**Sparksql、dataframe熟悉，能够使用**

Sparksql

Shark是SparkSQL的前身,SparkSQL支持查询原生的RDD,RDD是Spark平台的核心概念，是Spark能够高效的处理大数据的各种场景的基础,能够在Scala中写SQL语句。支持简单的SQL语法检查，能够在Scala中写Hive语句访问Hive数据，并将结果取回作为RDD使用。

Dataframe

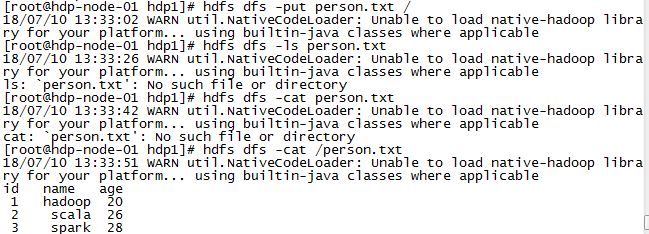
Dataframe是一个一个Row类型的RDD，Dataframe原生API可以操作Dataframe。

**将课堂上的场景全部操作一遍（demo部分先照着做、理解，再背着写出来，再总结；源码部分要点进去看）**

DataFrames

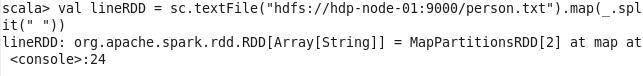
1. 在本地创建一个文件，有三列，分别是id、name、age，用空格分隔，然后上传到hdfs上

hdfs dfs -put person.txt /



2.在spark shell执行下面命令，读取数据，将每一行的数据使用列分隔符分割

val lineRDD = sc.textFile("hdfs://hdp-node-01:9000/person.txt").map(\_.split(""))



3.定义case class（相当于表的schema）

case class Person(id:Int, name:String, age:Int)

1. 将RDD和case class关联

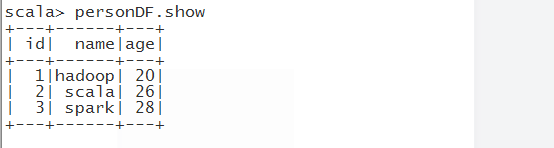
val personRDD = lineRDD.map(x => Person(x(0).toInt, x(1), x(2).toInt))

1. 将RDD转换成DataFrame

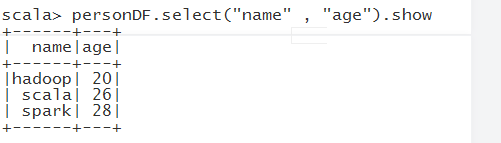
val personDF = personRDD.toDF

6.对DataFrame进行处理

personDF.show



1. 通过DataFrame查询2个字段(DSL风格)



### DSL风格语法（dataframe常用操作）

//查看DataFrame中内容

personDF.show

//查看DataFrame部分列中的内容

personDF.select(personDF.col("name")).show

personDF.select(col("name"), col("age")).show

personDF.select("name").show

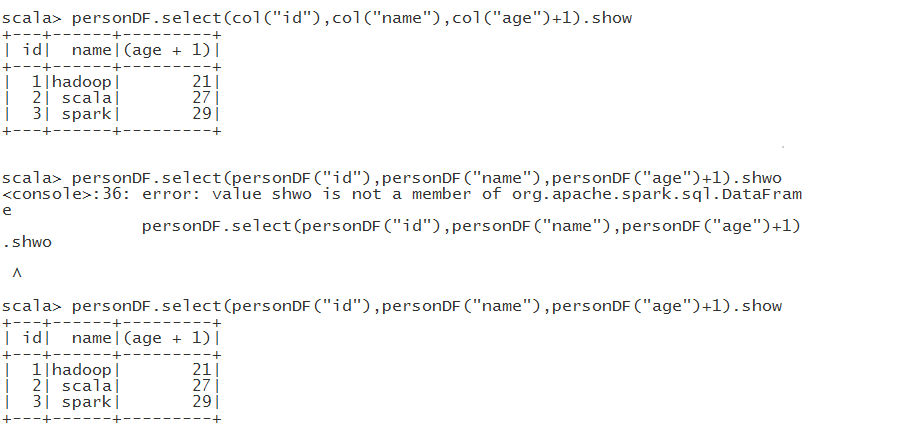
//打印DataFrame的Schema信息

personDF.printSchema

//查询所有的name和age，并将age+1

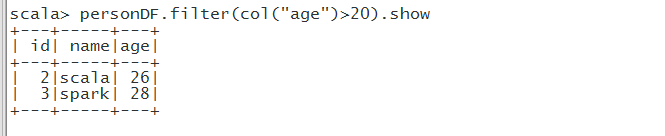
personDF.select(col("id"), col("name"), col("age") + 1).show

personDF.select(personDF("id"), personDF("name"), personDF("age") + 1).show



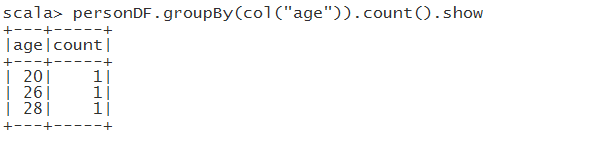
//过滤age大于等于20的

personDF.filter(col("age") >= 20).show



//按年龄进行分组并统计相同年龄的人数

personDF.groupBy("age").count().show()



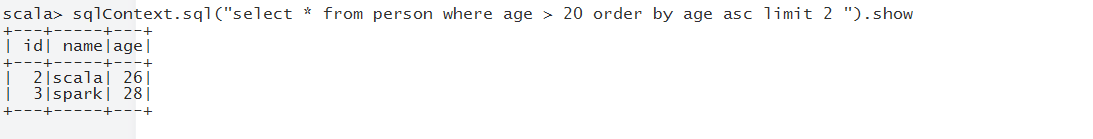
### SQL风格语法

如果想使用SQL风格的语法，需要将DataFrame注册成表

personDF.registerTempTable("person")

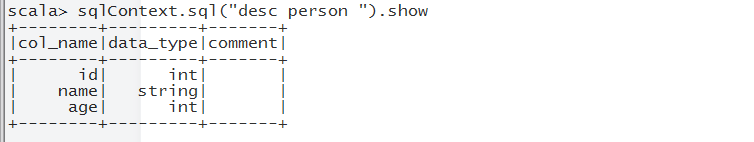
//查询年龄最大的前两名

sqlContext.sql("select \* from t\_person order by age desc limit 2").show



//显示表的Schema信息

sqlContext.sql("desc person").show



### 下面使用程序查询

首先在maven项目的pom.xml中添加Spark SQL的依赖

<dependency>  
<groupId>org.apache.spark</groupId>  
<artifactId>spark-sql\_2.20</artifactId>  
<version>2.2.0</version>  
</dependency>

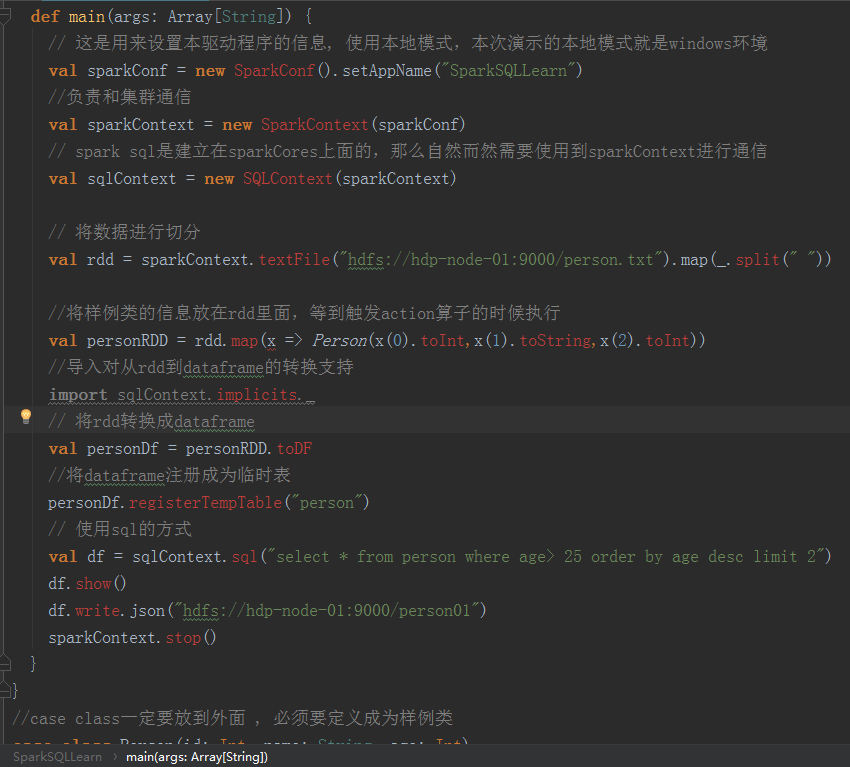
通过反射推断Schema

object SparkSQLLearn {  
 def main(args: Array[String]) {  
 // 这是用来设置本驱动程序的信息, 使用本地模式，本次演示的本地模式就是windows环境,启动2个线程  
 val sparkConf = new SparkConf().setAppName("SparkSQLLearn").setMaster("local[2]")  
 //负责和集群通信  
 val sparkContext = new SparkContext(sparkConf)  
 // spark sql是建立在sparkCores上面的，那么自然而然需要使用到sparkContext进行通信  
 val sqlContext = new SQLContext(sparkContext)  
  
 // 将数据进行切分  
 val rdd = sparkContext.textFile("G:\\spark\\sparksql\\person.txt").map(\_.split(""))  
  
 //将样例类的信息放在rdd里面，等到触发action算子的时候执行  
 val personRDD = rdd.map(x => Person(x(0).toInt,x(1).toString,x(2).toInt))  
 //导入对从rdd到dataframe的转换支持  
 import sqlContext.implicits.\_  
 // 将rdd转换成dataframe  
 val personDf = personRDD.toDF  
 //将dataframe注册成为临时表  
 personDf.registerTempTable("person")  
 // 使用sql的方式  
 sqlContext.sql("select \* from person where age> 25 order by age desc limit 2").show()  
 sparkContext.stop()  
}  
}  
//case class一定要放到外面 , 必须要定义成为样例类  
case class Person(id: Int, name: String, age: Int)



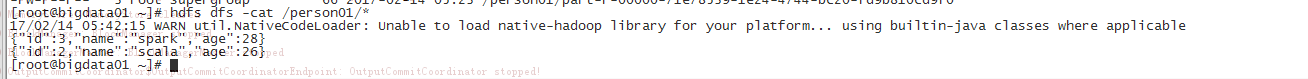
打包到集群的方式：

def main(args: Array[String]) {  
 // 这是用来设置本驱动程序的信息, 使用本地模式，本次演示的本地模式就是windows环境  
 val sparkConf = new SparkConf().setAppName("SparkSQLLearn")  
 //负责和集群通信  
 val sparkContext = new SparkContext(sparkConf)  
 // spark sql是建立在sparkCores上面的，那么自然而然需要使用到sparkContext进行通信  
 val sqlContext = new SQLContext(sparkContext)  
  
 // 将数据进行切分  
 val rdd = sparkContext.textFile("hdfs://hdp-node-01:9000/person.txt").map(\_.split(""))  
  
 //将样例类的信息放在rdd里面，等到触发action算子的时候执行  
 val personRDD = rdd.map(x => Person(x(0).toInt,x(1).toString,x(2).toInt))  
 //导入对从rdd到dataframe的转换支持  
 import sqlContext.implicits.\_  
 // 将rdd转换成dataframe  
 val personDf = personRDD.toDF  
 //将dataframe注册成为临时表  
 personDf.registerTempTable("person")  
 // 使用sql的方式  
 val df = sqlContext.sql("select \* from person where age> 25 order by age desc limit 2")  
 df.show()  
 df.write.json("hdfs://hdp-node-01:9000/person01")  
 sparkContext.stop()  
}  
}  
//case class一定要放到外面 , 必须要定义成为样例类  
case class Person(id: Int, name: String, age: Int)



提交的指令：

/usr/local/apps/spark-2.2.0-bin-hadoop2.6/bin/spark-submit --master spark://hdp-node-01:7077 --class com.youe.spark.SparkSQLLearn /root/hello-spark-1.0.jar



### 通过StructType直接指定Schema

def main(args: Array[String]): Unit = {  
 //这是设置本驱动程序的信息 , 设置为本地模式，本次演示是在windows上面跑动，并启动4个线程  
 val sparkConf = new SparkConf().setAppName("SparkSqlContext").setMaster("local[4]")  
 //负责和集群通讯  
 val sparkContext = new SparkContext(sparkConf)  
 // 这是sql环境，sparksql会用到sparkcores 所以需要传入sparkContext和集群进行通信  
 val sparkSqlContext = new SQLContext(sparkContext)  
 //对每行数据进行切分  
 val rdd = sparkContext.textFile("G:\\spark\\sparksql\\person.txt").map(line => line.split(""))  
 // 定义schema 表的元信息  
 val schema = StructType(List(  
 StructField("id" , IntegerType , false),  
 StructField("name" ,StringType,true),  
 StructField("age" ,IntegerType,true)  
 ))  
 // 将rdd转变为rowRDD  
 val rowRDD = rdd.map(x => Row(x(0).toInt , x(1).toString,x(2).toInt))  
 val personDF = sparkSqlContext.createDataFrame(rowRDD,schema)  
 personDF.registerTempTable("person")  
 val df = sparkSqlContext.sql("select \* from person order by age asc")  
 df.show()  
 sparkContext.stop()  
}

