся просто в виде файлов на hdfs — необходимо где-то хранить соответствие между объектами hive и реальными файлами.  
  
В качестве metastorage используется обычная реляционная СУБД, такая как MySQL, PostgreSQL или Oracle.

## Data Units

При работе с hive можно выделить следующие объекты которыми оперирует hive:

1. База данных
2. Таблица
3. Партиция (partition)
4. Бакет (bucket)

Разберем каждый из них подробнее:

### **База данных**

**База данных** представляет аналог базы данных в реляционных СУБД. База данных представляет собой **пространство имён**, содержащее таблицы. Команда создания новой базы данных выглядит следующим образом:

****

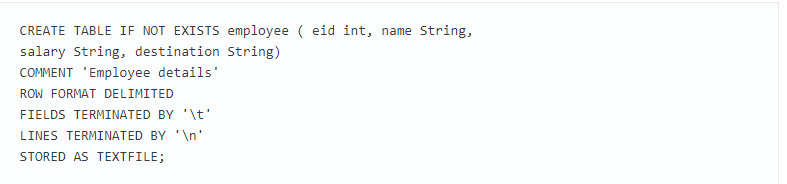
Database и Schema в данном контексте это одно и тоже. Необязательная добавка **IF NOT EXISTS** как не сложно догадаться создает базу данных только в том случае если она еще не существует.

### **Таблица**

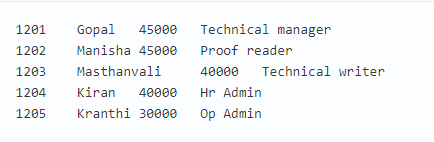
Таблица в hive представляет из себя аналог таблицы в классической реляционной БД. Основное отличие — что данные hive’овских таблиц хранятся просто в виде обычных файлов на hdfs. Это могут быть обычные текстовые csv-файлы, бинарные sequence-файлы, более сложные колоночные parquet-файлы и другие форматы. Но в любом случае данные, над которыми настроена hive-таблица очень легко прочитать и не из hive.

Таблицы в hive бывают двух видов:

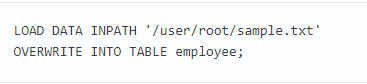
**Классическая таблица**, данные в которую добавляются при помощи hive. Вот пример создания такой таблицы

****

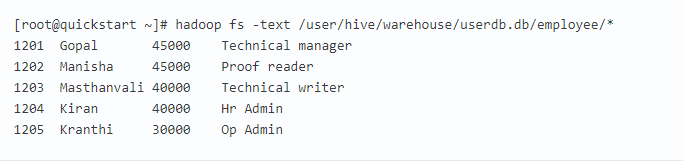
Тут мы создали таблицу, данные в которой будут храниться в виде обычных csv-файлов, колонки которой разделены символом табуляции.  После этого данные в таблицу можно загрузить. Пусть у нашего пользователя в домашней папке на hdfs есть файл sample.txt вида:

****

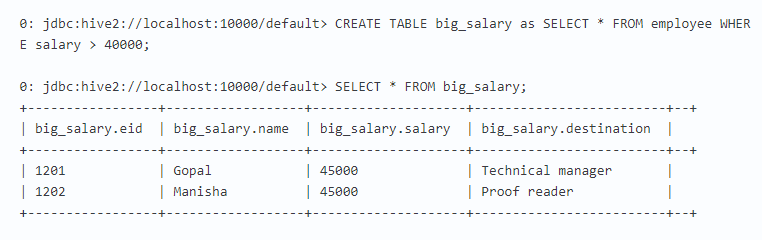
Загрузить данные мы сможем при помощи следующей команды:

****

После hive **переместит** данныe, хранящемся в нашем файле в хранилище hive. Убедиться в этом можно прочитав данные напрямую из файла в хранилище hive в hdfs:

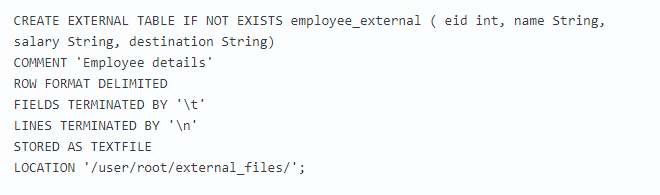
****

Классические таблицы можно также создавать как результат select-запроса к другим таблицам:

****

SELECT для создания таблицы в данном случае уже запустит mapreduce-задачу.

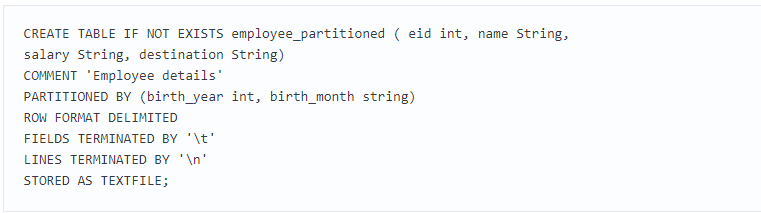
**Внешняя таблица**, данные в которую загружаются внешними системами, без участия hive. Для работы с  внешними таблицами при создании таблицы нужно указать ключевое слово **EXTERNAL**, а также указать путь до папки, по которому хранятся файлы:

****

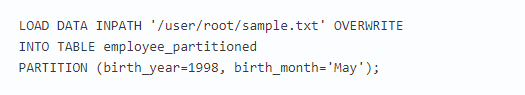
После этого таблицей можно пользоваться точно так же как и обычными таблицами hive. Самое удобное в этом, что вы можете просто скопировать файл в нужную папочку в hdfs, а hive будет автоматом подхватывать новые файлы при запросах к соответствующей таблице. Это очень удобно при работе например с логами.

### **Партиция (partition)**

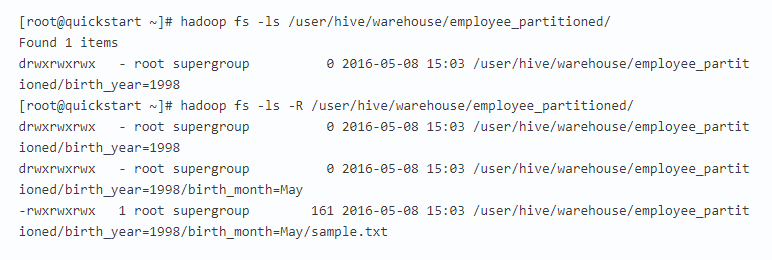
Так как hive представляет из себя движок для трансляции SQL-запросов в mapreduce-задачи, то обычно даже простейшие запросы к таблице приводят к полному сканированию данных в этой таблицы. Для того чтобы избежать полного сканирования данных по некоторым из колонок таблицы можно произвести партиционирование этой таблицы. Это означает, что данные относящиеся к разным значениям будут физически храниться в разных папках на HDFS.  
  
Для создания партиционированной таблицы необходимо указать по каким колонкам будет произведено партиционирование:

****

При заливке данных в такую таблицу необходимо явно указать, в какую партицию мы заливаем данные:

****

Посмотрим теперь как выглядит структура директорий:

****

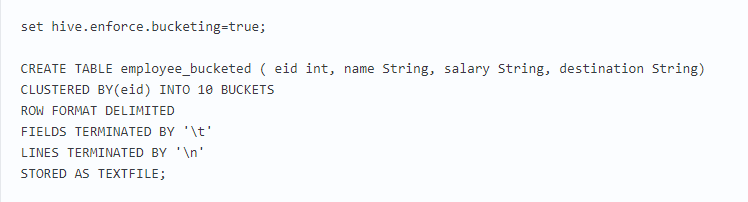
Видно, что структура директорий выглядит таким образом,  что каждой партиции соответствует отдельная папка на hdfs. Теперь, если мы будем запускать какие-либо запросы, у казав в условии WHERE ограничение на значения партиций — mapreduce возьмет входные данные только из соответствующих папок.  
  
В случае External таблиц партиционирование работает аналогичным образом, но подобную структуру директорий придется создавать вручную.  
  
Партиционирование очень удобно например для разделения логов по датам, так как правило любые запросы за статистикой содержат ограничение по датам. Это позволяет существенно сократить время запроса.

### **Бакет**

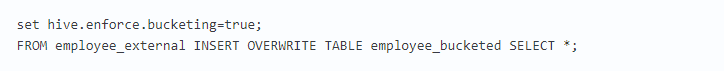
Партиционирование помогает сократить время обработки, если обычно при запросах известны ограничения на значения какого-либо столбца. Однако оно не всегда применимо. Например — если количество значений в столбце очень велико. Напрмер — это может быть ID пользователя в системе, содержащей несколько миллионов пользователей.  
  
В этом случае на помощь нам придет разделение таблицы на бакеты. В один бакет попадают строчки таблицы, для которых значение совпадает значение хэш-функции вычисленное по определенной колонке.  
  
При любой работе с бакетированными таблицами необходимо не забывать включать поддержку бакетов в hive (иначе hive будет работать с ними как с обычными таблицами):

****

Для создания таблицы разбитой на бакеты используется конструкция CLUSTERED BY

****

Так как команда Load используется для простого перемещения данных в хранилище hive — в данном случае для загрузки она не подходит, так как данные необходимо предобработать, правильно разбив их на бакеты. Поэтому их нужно загрузить при помощи команды INSERT из другой таблицы(например из внешней таблицы):

****

После выполнения команды убедимся, что данные действительно разбились на 10 частей:

****

Теперь при запросах за данными, относящимися к определенному пользователю, нам не нужно будет сканировать всю таблицу, а только 1/10 часть этой таблицы.

## Checklist по использованию hive

Теперь мы разобрали все объекты, которыми оперирует hive. После того как таблицы созданы — можно работать с ними, так как с таблицами обычных баз данных. Однако не стоит забывать о том что hive — это все же движок по запуску mapreduce задач над обычными файлами, и полноценной заменой классическим СУБД он не является. Необдуманное использование таких тяжелых команд, как JOIN может привести к очень долгим задачам. Поэтому прежде чем строить вашу архитектуру на основе hive — необходимо несколько раз подумать.  Приведем небольшой checklist по использованию hive:

* Данных которые надо обрабатывать много и они не влазят на диск одной машины (иначе лучше подумать над классическими SQL-системами).
* Данные в основном только добавляются и редко обновляются (если обновления часты — возможно стоит подумать об использовании Hbase например, см наш [предыдущий материал](https://habrahabr.ru/company/dca/blog/280700/).
* Данные имеют хорошо структурированную структуру и хорошо разбиваются на колонки.
* Паттерны обработки данных хорошо описываются декларативным языком запросов (SQL).
* Время ответа на запрос не критично(так как hive работает на основе MapReduce — интерактивности ждать не стоит).