BAAHU'S BLOG

GoLang Developer 170495 DE…
GoLang Developer | 170495

SPARK DataFrame – Java Example

baahu October 2, 2016 No Comments

Tweet

A `DataFrame` is a collection of data, organized into named columns. `DataFrames` are similar to tables in a traditional database
`DataFrame` can be constructed from sources such as Hive tables, Structured Data files, external databases, or existing RDDs.
Under the hood, a `DataFrame` contains an RDD composed of `Row` objects with additional schema information of the types in each col.
In the Java example code below we are retrieving the details of the employee who draws the max salary(i.e get the name of the CEO 😉)
We are going to create a `DataFrame` over a text file, every line of this file contains employee information in the below format EmployeeID,Name,Salary

1. We start with creating `SQLContext`.SQLContext is used for initializing the functionalities of Spark SQL. `SparkContext` class object is required for initializing SQLContext class object.
2. We store the schema of the table in a String(empid,name,salary) and create a ArrayList of `StructField` objects.
   I am avoiding using a for loop to create `StructField` objects since the types of columns are heterogeneous – empid,name are `String` and salary is `Integer`.
3. Create a `StructType` object using the `StructField` objects created in the above step.
4. Load the file into an RDD and create a `JavaRDD` of type `Row` for each of the entries present in the input file.
5. Create a `DataFrame` object using the `JavaRDD` in the above step and the `StructType` object created in step 3.
6. Register a temporary table "employee" , so that we can execute `sql` style queries using a `SQLContext`.
7. Execute the sql query to find details of the employee who draws the max salary and print the same using by invoking `foreach()` action.

Java
```java
import java.util.List;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.Function;
import org.apache.spark.api.java.function.VoidFunction;
import org.apache.spark.sql.DataFrame;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.RowFactory;
import org.apache.spark.sql.SQLContext;
import org.apache.spark.sql.types.DataTypes;
import org.apache.spark.sql.types.StructType;
import org.apache.spark.sql.types.StructField;



public class DataFrameExample {

    public static void main(String[] args) {

        SparkConf conf = new SparkConf().setAppName("DataFrameExample").setMaster("local[*]");
        JavaSparkContext jsp = new  JavaSparkContext(conf);

        //Create the sql context
        SQLContext sqlContext =  new SQLContext(jsp);

        String schemaString = "name,empid,salary";

        List<StructField> fieldList = new ArrayList<StructField>();

        //Create 3 objects for 3 columns, please note that salary is an integer
        fieldList.add(DataTypes.createStructField("name",DataTypes.StringType,true));
        fieldList.add(DataTypes.createStructField("empid",DataTypes.StringType,true));
        fieldList.add(DataTypes.createStructField("salary",DataTypes.IntegerType,true));

        //Now create a StructType object from the list
        StructType schema = DataTypes.createStructType(fieldList);

        //Load the file which contains the data into an RDD
        JavaRDD<String> rdd = jsp.textFile("file:///home/baahu/employee");

        //convert the input to a list of row objects
        JavaRDD<Row> rowsRDD = rdd.map(new Function<String,Row>(){

            @Override
            public Row call(String arg0) throws Exception {
                //arg0 contains each line from the rdd , i.e empid,name,salary
                String [] data = arg0.split(",");

                //We make sure that we convert the salary field to integer
                return RowFactory.create(data[0],data[1],Integer.valueOf(data[2]));
            }});

        //Create the dataframe from the rowsRDD and schema
        DataFrame dataFrame = sqlContext.createDataFrame(rowsRDD, schema);
        dataFrame.registerTempTable("employee");

        //Run the sql query to find details of employee who draws the max salary
        DataFrame maxSalDataFrame = sqlContext.sql("Select * from employee order by salary desc limit 1 ");

        maxSalDataFrame.toJavaRDD().foreach(new VoidFunction<Row>(){

            @Override
            public void call(Row arg0) throws Exception {
                System.out.println(" Name is :"+arg0.get(0)+" Emp id is:"+arg0.get(1)+" Salary is:"+arg0.get(2));
            }});
    }
}
```

Sample Input

```
John,54001,10000
Johhny,54002,10000000
Janardhan,54003,5000
```

Sample Output

```
Name is :Johhny Emp id is:54002 Salary is:10000000
```

Tweet

SPARK DataFrame, spark, SQLContext