spark做etl清洗json数据

没有环境可以用 spark-shell␀0␀2␀0␀2 :paste␀0␀2 代码 测试程序␀0␀2 按ctrl+D结束输入

sudo -u hdfs spark-shell --jars /data/yz/fastjson-1.2.6.jar --master yarn

```
import com.alibaba.fastjson.JSON
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.functions.{col, concat_ws, regexp_replace}
import org.apache.spark.sql.types.{StringType, StructField, StructType}
import org.apache.spark.{SparkConf, SparkContext}
import org.joda.time.DateTime

/**
  * spark-submit --class spark_hive_message_info_sms --deploy-mode client --num-executors 2 --executor-memory 2g --
executor-cores 2 --driver-memory 5g --master yarn /data2/gsj/ql_etl_message_info_sms.jar
  */
object spark_hive_message_info_sms {
  def main(args: Array[String]) {
        if (args.length<1){
          System.err.println("please give the correct params")
          System.exit(1)
        }

    val conf = new SparkConf().setAppName("spark_hive_message_info_sms")//.setMaster("local")
    val sc = new SparkContext(conf)
    //spark-shell paste
    val sqlContext = new SQLContext(sc)
    import sqlContext.implicits._



    @transient
    val hdfsconf = sc.hadoopConfiguration
    @transient
    val fs = org.apache.hadoop.fs.FileSystem.get(hdfsconf)

    //val date_etl = DateTime.now().withTimeAtStartOfDay().plusDays(-1).toString("yyyy-MM-dd")
    val date_etl =args(0).toString

    println("============================================================date_etl:"+date_etl)

    val originToJson = (originText:String) => {
      var result = ("")
      try {
        val jsonData = JSON.parseObject(originText)
        val data = jsonData.getString("data")
        result = (data)
      } catch {
        case ex: Exception => {
          println("error:"+ex.printStackTrace()+"orgintext==============>"+originText)
        }
      }
      result
```

```
    }

    val schemaString = "batchNumber,flagId,sendStatus,sendTime"

    val schema =StructType(schemaString.split(",").map(fieldName => StructField(fieldName, StringType, true)))

    val json_rdd =
sc.textFile("/datahouse/ods/topic/bigdata_rca_jsystem_message/"+date_etl+"/*").map(x=>x.replaceAll("\\\\r\\\\n","
").replaceAll("\\\\n"," ").replaceAll("\\\\r"," "))
      .map(x=>originToJson(x))
    val message_info_sms = sqlContext.read.schema(schema).json(json_rdd)

    val message_info_sms_select = message_info_sms.select(
      regexp_replace(col("batchNumber"),"\\`","_").alias("batchNumber"),
      regexp_replace(col("flagId"),"\\`","_").alias("flagId"),
      regexp_replace(col("sendStatus"),"\\`","_").alias("sendStatus"),
      regexp_replace(col("sendTime"),"\\`","_").alias("sendTime"))

    @transient
    val expr = concat_ws("`", message_info_sms_select.columns.map(col): _*)

    if(fs.exists(new org.apache.hadoop.fs.Path("/tmp/message_info_sms_tmp/")))
      fs.delete(new org.apache.hadoop.fs.Path("/tmp/message_info_sms_tmp/"),true)


message_info_sms_select.na.fill("NULL").filter($"batchNumber"!=="NULL").select(expr).repartition(1).map(_.getString(0))



    if(fs.exists(new org.apache.hadoop.fs.Path("/user/hive/warehouse/h_biz_data.db/message_info_sms_tmp/part-
00000.snappy"))){
      fs.delete(new org.apache.hadoop.fs.Path("/user/hive/warehouse/h_biz_data.db/message_info_sms_tmp/part-
00000.snappy"),true)
    }

    fs.rename(new org.apache.hadoop.fs.Path("/tmp/message_info_sms_tmp/part-00000.snappy"),new
org.apache.hadoop.fs.Path("/user/hive/warehouse/h_biz_data.db/message_info_sms_tmp/part-00000.snappy"))
    sc.stop()

    //
message_info_sms_select.na.fill("NULL").filter($"user_id"!=="NULL").select(expr).repartition(1).map(_.getString(0)).sa
"+year)
    //      println("year:"+year+" is done============================")
    //    }
  }
}
```