Using Regular Expressions to Extract Fields for Hive Tables

hcc-58591.zipHive RegexSerDe can be used to extract columns from the input file using regular expressions. It's used only to deserialize data, while data serialization is not supported (and obviously not needed). The initial motivation to create such a SerDe was to process Apache web logs. There are two classes available:

- org.apache.hadoop.hive.contrib.serde2.RegexSerDe, introduced in Hive-0.4 by HIVE-662, and
- org.apache.hadoop.hive.serde2.RegexSerDe, a built-in class introduced in Hive-0.10 by HIVE-1719

The former is kept to facilitate easier migration for legacy apps, while the letter is recommended for the new apps. The SerDe works by matching columns in the table definition with regex groups defined and captured by the regular expression. A regex group is defined by parenthesis "(...)" inside the regex. Note that this is one of common mistakes by beginners who spend time creating great regular expressions but displace or fail to mark regex groups. The new, built-in version supports following primitive column types: TINYINT, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, STRING, BOOLEAN and DECIMAL, in contrast to the "Contrib" version which supported only STRING column type.

Regarding the number of columns in the table definition and the number of regex group, they must match, otherwise a warning is printed and the table is not populated. On individual lines, if a row matches the regex but has less than expected groups, the missing groups and table fields will be NULL. If a row matches the regex but has more than expected groups, the additional groups are just ignored. If a row doesn't match the regex then all fields will be NULL. The regex is provided as a SerDe required property called "input.regex". Another supported property is "input.regex.case.insensitive" which can be "true" or "false" (default), while ""output.format.string" supported by the contrib version is not supported any more.

As an example consider a tab separated text input file composed of 5 fields: id int, city_org string, city_en string, country string, ppl float, and we'd like to create a table using only 3 of those 5 fileds, namely: id, city_org, and ppl, meaning that we'd like to ignore 3rd and 4th column. (Of course we can do the same using a view, but for the sake of the discussion let's do it using RegexSerDe.) We can define our table as:

```
$ hdfs dfs -mkdir -p hive/serde/regex
$ hdfs dfs -put allcities.utf8.tsv hive/serde/regex
hive> CREATE EXTERNAL TABLE citiesr1 (id int, city_org string, ppl float) ROW FORMAT SERDE
```

Note that the regex contains 3 regex groups capturing the first, second and fifth field on each line, corresponding to 3 table columns:

- (\\d+), the leading integer id composed of 1 or more digits,
- ([^\\t]*), a string, everything except tab, positioned between 2nd and 3rd delimiting tabs. If we know that the column contains no spaces we can also use "\\S+" in our example this is not the case, (however, we are making such assumption about the 3rd and the 4th field) and
- (\\d++.\\d++).*'), a float with at least 1 digit before and after the decimal point.

Input sample (files used in examples are available in the attachment):

```
110    La Coruña    Corunna    Spain    0.37
112    Cádiz    Cadiz    Spain    0.4
120    Köln    Cologne    Germany    0.97

hive> select * from citiesr1 where id>100 and id<121;
110    La Coruña    0.37
112    Cádiz    0.4
```

```
120    Köln    0.97
```

Now, let's consider a case when some fields are missing in the input file, and we attempt to read it using the same regex used for the table above:

```
$ hdfs dfs -mkdir -p hive/serde/regex2
$ hdfs dfs -put allcities-flds-missing.utf8.tsv hive/serde/regex2
hive> CREATE EXTERNAL TABLE citiesr2 (id int, city_org string, ppl float) ROW FORMAT SERDE
```

Input sample:

```
2<tab><tab>Osaka<tab><tab>
31<tab><tab>Yakutsk<tab>Russia
121<tab>München<tab>Munich<tab><tab>1.2
```

On lines 1 and 3 we have 5 fields, but some are empty, while on the second line we have only 4 fields and 3 tabs. If we attempt to read the file using the regex given for table citiesr1 we'll end up with all NULLs on these 3 lines because the regex doesn't match these lines. To rectify the problem we can change the regex slightly to allow for such cases:

```
hive> CREATE EXTERNAL TABLE citiesr3 (id int, city_org string, ppl float) ROW FORMAT SERDE
```

The first 2 groups are unchanged, however we have replaced both "\\S+" for unused columns with [^\\t], *the last delimiting tab is optional, and the last group is not set to "(.)*" meaning everything after the last tab including empty string. With this changes, the above 3 lines become:

```
hive> select * from citiesr3 where id in (2, 31, 121);
2        NULL
31        NULL
121    München    1.2
```

The real power of RegexSerDe is that it can operate not only on delimiter boundaries, as shown above, but also inside individual columns. Besides processing web logs and extracting desired fields and patterns from the input file another common use case of RegexSerDe is to read files with multi-character field delimiters because "FIELDS TERMINATED BY" doesn't support them. (However, since Hive-0.14 there is also a contributed MultiDelimitSerDe which supports multi-char delimiters.)

Note: All tests done on a HDP-2.4.0 cluster running Hive-1.2.1.

Related questions: regex pattern for hive regex serde