TP_SPARK_SQL

Axel Jacquin

January 18, 2017

In this exercise you will use Spark SQL to load data from MySQL, process it, and store it to HDFS.

1 Review the data in MySQL

1. Connect to MySQL using credentials training/training. You have to specify that you are going to use a password using -p option.

2. Review the table loudacre.webpage (structure + some records)

Note that the data in the associated_files column is a comma-delimited string. Loudacre would like to make this data available in an Hive table, but in order to perform required analysis, the associated_files data must be extracted and normalized.

Your goal in the next section is to use Spark SQL to extract the data in the column, split the string, and create a new dataset in HDFS containing each web page number, and its associated files in separate rows.

2 Load the data from MySQL

3. In a spark-shell, import the SQLContext class definition then create new SQLContext instance using sc.

```
scala> import org.apache.spark.sql.SQLContext
scala> val sqlCtx = new SQLContext(sc)
```

4. Create a new DataFrame named webpages based on the webpage table in MySQL

```
scala> val webpages=sqlCtx.load("jdbc",Map(
"url"->"jdbc:mysql://localhost/loudacre?user=training&password
=training",
"dbtable" -> "webpage"))
```

5. Examine the schema of the new DataFrame.

6. Create a new DataFrame by selecting the web_page_num and associated_files columns from the existing DataFrame using select method.

7. Use rdd method to convert the last dataframe in a RDD.

The RDD obtained is a RDD[Row]. You can use getInt(i) on a Row object to obtained the value of the field n°i of the Row. Use getString(i) if this field is a String.

8. Use a map transformation to optain a PairRDD where the key is web_page_num and the value is associated_files.

9. Now that you have an RDD, you can use the flatMapValues transformation to split and extract the filenames in the associated_files column.

Note: As a reminder flatMapValues applies a fonction on just the values of a PairRDD, keeping the key the same.

10. Create a new Dataframe from this RDD

11. Print the schema of this new Dataframe
Note that SparkSQL gave the columns generic names _1 and _2

12. Create a new DataFrame by renaming the columns to reflect the data they hold.
In Scala, you can use the toDF shortcut method to create a new DataFrame based on an existing one with the columns renamed.

13. This last DataFrame contains the processed dataset. Check it.

14. Save this final Dataframe to HDFS in Parquet format (the default one) using save method.

15. Optional: Create the corresponding table in Hive.