如何在 Apache Hive 中解析 Json 数组

问题

我们都知道，Hive 内部提供了大量的内置函数用于处理各种类型的需求，参见官方文档：Hive Operators and User-Defined Functions (UDFs)。我们从这些内置的 UDF 可以看到两个用于解析 Json 的函数：get_json_object 和
json_tuple。用过这两个函数的同学肯定知道，其职能解析最普通的 Json 字符串，如下：

```
hive (default)>  SELECT get_json_object('{"website":"www.iteblog.com","name":"过往记忆"}', '$.website');
OK
www.iteblog.com

hive (default)> SELECT json_tuple('{"website":"www.iteblog.com","name":"过往记忆"}', 'website', 'name');
OK
www.iteblog.com 过往记忆
Time taken: 0.074 seconds, Fetched: 1 row(s)
```

json_tuple 相对于 get_json_object 的优势就是一次可以解析多个 Json 字段。但是如果我们有个 Json 数组，这两个函数都无法处理，get_json_object 处理 Json 数组的功能很有限，如下：

```
hive (default)>
            >
            > SELECT get_json_object('[{"website":"www.iteblog.com","name":"过往记忆"}, {"website":"carbondata.iteblog.com","name":"carbondata 中文文档"}]', '$.[0].website');
OK
www.iteblog.com
Time taken: 0.069 seconds, Fetched: 1 row(s)
```

如果我们想将整个 Json 数组里面的 website 字段都解析出来，如果这么写将非常麻烦，因为我们无法确定数组的长度，而且即使确定了，这么写可维护性也很差，所以我们需要想别的办法。

如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

使用 Hive 自带的函数解析 Json 数组

在介绍如何处理之前，我们先来了解下 Hive 内置的 explode 函数，官方的解释是：explode() takes in an array (or a map) as an input and outputs the elements of the array (map) as separate rows. UDTFs can be used in the SELECT expression list and as a part of LATERAL VIEW. 意思就是 explode() 接收一个 array 或 map 类型的数据作为输入，然后将 array 或 map 里面的元素按照每行的形式输出。其可以配合 LATERAL VIEW
一起使用。光看文字描述很不直观，咱们来看看几个例子吧。

```
hive (default)> select explode(array('A','B','C'));
OK
A
B
C
Time taken: 4.188 seconds, Fetched: 3 row(s)

hive (default)> select explode(map('A',10,'B',20,'C',30));
OK
A 10
B 20
C 30
```

相信不需要我描述大家就能看明白这个函数的意义。大家可能会问，这个函数和我们解析 Json 数组有毛关系啊。其实有关系，我们其实可以使用这个函数将 Json 数组里面的元素按照一行一行的形式输出。根据这些已有的知识，我们可以写出以下的 SQL 语句：

```
hive (default)> SELECT explode(split(regexp_replace(regexp_replace('[{"website":"www.iteblog.com","name":"过往记忆"},{"website":"carbondata.iteblog.com","name":"carbondata 中文文档"}]', '{','₩₩}₩₩;₩₩{'),'₩₩[|₩₩]',''),'₩₩;'));
OK
{"website":"www.iteblog.com","name":"过往记忆"}
{"website":"carbondata.iteblog.com","name":"carbondata 中文文档"}
```

现在我们已经能正确的解析 Json 数据了。

你现在肯定不知道上面一堆的 SQL 是啥含义，这里我来一步一步的解释。

- explode 函数只能接收数组或 map 类型的数据，而 split 函数生成的结果就是数组；
- 第一个 regexp_replace 的作用是将 Json 数组元素之间的逗号换成分号，所以使用完这个函数之后，[{"website":"www.iteblog.com","name":"过往记忆"},{"website":"carbondata.iteblog.com","name":"carbondata 中文文档"}] 会变成 [{"website":"www.iteblog.com","name":"过往记忆"};{"website":"carbondata.iteblog.com","name":"carbondata 中文文档"}]
- 第二个 regexp_replace 的作用是将 Json 数组两边的中括号去掉，所以使用完这个函数之后，[{"website":"www.iteblog.com","name":"过往记忆"},{"website":"carbondata.iteblog.com","name":"carbondata 中文文档"}] 会变成 {"website":"www.iteblog.com","name":"过往记忆"},{"website":"carbondata.iteblog.com","name":"carbondata 中文文档"}

然后我们可以结合 get_json_object 或 json_tuple 来解析里面的字段了：

```
hive (default)> select json_tuple(json, 'website', 'name') from (SELECT explode(split(regexp_replace(regexp_replace('[{"website":"www.iteblog.com","name":"过往记忆"},{"website":"carbondateblog.com","name":"carbondata 中文文档"}]', '₩₩}₩₩,₩₩{','₩₩}₩₩;₩₩{'),'₩₩[|₩₩]','
```

'),'₩₩;')) as json) iteblog;
OK
www.iteblog.com 过往记忆
carbondata.iteblog.com carbondata 中文文档
Time taken: 0.189 seconds, Fetched: 2 row(s)

自定义函数解析 Json 数组

虽然可以使用 Hive 自带的函数类解析 Json 数组，但是使用起来还是有些麻烦。值得高兴的是，Hive 提供了强大的自定义函数（UDF）的接口，我们可以使用这个功能来编写解析 Json 数组的 UDF。具体的代码如下：

```
package com.iteblog.udf.json;

import org.apache.hadoop.hive.ql.exec.Description;
import org.apache.hadoop.hive.ql.exec.UDF;
import org.json.JSONArray;
import org.json.JSONException;

import java.util.ArrayList;


@Description(name = "json_array",
       value = "_FUNC_(array_string) - Convert a string of a JSON-
encoded array to a Hive array of strings.")
public class UDFJsonAsArray extends UDF {
   public ArrayList<String> evaluate(String jsonString) {
      if (jsonString == null) {
         return null;
      }
      try {
         JSONArray extractObject = new JSONArray(jsonString);
         ArrayList<String> result = new ArrayList<String>();
         for (int ii = 0; ii < extractObject.length(); ++ii) {
            result.add(extractObject.get(ii).toString());
         }
         return result;
      } catch (JSONException e) {
         return null;
      } catch (NumberFormatException e) {
         return null;
      }
   }
```

```
}
```

上面的代码逻辑很简单，我就不介绍了。将上面的代码进行编译打包，假设打包完的 jar 包名称为 iteblog.jar，然后我们就可以如下使用这个函数了。

```
hive (default)> add jar /home/iteblog/iteblog.jar;
Added [/home/iteblog/iteblog.jar] to class path
Added resources: [/home/iteblog/iteblog.jar]
hive (default)> create temporary function json_array as 'com.iteblog.udf.json.UDFJsonAsArray';
OK
Time taken: 0.013 seconds
hive (default)>
            > select explode(json_array('[{"website":"www.iteblog.com","name":"过往记忆"},{"webs
ite":"carbondata.iteblog.com","name":"carbondata 中文文档"}]'));
OK
{"website":"www.iteblog.com","name":"过往记忆"}
{"website":"carbondata.iteblog.com","name":"carbondata 中文文档"}
Time taken: 0.08 seconds, Fetched: 2 row(s)

hive (default)> select json_tuple(json, 'website', 'name') from (SELECT explode(json_array('[{"we
bsite":"www.iteblog.com","name":"过往记忆"},{"website":"carbondata.iteblog.com","name":"ca
rbta 中文文档"}]')) as json) iteblog;
OK
www.iteblog.com 过往记忆
carbondata.iteblog.com  carbondata 中文文档
Time taken: 0.082 seconds, Fetched: 2 row(s)
```

这个结果和上面使用 Hive 内置的函数结果一致。当然，你还可以实现其他的 UDF，逻辑和这个类似，就不再介绍了。