使用Spark读写CSV格式文件

CSV格式的文件也称为逗号分隔值（Comma-Separated Values，CSV，有时也称为字符分隔值，因为分隔字符也可以不是逗号。在本文中的CSV格式的数据就不是简单的逗号分割的），其文件以纯文本形式存表格数据（数字和文本）。CSV文件由任意数目的记录组成，记录间以某种换行符分隔；每条记录由字段组成，字段间的分隔符是其它字符或字符串，最常见的是逗号或制表符。通常，所有记录都有完全相同的字段序列。

本篇文章将介绍如何使用Spark 1.3+的外部数据源接口来自定义CSV输入格式的文件解析器。这个外部数据源接口是由databricks公司开发并开源的（地址：https://github.com/databricks/spark-csv），通过这个类库我们可以在Spark
SQL中解析并查询CSV中的数据。因为用到了Spark的外部数据源接口，所以我们需要在Spark 1.3+上面使用。在使用之前，我们需要引入以下的依赖：

```
<dependency>
 <groupId>com.databricks</groupId>
 <artifactId>spark-csv_2.10</artifactId>
 <version>1.0.3</version>
</dependency>
```

目前spark-csv_2.10的最新版就是1.0.3。如果我们想在Spark
shell里面使用，我们可以在--jars选项里面加入这个依赖，如下：

```
[iteblog@spark $] bin/spark-shell --packages com.databricks:spark-csv_2.10:1.0.3
```

和《Spark SQL整合PostgreSQL》文章中用到的load函数类似，在使用CSV类库的时候，我们需要在options中传入以下几个选项：

1、path：看名字就知道，这个就是我们需要解析的CSV文件的路径，路径支持通配符；
2、header：默认值是false。我们知道，CSV文件第一行一般是解释各个列的含义的名称，如果我们不需要加载这一行，我们可以将这个选项设置为true；
3、delimiter：默认情况下，CSV是使用英文逗号分隔的，如果不是这个分隔，我们就可以设置这个选项。
4、quote：默认情况下的引号是""，我们可以通过设置这个选项来支持别的引号。
5、mode：解析的模式。默认值是PERMISSIVE，支持的选项有
（1）、PERMISSIVE：尝试解析所有的行，nulls are inserted for missing tokens and extra tokens are ignored.
（2）、DROPMALFORMED：drops lines which have fewer or more tokens than expected
（3）、FAILFAST: aborts with a RuntimeException if encounters any malformed line

如何使用

1、在Spark SQL中使用

我们可以通过注册临时表，然后使用纯SQL方式去查询CSV文件：

```
CREATE TABLE cars
USING com.databricks.spark.csv
OPTIONS (path "cars.csv", header "true")
```

我们还可以在DDL中指定列的名字和类型，如下：

```
CREATE TABLE cars (yearMade double, carMake string, carModel string, comments string, blank string)
USING com.databricks.spark.csv
OPTIONS (path "cars.csv", header "true")
```

2、通过Scala方式

推荐的方式是通过调用SQLContext的load/save函数来加载CSV数据：

```
import org.apache.spark.sql.SQLContext

val sqlContext = new SQLContext(sc)
```

```
val df = sqlContext.load("com.databricks.spark.csv", Map("path" -> "cars.csv", "header" -> "true
"))
df.select("year", "model").save("newcars.csv", "com.databricks.spark.csv")
```

当然，我们还可以使用com.databricks.spark.csv._的隐式转换：

```
import org.apache.spark.sql.SQLContext
import com.databricks.spark.csv._

val sqlContext = new SQLContext(sc)

val cars = sqlContext.csvFile("cars.csv")
cars.select("year", "model").saveAsCsvFile("newcars.tsv")
```

3、在Java中使用

和在Scala中使用类似，我们也推荐调用SQLContext类中 load/save函数

```
/**
 * User: 过往记忆
 * Date: 2015-06-01
 * Time: 下午23:26
 * bolg:
 * 本文地址：/archives/1380
 * 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 * 过往记忆博客微信公共帐号：iteblog_hadoop
 */

import org.apache.spark.sql.SQLContext

SQLContext sqlContext = new SQLContext(sc);

HashMap<String, String> options = new HashMap<String, String>();
options.put("header", "true");
options.put("path", "cars.csv");

DataFrame df = sqlContext.load("com.databricks.spark.csv", options);
df.select("year", "model").save("newcars.csv", "com.databricks.spark.csv");
```

在Java或者是Scala中，我们可以通过CsvParser里面的函数来读取CSV文件:

```
import com.databricks.spark.csv.CsvParser;
SQLContext sqlContext = new org.apache.spark.sql.SQLContext(sc);

DataFrame cars = (new CsvParser()).withUseHeader(true).csvFile(sqlContext, "cars.csv");
```

4、在Python中使用

在Python中，我们也可以使用SQLContext类中 load/save函数来读取和保存CSV文件：

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)

df = sqlContext.load(source="com.databricks.spark.csv", header="true", path = "cars.csv")
df.select("year", "model").save("newcars.csv", "com.databricks.spark.csv")
```