在 Hive 中使用 OpenCSVSerde

OpenCSVSerde 使用

大家使用 Hive 分析数据的时候，CSV 格式的数据应该是很常见的，所以从 0.14.0 开始（参见 HIVE-7777）Hive 跟我们提供了原生的 OpenCSVSerde 来解析 CSV 格式的数据。从名字可以看出，OpenCSVSerde 是基于 Open-CSV 2.3 类库实现的，其解析 csv 的功能还是很强大的。

为了在 Hive 中使用这个 serde，我们需要在建表的时候指定 row format serde 为 org.apache.hadoop.hive.serde2.OpenCSVSerde，具体如下：

```
create external table test_open_csv_serde
(
   id         int,
   version    int,
   name       varchar(16),
   create_time date,
   status     timestamp,
   amount     decimal(9,2),
   approved   boolean,
   comment    varchar(40)
)
row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
with serdeproperties
(
   'separatorChar' = ',',
   'quoteChar'     = '₩"',
   'escapeChar'    = '₩₩'
)
location '/user/iteblog/test.db';
```

OpenCSVSerde 默认的分隔符(separator)、quote 以及逃逸字符（escape characters ）分别为 ₩、" 以及 '。这个可以解决我们读写 CSV 的需求。

OpenCSVSerde 的问题

如果我们查看表结构的时候，我们会发现如果 row format serde 为 org.apache.hadoop.hive.serde2.OpenCSVSerde，不管你建表的时候指定字段是什么类型，其显示的都是 string 类型：

```
hive> show create table test_open_csv_serde;
OK
CREATE EXTERNAL TABLE `test_open_csv_serde`(
  `id` string COMMENT 'from deserializer',
  `version` string COMMENT 'from deserializer',
  `name` string COMMENT 'from deserializer',
  `create_time` string COMMENT 'from deserializer',
  `status` string COMMENT 'from deserializer',
  `amount` string COMMENT 'from deserializer',
  `approved` string COMMENT 'from deserializer',
  `comment` string COMMENT 'from deserializer')
ROW FORMAT SERDE
  'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  'escapeChar'='₩₩',
  'quoteChar'='₩"',
  'separatorChar'=',')
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  'file:/data/datasets'
TBLPROPERTIES (
  'COLUMN_STATS_ACCURATE'='false',
  'numFiles'='0',
  'numRows'='-1',
  'rawDataSize'='-1',
  'totalSize'='0',
  'transient_lastDdlTime'='1587115388')
Time taken: 0.051 seconds, Fetched: 28 row(s)
```

但是我们到 Hive 的元数据表里面发现字段类型是按照我们建表的时候存储的：

```
mysql> select * from COLUMNS_V2 where CD_ID = 16 order by INTEGER_IDX;
+-------+---------+-------------+-------------+-------------+
| CD_ID | COMMENT | COLUMN_NAME | TYPE_NAME   | INTEGER_IDX |
+-------+---------+-------------+-------------+-------------+
|    16 | NULL    | id          | int         |           0 |
|    16 | NULL    | version     | int         |           1 |
|    16 | NULL    | name        | varchar(16) |           2 |
|    16 | NULL    | create_time | date        |           3 |
|    16 | NULL    | status      | timestamp   |           4 |
```

```
|    16 | NULL    | amount      | decimal(9,2) |          5 |
|    16 | NULL    | approved    | boolean      |          6 |
|    16 | NULL    | comment     | varchar(40)  |          7 |
+-------+---------+-------------+--------------+------------+
8 rows in set (0.00 sec)
```

也就是说，这个行为是 OpenCSVSerde 导致的，早在2016年05月 Hive 社区就有人反馈了支持其他类型的列 HIVE-13709，不过到现在还没解决。另外，除了上面说的 show create table 的时候显示字段为 string 类型，我们在使用 OpenCSVSerde 读写数据的时候也是按照 string 处理的。

其实，按我的理解，软件不应该什么设计，如果只支持 string 类型，那你为什么不在建表的时候就不让大家指定除 string 类型之外的类型。现在这种行为确实不太好，修改了用户默认的行为。

为什么使用 OpenCSVSerde 时，show 的时候字段全变成 string 类型

下面我们来看看是哪里导致使用 OpenCSVSerde 时，show 的时候字段全变成 string 类型。org.apache.hadoop.hive.ql.exec.DDLTask#showCreateTable 里面会调用 org.apache.hadoop.hive.ql.metadata.Table#getCols 方法，这里的 getDeserializer() 调用会根据我们建表时候指定的 row format serde 来创建对应的 Deserializer，因为 test_open_csv_serde 表的 serde 是 OpenCSVSerde，所以在初始化 Deserializer 的时候会调用 org.apache.hadoop.hive.serde2.OpenCSVSerde#initialize 方法，这里面会把表的所有列类型设置为 PrimitiveObjectInspectorFactory.javaStringObjectInspector，这个就直接导致后面我们 show create table 的时候显示所有字段为 string。紧接着用 columnOIs 去初始化 inspector 对象。

```
@Override
public void initialize(final Configuration conf, final Properties tbl) throws SerDeException {

   final List<String> columnNames = Arrays.asList(tbl.getProperty(serdeConstants.LIST_COLUM
NS)
       .split(","));

   numCols = columnNames.size();

   final List<ObjectInspector> columnOIs = new ArrayList<ObjectInspector>(numCols);

   for (int i = 0; i < numCols; i++) {
     columnOIs.add(PrimitiveObjectInspectorFactory.javaStringObjectInspector);
   }

   inspector = ObjectInspectorFactory.getStandardStructObjectInspector(columnNames, colu
```

```
mnOIs);
    outputFields = new String[numCols];
    row = new ArrayList<String>(numCols);

    for (int i = 0; i < numCols; i++) {
      row.add(null);
    }

    separatorChar = getProperty(tbl, SEPARATORCHAR, CSVWriter.DEFAULT_SEPARATOR);
    quoteChar = getProperty(tbl, QUOTECHAR, CSVWriter.DEFAULT_QUOTE_CHARACTER);
    escapeChar = getProperty(tbl, ESCAPECHAR, CSVWriter.DEFAULT_ESCAPE_CHARACTER);
}
```

初始化完 OpenCSVSerde 之后，getCols 方法会调用 org.apache.hadoop.hive.metastore.MetaStoreUtils#getFieldsFromDeserializer 方法，这个方法第一行就是 ObjectInspector oi = deserializer.getObjectInspector();，也就是获取我们上面介绍的 OpenCSVSerde 的 inspector：后面用这个解析字段类型的时候就直接拿到所有字段为 string。

有什么好办法？

那我们不想这么处理咋办？一个好办法肯定是自己实现一个 CSVSerde，然后展现出数据的真实数据类型，但这个估计比较麻烦。其实我们可以参考下 aws 的产品处理原则

- 如果数据包含使用双引号 (") 括起的值，则可以使用 OpenCSV SerDe 将这些值反序列化。
- 如果数据不包含使用双引号 (") 括起的值，则无需指定任何 SerDe，也就是使用默认的 org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe，然后指定数据的分隔符：

  ```
  CREATE EXTERNAL TABLE iteblog_csv_test (
     id string,
     name string,
     age int
  )
     PARTITIONED BY (year STRING)
     ROW FORMAT DELIMITED
       FIELDS TERMINATED BY ','
       ESCAPED BY '₩₩'
       LINES TERMINATED BY '₩n'
     LOCATION 'hdfs://user/iteblog/testdb';
  ```

  这个是可以使得数据类型是真实的。