

**【多易教育】**

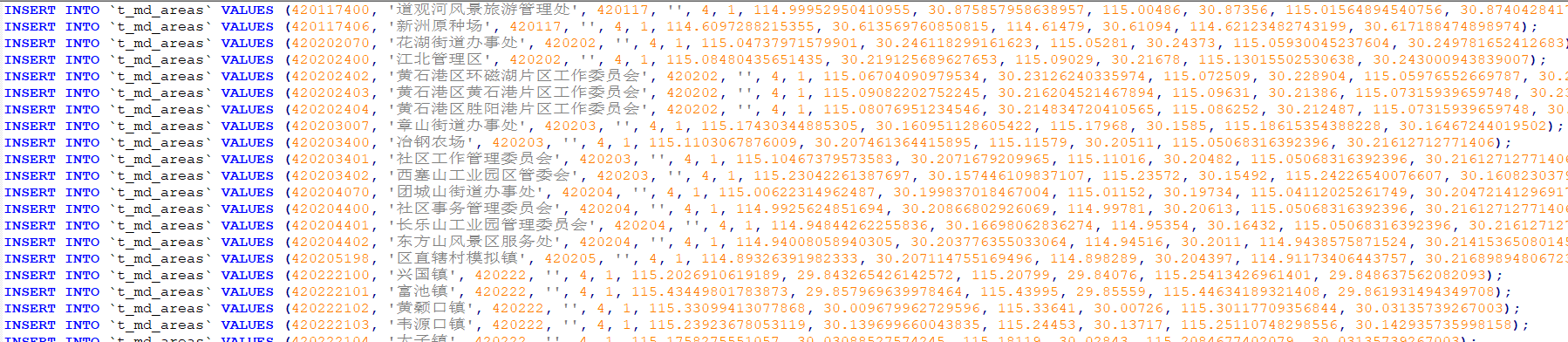
**<yiee-数据运营系统>**

**需求开发手册**

# 地理位置字典构建

## 需求描述

有如下格式的地理位置数据



如要整理成如下格式：

longitude latitude 省 市 区

----》

GEOHASH码（5位） 省 市 区

## 开发思路

方案1：

可以讲初始数据导入mysql，然后直接在mysql上执行一个查询语句来得到结果，并存入另一张表，然后，用spark读取mysql中的结果表，转成parquet格式写入hdfs

方案2：

可以将初始数据导入mysql，然后用spark读取该初始数据表，在spark中执行sql运算，得到结果，并以parquet格式写入hdfs

方案3：

直接用spark读取初始数据文件，按文本文件进行切割抽取转换处理，加工成结构化数据得到dataframe，注册临时表，运行sql逻辑，将结果以parquet格式写入hdfs

## 代码实现

### 方案1代码

1. 将初始数据导入mysql表t\_md\_areas
2. 执行查询sql

|  |
| --- |
| create table t\_area\_dict  AS  SELECT  a.BD09\_LNG,  a.BD09\_LAT,  d.areaname AS province,  c.areaname AS city,  b.areaname AS district  FROM  t\_md\_areas a  JOIN t\_md\_areas b ON a. LEVEL = 4 AND a.PARENTID = b.id  JOIN t\_md\_areas c ON b.parentid = c.id  JOIN t\_md\_areas d ON c.parentid = d.id |

1. 写spark代码来读取mysql中的结果数据并转存hdfs

|  |
| --- |
| **import** java.util.Properties  **import** org.apache.spark.sql.SparkSession  */\*\*  \** **@date:** *2019/8/25  \** **@site:** *www.doitedu.cn  \** **@author:** *hunter.d 涛哥  \** **@qq:** *657270652  \** **@description:** *地理位置名称字典构建程序  \*/* **object** AreaDict {   **def** main(args: Array[String]): Unit = {   **val** spark = SparkSession.*builder*()  .appName(**this**.getClass.getSimpleName)  .master(**"local[\*]"**)  .getOrCreate()   **val** props = **new** Properties()  props.setProperty(**"user"**,**"root"**)  props.setProperty(**"password"**,**"root"**)  **val** df = spark.read.jdbc(**"jdbc:mysql://localhost:3306/demo"**,**"t\_area\_dict"**,props)   *//df.show(10,false)* df.coalesce(1).write.parquet(**"data\_ware/data/dict/out\_area\_dict"**)   spark.close()   }  } |

### **方案2代码**

1. 讲初始数据导入mysql
2. 写spark代码

|  |
| --- |
| **import** java.util.Properties **import** org.apache.spark.sql.SparkSession **object** AreaDict2 {  **def** main(args: Array[String]): Unit = {  **val** spark = SparkSession.*builder*()  .appName(**this**.getClass.getSimpleName)  .master(**"local[\*]"**)  .getOrCreate()  **val** props = **new** Properties()  props.setProperty(**"user"**,**"root"**)  props.setProperty(**"password"**,**"root"**)  **val** df = spark.read.jdbc(**"jdbc:mysql://localhost:3306/demo"**,**"t\_md\_areas"**,props)  df.createTempView(**"area"**)  **val** df2 = spark.sql(  **"""  |  |SELECT  | a.BD09\_LNG,  | a.BD09\_LAT,  | d.areaname AS province,  | c.areaname AS city,  | b.areaname AS district  |FROM  | area a  |JOIN area b ON a. LEVEL = 4 AND a.PARENTID = b.id  |JOIN area c ON b.parentid = c.id  |JOIN area d ON c.parentid = d.id  |  """**.stripMargin)  df2.coalesce(1).write.parquet(**"data\_ware/data/dict/out\_area\_dict"**)  spark.close()  } } |

### **方案3代码**

|  |
| --- |
| **import** cn.doitedu.commons.utils.SparkUitl  */\*\*  \** **@date:** *2019/8/25  \** **@site:** *www.doitedu.cn  \** **@author:** *hunter.d 涛哥  \** **@qq:** *657270652  \** **@description:** *地理位置字典构建程序、方案3实现代码  \*/* **case class** Bean(id:String,areaName:String,pid:String,level:String,longitude:String,latitude:String)   **object** AreaDict3 {   **def** main(args: Array[String]): Unit = {   **val** spark = SparkUitl.*getSparkSession*()  **import** spark.implicits.\_   **val** df = spark.read.text(**"data\_ware/data/dict/rawdata"**)    **val** area = df.map(row=>{  **val** line = row.getAs[String](**"value"**)  **val** arr = line.split(**""**)   **val** id = arr(4).substring(0,arr(4).length-1)  **val** areaName = arr(5).substring(0,arr(5).length-1)  **val** pid = arr(7).substring(0,arr(5).length-1)  **val** level = arr(8).substring(0,arr(5).length-1)  **val** longitude = arr(13).substring(0,arr(5).length-1)  **val** latitude = arr(14).substring(0,arr(5).length-2)   *// 把数据结构化：用tuple组装，或者用case class组装即可  //Bean(id,areaName,pid,level,longitude,latitude)* (id,areaName,pid,level,longitude,latitude)  }).toDF(**"id"**,**"name"**,**"pid"**,**"level"**,**"lng"**,**"lat"**)  area.createTempView(**"area"**)  **val** areaDict = spark.sql(  **"""  |  |SELECT  | a.lng,  | a.lat,  | d.name AS province,  | c.name AS city,  | b.name AS district  |FROM  | area a  |JOIN area b ON a. LEVEL = '4' AND a.pid = b.id  |JOIN area c ON b.pid = c.id  |JOIN area d ON c.pid = d.id  |  """**.stripMargin)  areaDict.coalesce(1).write.parquet(**""**)  spark.close()  } } |

## 需求开发总结

### 用到什么

Sql数据脚本导入mysql

Navicat中写sql查询

Spark读取mysql数据

Spark写parquet格式数据

### 学到了什么

1. Maven中子项目的互相依赖

1）将被依赖的子工程用maven的install命令进行编译打包和安装到maven仓库

2）在需要引入依赖的工程的pom中导入这个被依赖的工程的坐标



1. 数据处理中的一种思想

非结构化转成结构化数据

套路： 通过初始rdd的引入，各种算子抽取数据，最后返回（元组）或者（case class），然后这个rdd就可以toDF()，后续就可以用sql开发了；

Sql ： 一个一本万利的好东西！

# 行为日志预处理

## 需求描述

输入数据： 流量日志

处理逻辑：

1. 切字段，过滤字段数不够的
2. 解析json，废弃无用字段
3. 过滤掉空标识记录，解析异常记录
4. 根据gps坐标匹配地理位置信息

输出结果：

输出为parquet格式的文件

## 开发思路

顺着需求的流程，写代码即可

## 代码实现

## 需求开发总结

### 用到了什么

Json解析工具：FastJSON

Java集合转scala集合的隐式转换

广播变量的原理和使用

高德地图开放api的调用

mapPartitions算子的用法和本质

模式匹配从Row中抽取数据的方法

Spark算子中对异常的处理办法（异常了也要返回同类型结果，后续再过滤）

### 学到了什么

1. Spark分布式运算的基本原理

Task

ShuffleMapTask

ResultTask

TaskSet

Stage

Shuffle

Executor

1. DataSet
   * 1. DATASET的原理（rdd+特定类型+schema），schema靠特定类型的属性描述，或Row的描述
     2. DATASET和DataFrame的区别及联系
        + 1. Dataset一旦select后，就退化成了Dataset[Row]，就是dataframe
          2. Dataset在底层操作时，是有强类型约束检查的
     3. DATASET和rdd的区别：
        + 1. Dataset拥有schema信息，而rdd没有
          2. Dataset支持sql查询，而rdd没有

# 数仓ODS层建表及数据加载