1. Motivation

In the big data ecosystem, the file system is a very common storage system, especially the hdfs file system , and flink sql is a very common way to develop flink jobs, but unfortunately, in flink's job, we can't use sql to read the data and then write it to file system with kind of formats, so we add a bucket file system connector. Users can write data to the file system by using flink sql.

2. Features

1) In order to avoid multiple file system connectors, we will enrich the functionality of the existing File System Connector [1] instead of adding a new connector, and retain its existing features to avoid the code which referencing the connector is wrong.
2) For streaming output, the underlying code will use StreamingFileSink to write data to the file system instead of the CsvTableSink#consumeDataStream method in the File System Connector. For the batch output, we will not make changes this time, and then add them later.
3) The currently supported data formats are json, csv, arvo, and parquet. Subsequent additions to other format support, such as orc, sequence-file, etc.
4) Add support for the partition. If the user specifies the partition field, the data is written to the relevant partition according to the partition field set by the user. If the user does not specify the partition field, the data is written to the relevant time partition

according to the default policy of StreamingFileSink. Of course, the user can specify the partition time format for the outoup.

3. Code

```
.connect(
      new FileSystem ()
       // required: the file system path where data write to
       .path("hdfs://localhost/tmp/flink-data/csv")
       // optional,the data type ,the rowFormat is used to write row-wise data,e.g. json or csv
       // the bultFormat is used to write bulk-encoding data,e.g. Parquet、avro
       // the default is rowFormat(),
       .rowFormat()
       .bultFormat()

       // optional ,partition columns
       .partition(new String[]{"date","country"})
       # optional ,date partition of the data, default is yyyy-MM-dd--HH
        .dateFormat("yyyy-MM-dd-HHmm")
)
```

explanation：

path：

The directory where the user will write the data.

rowFormat & bultFormat ：

When writing different formats of data to the file system, the way it is used is different, just like the forRowFormat and forBulkForma methods in StreamingFileSink, you need to use different write methods。

So if we want to uniquely determine the relevant Table Sink Factory for each output format (similar to finding different kafka table sink factories via kafka version), we need to specify the output format in the FileSytem Connector, but this data format information already exists in FormatDescriptor. In the meantime, if we specify the output format again in FileSystem Connector, then it is repeated, and we also need to verify that the output format in FileSystem Connector is the same as the output format in FormatDescriptor. In addition,the data with row-encoding format such as json and csv can be used with RowFileSytemTableSinkFactory. the data with bult-encoding format like Parquet requires a relevant ParquetFileSystemTableSinkFactory..

So consider various factors, we add the rowFormat and bultFormat methods to determine the data format that the user wants to write. If it is the json, csv with row-encoding format , you need to select the rowFormat method, the system will select the RowFileSystemTableSinkFactory to build the relevant Sink; If user selects the bultFormat method, the system will combine the FormatDescriptor to select the relevant

ParquetFileSystemTableSinkFactory or AvroFileSystemTableSinkFactory.。

If the user adds both rowFormat and bultFormat, the latter will replace the former and the former will fail. If the user does not specify, the rowFormat is used by default, in order to be compatible with the current system using the File System Connector but not specifying the rowFormat or bultFormat.

partition:

If the user specifies a partition field, the data will be partitioned according to the user's field. Otherwise, according to the default policy of StreamingFileSink towrite data, ie yyyy-MM-dd-HH format,

dateFormat：

If the user does not set the partition field, you can specify the output partition format by the dateFormat method.

4. DDL

```
CREATE TABLE MyUserTable (
  a int,
  b varchar,
  c double,
  date varchar,
  country varchar
```

```
  )
  PARTITIONED BY (date, country)
   WITH (
    'connector.type' =  'filesystem',
    -- required: the file system path where data write to
    'connector.path' = 'hdfs:///tmp/json',
    'connector.format.type' = 'row', -- required ,the data type ,the rowFormat is used to write row-wise data,e.g. json or csv
                                     -- the bultFormat is used to write bulk-encoding data,e.g. Parquet
    'connector.date.format' = 'yyyyMMddHHmm', -- optional ,date partition of the data, default is yyyy-MM-dd--HH
    'update-mode' = 'append'         -- required: update mode when used as table sink,only support append mode now.
  )
```

5. Add Parquet Format Descriptor

Currently, the flink-formats module in the system already has a parquet format, but the parquet is not a Format Descriptor, so we add a Parquet Format Descriptor,user can write data to filesystem with parquet data format.

Support for constructing Parquet Format Descriptor in three ways, users can only choose one

```
public Parquet specificClass(Class<? extends
SpecificRecordBase> specificClass) {
    ...........
    return this;
}

public Parquet schema(Schema schema) {
    ...........
    return this;
}

public Parquet reflectClass(Class reflectClass) {
    ...........
    return this;
}
```

[1]. https://ci.apache.org/projects/flink/flink-docs-release-1.9/dev/table/connect.html#file-system-connector