BIGDATA TEAM

#W4L106: Hive DDL и RegexSerDe. Workshop.

1. Цель занятия **2**

2. Запуск Hive **2**

3. Hive в качестве shell **2**

4. Создание базы данных (Warehouse) **3**

5. Создание Managed таблиц **3**

6. Создание таблиц на основе Regex **5**

7. Задачи на Hive Query Language (HiveQL) **8**

8. Обратная связь **9**

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;
5. Научиться запускать простые HiveQL запросы и читать план запроса.

2. Запуск Hive

Hive можно запускать в трех различных режимах. Попробуйте каждый из них для выполнения простой команды - вывод доступных баз данных в Hive.

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.hql`
    - `$ hive -f sh_db.hql`

3. Hive в качестве shell

В hive можно запускать внешние (системные) команды:

```
hive> !nproc;
```

Команды hdfs интегрированы в консоль Hive и их можно удобно запускать следующим образом:

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

Все команды в Hive должны завершаться символом ‘;’. Форматирование на несколько строк необходимо для удобного чтения запросов другими членами команды.

4. Создание базы данных (Warehouse)

Создадим личную базу данных для экспериментов:

```
hive> create database <user> location '/user/<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 <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`”..

5. Создание Managed таблиц

Возьмите данные, которые лежат по адресу:

- github:big-data-team/big-data-course/../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 /path/to/tab_delimited.txt /user/<user>/hive_practice_data/
```

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

```
use <user>;

drop table if exists tab_dataset;

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

Запустим скрипт:

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

Теперь посмотрим на содержимое таблицы:

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

Вопрос: что получилось на выходе? Почему?
Размышления предлагается обсудить в группе (канал #hive)

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

- github:big-data-team/big-data-course/../hive/hive_delimiter.hql

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 <user>; select * from tab_dataset;"
```

Вопрос: что получилось на выходе? Почему?
комментарии - на следующей странице..

В среднем, по результатам входного тестирования обычно **70%** слушателей группы могут 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.

Коротко - клауза MANAGED означает, что это вы, кто ответственны за управление данными в HDFS, а не только создаете метаинформацию по имеющимся данных в HDFS. Поэтому при удалении таблицы из Hive все MANAGED данные из HDFS будут удалены тоже. MANAGED таблицы удобны для экспериментов, чтобы не забыть удалить временные данные и не забивать кластерное пространство. External таблицы удобны для таблиц поверх сырых данных, заливаемых в HDFS роботными процессами (например access логи сервисов).

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 метаинформация по таблице существует.

6. Создание таблиц на основе Regex

Далее мы будем пользоваться регулярными выражениями для парсинга входных данных. Для проверки регулярных выражений (regex) рекомендуем пользоваться доступным онлайн regex checker'ом: https://regex101.com/

FAQ: Где узнать больше про регулярные выражения?
В рамках курса Big Data Analysis на Coursera есть опциональная лекция с ликбезом по регулярным выражениям (в Python): • Regular Expressions, Likbez (10 min) В презентации даются ссылки на полезные ресурсы: Python "re": • https://docs.python.org/2/library/re.html • https://docs.python.org/2/howto/regex.html#regex-howto Про регулярные выражения:

- https://regexone.com/lesson/introduction_abcs
- https://regex101.com/

Рассмотрим пример простых Web-логов:

```
$ 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 /usr/local/hive/lib/hive-serde.jar;

USE <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;
```

Вопрос: 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 этим цветом):

```
ADD JAR /usr/local/hive/lib/hive-serde.jar;

USE <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.*"[1]
      )
STORED AS textfile
LOCATION '/data/user_logs/user_logs_M';

SELECT * FROM ser_de_example LIMIT 10;
```

[1] Эту конструкцию нужно обновить, чтобы распарсить ровно 4 поля

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 мы не рекомендуем так делать, поскольку тогда вы не сможете получить что-либо на выходе помимо STRING (как например INT для поля `response_code`), а также путь до класса RegexpSerDe будет другим:
    - `org.apache.hadoop.hive.contrib.serde2.RegexSerDe`

7. Задачи на Hive Query Language (HiveQL)

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 <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.).

Выполните запрос и проверьте содержимое таблицы:

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

Задача 1. Посчитать количество различных масок подсети.
Задача 2. Посчитать количество адресов, имеющих маску 255.255.255.128.
Задача 3. Посчитать среднее количество адресов по маскам.
Задача 4. По каждой задаче выведите план запроса и посчитайте по нему кол-во MapReduce Job (см. команду "explain")

8. Обратная связь

Обратная связь: https://rebrand.ly/mailbd2021q1_feedback_module

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.