

#04: Hive. Workshop.

1. Цель занятия.	1
2. Общие рекомендации.	1
3. Запускаем Hive.	2
4. Hive as a Shell.	2
5. Создание базы данных (Warehouse).	2
6. Создание Managed таблиц.	3
7. Пример создания Hive-таблицы (с пояснениями)	5
8. Создание таблиц на основе Regexp.	6
9. Задачи на Hive Query Language (HiveQL).	8
10. Обратная связь	8

1. Цель занятия.

1. Научиться запускать Hive в 3х различных режимах;
2. Научиться создавать баз данных в Hive;
3. Научиться создавать External и Managed таблицы;
4. Научиться создавать таблицы на основе регулярных выражений.

DISCLAIMER: Не надо беспокоиться, если у Вы что-либо не успели. Всегда остается возможность продолжить погружение дома и иметь возможность спрашивать вопросы в Telegram-канале.

2. Общие рекомендации.

Чтобы пробросить порты 8088, 50070 воспользуйтесь инструкцией из [User Guides](https://bigdatateam.org).

Для удобства необходимые материалы для копирования лежат в папке:
`/home/aadral/public_examples/hive`

3. Запускаем Hive.

Hive можно запускать в трех различных режимах:

1. Запуск в интерактивной оболочке:
 - `$ hive`
 - `hive> show databases;`
2. Запуск внешней команды:
 - `$ hive -e 'show databases;'`
3. Запуск команды из файла:
 - `$ echo 'show databases;' > sh_db.sql`
 - `$ hive -f sh_db.sql`

4. Hive as a Shell.

В hive можно запускать внешние (системные) команды:

```
hive> !nproc;
```

Можно удобно запускать hdfs команды:

```
hive> dfs -ls;
```

5. Создание базы данных (Warehouse).

Создадим личную базу данных для экспериментов:

```
hive> create database mf_<user> location '/user/mf_<user>/hive/warehouse';
```

Таким образом мы задали путь в HDFS, который будет использоваться по умолчанию для всех **MANAGED** таблиц, которые будем создавать в этой БД.

Важно: путь должен быть абсолютным.

Посмотреть описание базы данных можно с помощью команды DESCRIBE:

```
hive> describe database mf_<user>;
```

Если вы указали неверное название базы или LOCATION, базу можно удалить с помощью следующей команды:

```
hive> drop database if exists <database_name>;
```

Для решения ДЗ возьмите на заметку конструкцию "if exists"..

6. Создание Managed таблиц.

Возьмем данные, которые лежат по адресу:

```
virtual-client:/home/aadral/public_examples/hive/tab_delimited.txt
```

Это txt-файл, в котором находятся 3 колонки:

```
$ cat tab_delimited.txt
first line 1
second line 3
last line 5
\n \n 10
```

Первые две колонки - строковые, третья - числовая. Заметьте, каким образом помечается "NULL" значение для Hive. Давайте перенесем эти данные в вашу HDFS директорию:

```
$ hdfs dfs -mkdir hive_practice_data
$ hdfs dfs -put /home/aadral/public_examples/hive/tab_delimited.txt
/user/mf_<user>/hive_practice_data/
```

Создадим Managed Hive-таблицу поверх этих данных. Сделаем HiveQL скрипт (например create_tab_table.hql):

```
use mf_<user>;

drop table if exists tab_dataset;

create table tab_dataset (
    first_column string,
    second_column string,
    value int
) location '/user/mf_<user>/hive_practice_data/';
```

Запустим скрипт:

```
$ hive -f create_tab_table.hql
```



Теперь посмотрим на содержимое таблицы:

```
$ hive -e "use mf_<user>; select * from tab_dataset;"
```

Вопрос: что получилось на выходе? Почему?

Размышления предлагается обсудить в группе.

Обновите, определение таблицы (DDL), добавив две строчки (вынесено в gist, чтобы не спойлерить ответ на предыдущий вопрос):

<https://gist.github.com/aadral/b319bda28f2c5049b0aa3f5f6c5ad412>

После обновления скрипта давайте запустим его еще раз:

```
$ hive -f create_tab_table.hql
```

Посмотрим на данные с помощью Hive еще раз:

```
$ hive -e "use mf_<user>; select * from tab_dataset;"
```

Вопрос: что получилось на выходе? Почему?

комментарии - на следующей странице..



По результатам входного тестирования **70%** слушателей группы могли произвести такую операцию на боевом кластере. Надеюсь, что вы никогда больше не повторите такую ошибку и будете всегда с умом использовать MANAGED и EXTERNAL таблицы.

Задание:

1. Скопируйте данные из локальной папки в HDFS еще раз;
2. Обновите HiveQL скрипт создания tab_dataset таблицы, только теперь создайте EXTERNAL таблицу;
3. Посмотрите правильно ли распарсились все колонки;
4. Выведите "DESCRIBE FORMATTED tab_dataset" и посмотрите какая важная метainформация по таблице существует.

7. Пример создания Hive-таблицы (с пояснениями)

Для исходных данных воспользуемся датасетом subnets ("подсети"):
/data/subnets/variant1

Датасет содержит 2 колонки:

1. IP-адрес;
2. Маска подсети, в которой он находится.

Создаем EXTERNAL Hive-таблицу поверх данных в HDFS (доступных нам на чтение):

```
USE mf_<user>;

DROP TABLE IF EXISTS subnets;

CREATE EXTERNAL TABLE subnets (
    ip STRING,
    mask STRING
)
ROW FORMAT DELIMITED
    FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
LOCATION '/data/subnets/variant1';
```

Пояснения:

- USE ... - подключение к базе данных. Без этой строки таблицы будут создаваться в базе "default". Также можно вместо USE использовать аргумент --database при запуске запроса.

- EXTERNAL - существует 2 типа таблиц: MANAGED и EXTERNAL. EXTERNAL таблицы работают с внешними данными не изменяя их, а MANAGED позволяют их изменять.
- STORED AS - здесь выбирается формат хранения таблицы. Для EXTERNAL таблиц формат должен совпадать с форматом хранения данных. Для MANAGED рекомендуется использовать сжатые форматы хранения (RCFile, ORC, AVRO и т.п.).

После этого выполните запрос и посмотрите содержимое таблицы:

```
select * from subnets limit 10;
```

8. Создание таблиц на основе Regexp.

Далее мы будем пользоваться регулярными выражениями для парсинга входных данных. Для тех, кто не знаком с регулярными выражениями (regexp) и для тех, кто знаком, рекомендуем пользоваться regexp checker'ом, доступным онлайн: <https://regex101.com/>

Рассмотрим пример простых Web-логов (/data/user_logs/user_logs_M/):

```
$ hdfs dfs -text /data/user_logs/user_logs_M/* | head -2
33.49.147.163 20140101014611 http://news.rambler.ru/3105700 378 431
Safari/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64;
Trident/5.0; .NET CLR 3.5.30729;) n
197.72.248.141 20140101020306 http://news.mail.ru/6344933 1412 203
Safari/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0;
.NET CLR 3.5.30729; .NET CLR 3.0.30729; n
```

Через "\t" заданы следующие поля:

- ip (строка)
- auth_unused (строка, не используется)
- auth_user (строка, часто пустая)
- request_time (строка, потом можно будет извлечь дату)
- request (строка)
- response_code (число, обфусцирован)
- size (число, размер страницы в байтах)
- user_agent (строка)

В данном случае, request_time представлен в машинно-читаемом виде, но когда разберетесь с этой задачей, попробуйте распарсить стандартный формат Apache Log:

```
127.0.0.1 - - [05/Feb/2012:17:11:55 +0000] "GET / HTTP/1.1" 200 140 "-"
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.19 (KHTML, like
Gecko) Chrome/18.0.1025.5 Safari/535.19"
```

Воспользуемся следующим скриптом:

```
ADD JAR /opt/cloudera/parcels/CDH/lib/hive/lib/hive-serde.jar;

USE mf_<user>;

DROP TABLE IF EXISTS ser_de_example;

CREATE EXTERNAL TABLE ser_de_example (
    ip STRING
)
ROW FORMAT
    serde 'org.apache.hadoop.hive.serde2.RegexSerDe'
    with serdeproperties (
        "input.regex" = "^((\\S*))\\t.*"
    )
STORED AS textfile
LOCATION '/data/user_logs/user_logs_M';

SELECT * FROM ser_de_example LIMIT 10;
```

Вопро: что получаем на выходе?

Как вы могли уже заметить, Hive требует двойного экранирования слешей внутри регулярных выражений:

```
\t --> \\t
\\w --> \\\w
\\S --> \\S
...
```

Задание: обновите регулярное выражение, чтобы распарсить следующие 4 поля (изменения выделены **желтым** цветом):

```
ADD JAR /opt/cloudera/parcels/CDH/lib/hive/lib/hive-serde.jar;

USE mf_<user>;

DROP TABLE IF EXISTS ser_de_example;

CREATE EXTERNAL TABLE ser_de_example (
    ip STRING,
    date STRING,
```



```
request STRING,  
response_code INT  
)  
ROW FORMAT  
  serde 'org.apache.hadoop.hive.serde2.RegexSerDe'  
  with serdeproperties (  
    "input.regex" = "^(\\S*)\\t.*"  
  )  
STORED AS textfile  
LOCATION '/data/user_logs/user_logs_M';  
  
SELECT * FROM ser_de_example LIMIT 10;
```

Важно:

1. Заметьте, что поля типа auth_unused пропущены;
2. Для поля response_code используется тип INT;
3. Вы можете использовать hive-contrib.jar вместо hive-serde.jar, но тогда - вы не сможете получить что-либо на выходе помимо STRING (как например INT для поля response_code), а также путь до класса RegexSerDe будет другим:
 - o org.apache.hadoop.hive.contrib.serde2.RegexSerDe

9. Задачи на Hive Query Language (HiveQL).

Задача 0. Посчитать количество различных масок подсети.

Задача 1. Посчитать количество адресов, имеющих маску 255.255.255.128.

Задача 2. Посчитать среднее количество адресов по маскам.

Задача 3. По каждой задаче выведите план запроса и посчитайте по нему кол-во MapReduce Job (см. команду "explain")

10. Обратная связь

Обратная связь: http://rebrand.ly/mf2019q2_feedback_04_hive

Просьба потратить 1-2 минут Вашего времени, чтобы поделиться впечатлением, описать что было понятно, а что непонятно. Мы учитываем рекомендации и имеем возможность переформатируем учебную программу под Ваши запросы.