尚硅谷电商用户画像三

版本：V 1.0

张晨

#### 

# 第六章 开发任务二：

# 完成标签的宽表合并

## 6.1 任务目标

当所有的单独标签任务都计算完成时，为了更加方便的查询及导出数据，要拼接出一张以用户ID为主键的宽表。宽表的每一列用**三级标签编号作为列名**。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| uid | person\_base\_agegroup | person\_base\_gender | person\_behavior\_xxcount | ….. | ….. |
| 1010 | 90后 | 男 | 12000 | … | … |
| 1011 | 80后 | 女 | 23000 | … | … |
| 1012 | …. | … | …. | … | … |

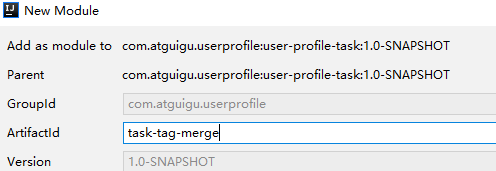
这张大宽表，包含了所有的标签，有多少个标签，就会有多少列。

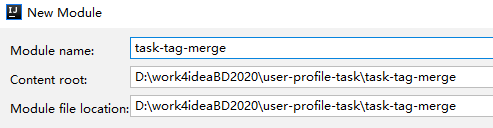
## 6.2 设计分析

1. 读取所有启动的标签任务中的标签列表
2. 读取标签列表中的标签编码和标签值类型，获得字段名和字段值，拼接成建表语句
3. 根据标签列表组合多表合并，同时进行行转列，组合成insert select 语句。

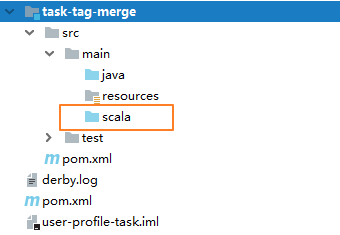
## 6.3 代码实现

### 6.3.1 新建模块





建立scala 源码目录



### 6.3.2 pom.xml加入依赖

|  |
| --- |
| <**dependencies**>  <**dependency**>  <**groupId**>com.atguigu.userprofile</**groupId**>  <**artifactId**>task-common</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**> </**dependencies**> |

### 6.3.3 task-common中的TagInfoDAO中补充方法

|  |
| --- |
| **def** getTagListOnTask(): List[TagInfo] ={   **val** tagListSql=**"select tg.id,tag\_code,tag\_name,parent\_tag\_id,tag\_type,tag\_value\_type,tag\_value\_limit,tag\_task\_id,tag\_comment,tg.create\_time"** +  **" from tag\_info tg join task\_info tk on tg.tag\_task\_id=tk.id where tk.task\_status='1' "  val** tagInfoList: List[TagInfo] = MySqlUtil.*queryList*(tagListSql, *classOf*[TagInfo],**true**)  tagInfoList } |

## 6.4主程序

|  |
| --- |
| object TaskTagMergeApp {    def main(args: Array[String]): Unit = {  *//1 环境  //2 查询所有开启的标签任务  //3 通过tagcode 创建当日新表  //4 拼接 insert select语句* val sparkConf: SparkConf = new SparkConf().setAppName("tag\_merge\_app")  *//.setMaster("local[\*]")* val sparkSession: SparkSession = SparkSession.*builder*()  .config(sparkConf).enableHiveSupport().getOrCreate()   *//需要什么参数？ 时间* val taskId: String = args(0)  val taskDate: String = args(1)   *//加载数据库名的配置* val properties: Properties = MyPropertiesUtil.*load*("config.properties")  val userProfileDbName = properties.getProperty("user-profile.dbname")  val wareHouseDbName = properties.getProperty("data-warehouse.dbname")  val hdfsStorePath = properties.getProperty("hdfs-store.path")   val tagInfoList: List[TagInfo] = TagInfoDAO.*getTagListOnTask*()  *//表名 注意不能含有 “-” 减号  // 采用每天建表的方式，而不是一张表多个分区  //主要考虑到，随着标签的增加修改，每天宽表的列都不一样，所以不能用一表多分区解决* val mergeTableName: String = "up\_tag\_merge\_" + taskDate.replaceAll("-","")   *//获得删表的语句* val dropTableSql: String = *genDropTableSql*(mergeTableName)   *//获得建表语句* val createTableSql: String = *genCreateTableSql*(tagInfoList: List[TagInfo],mergeTableName,userProfileDbName, hdfsStorePath)  *println*(createTableSql)   *//获得插入语句* val insertTableSql: String = *genInsertTableSql*(tagInfoList: List[TagInfo],mergeTableName,taskDate)  *println*(insertTableSql )      sparkSession.sql("use "+userProfileDbName)  sparkSession.sql(dropTableSql)  sparkSession.sql(createTableSql)  sparkSession.sql(insertTableSql)    }   */\*  删表的SQL语句 \*/* def genDropTableSql(tableName :String): String = {  " drop table if exists "+tableName;  }   */\*  建表的SQL语句  \*/* def genCreateTableSql(tagInfoList: List[TagInfo],tableName :String,dbName:String, hdfsStorePath:String): String ={   val columnList: List[String] = tagInfoList.map { tagInfo =>  val columnName: String = tagInfo.tagCode.toLowerCase  columnName + " string"  }   val columnListSql: String = "uid string ," +columnList.mkString(",")  "create table "+tableName+" ( "+columnListSql+") " +  "ROW FORMAT DELIMITED FIELDS TERMINATED BY '\\t' LOCATION '"+hdfsStorePath+"/"+dbName+"/"+tableName.toLowerCase+"/'"   }   }  } |

## 6.5 关于pivot

**pivot** ，Spark-sql 、Oracle特有关键词，即**旋转**，将指列的字段值，旋转成为多个列。并且可以指定某些列成为旋转列的聚合值。

比如：

### 6.5.1 例一

|  |
| --- |
| CREATE TABLE test\_order\_info  (uname STRING, product STRING, age INT, city string, amount decimal);  INSERT INTO test\_order\_info VALUES  ( 'zhang3' , 'tv' , 22 , 'bj', 3000),  ( 'li4', 'notebook', 41, 'bj', 8000),  ( 'wang5', 'phone', 32, 'sh', 4000),  ( 'zhao6', 'notebook', 22, 'sz', 3000),  ( 'zhang3', 'phone', 22, 'bj', 3000),  ( 'li4', 'tv', 41, 'sz', 4000) ; |

求

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| uname | age | tv\_amount | notebook\_amount | phone\_amount | … |
| zhang3 | 22 | ?? | ?? | ?? |  |
| li4 |  |  |  |  |  |
| wang5 |  |  |  |  |  |
| zhao6 |  |  |  |  |  |
| … |  |  |  |  |  |

### 6.5.2原理说明

把整个表整理成3种列：维度列、旋转列、聚合列

**格式：**

select \* from tablename pivot ( sum(聚合列) as 列标识 for 旋转列 in( 旋转列值1 ,旋转列值2,旋转列值3) )

除了旋转列和聚合列，默认都是维度列,如果存在这三种以外的字段，需要提前用子查询去除。

实现sql

|  |
| --- |
| select \* from test\_order\_info pivot ( sum(amount) as amount for product in ('tv','notebook','phone' )) |

### 6.5.3 例二

求

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| product | bj\_amount | bj\_avg\_age | sh\_amount | sh\_avg\_age | sz\_amount | sz\_avg\_age | … |
| notebook | ?? | ?? | ?? | ?? | ?? | ?? |  |
| phone |  |  |  |  |  |  |  |
| tv |  |  |  |  |  |  |  |
| … |  |  |  |  |  |  |  |

实现sql

|  |
| --- |
| select \* from (select product,age,city,amount from test\_order\_info ) oi  pivot ( sum(amount) as amount ,avg(age) as age for city in ('bj','sh','sz' )) |

测试需要使用sparksql 。 hive不支持pivot.

|  |
| --- |
| spark-sql --hiveconf hive.cli.print.header=true |

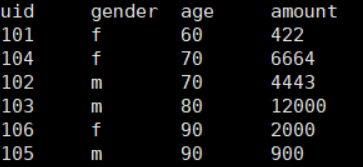
### 6.5.4 例三

|  |
| --- |
| create table test\_user\_tags (uid string, tag\_code STRING, tag\_value STRING);  INSERT INTO test\_user\_tags VALUES  ( '101','gender' ,'f' ),  ( '102', 'gender', 'm' ),  ( '103', 'gender', 'm' ),  ( '104', 'gender', 'f' ),  ( '105', 'gender', 'm' ),  ( '106', 'gender', 'f' ),  ( '101','age' ,'60' ),  ( '102', 'age', '70' ),  ( '103', 'age', '80' ),  ( '104', 'age', '70' ),  ( '105', 'age', '90' ),  ( '106', 'age', '90' ) ,  ( '101','amount' ,'422' ),  ( '102', 'amount', '4443' ),  ( '103', 'amount', '12000' ),  ( '104', 'amount', '6664' ),  ( '105', 'amount', '900' ),  ( '106', 'amount', '2000' ) ; |

求：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UID | gender | age | amount | … |
| 101 | ?? | ?? | ?? |  |
| 102 |  |  |  |  |
| 103 |  |  |  |  |
| … |  |  |  |  |

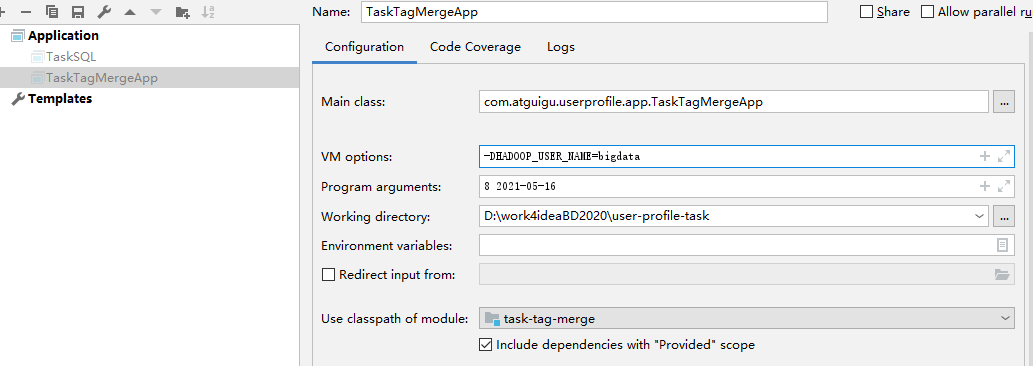
|  |
| --- |
| select \* from test\_user\_tags pivot ( concat\_ws(',',collect\_list(tag\_value)) as tv for tag\_code in ('gender','age','amount' )) |

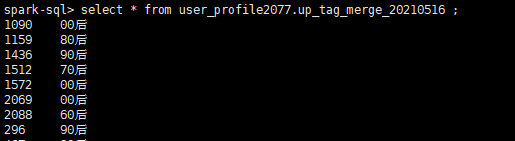


### 6.6 生成insert select语句

|  |
| --- |
| /\*  生成表insert select的SQL语句  \*/  // insert overwrite table mergeTableName  // select \* from  // (  // select uid, tag\_value , 'tag\_oamount' tag\_name from tag\_oamount  // union all  // select uid, tag\_value , 'tag\_usegroup' tag\_name from tag\_usegroup  // ) tags  // pivot ( concat\_ws(',',collect\_list(tag\_value)) as v for tag\_name in ('tag\_oamount','tag\_usegroup' ));  def genInsertTableSql(tagInfoList: List[TagInfo],tableName :String,taskDate:String): String ={  val tagValueTableList: List[String] = tagInfoList.map(tagInfo => {  "select uid,tag\_value,'" + tagInfo.tagCode + "' tag\_name from " + tagInfo.tagCode.toLowerCase +" where dt='"+taskDate+"'"  })  val tagsSql: String = tagValueTableList.mkString(" union all ")   val tagNameList: List[String] = tagInfoList.map(tagInfo => {  "'"+tagInfo.tagCode+"'"  })  val tagNameSql: String = tagNameList.mkString(",")   var insertSql="insert overwrite table " +tableName  var selectSql=" select \* from ("+tagsSql+") " +  "tags pivot ( concat\_ws(',',collect\_list(tag\_value)) as v for tag\_name in ("+tagNameSql+"))"  insertSql+selectSql     } |

测试





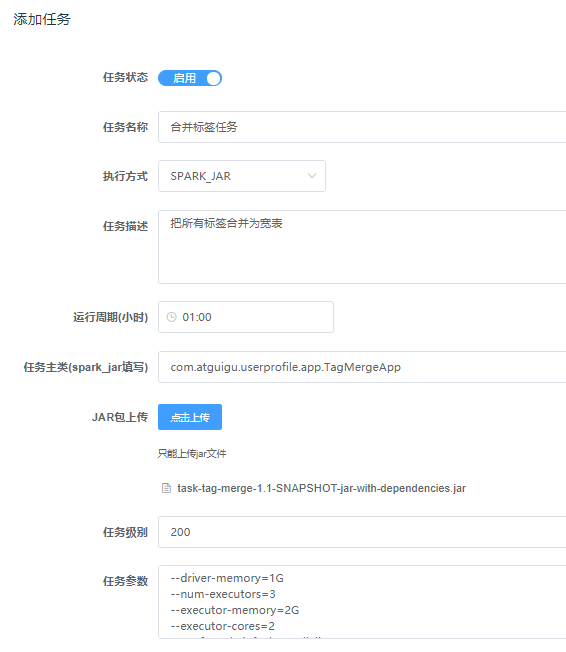
## 6.7 任务发布

因为这个合并宽表任务是要在所有标签任务执行之后完成的。

所以要新增一个“流程任务”，在【流程任务管理】中，点击添加流程任务



填写流程任务



说明：

执行方式选择：SPARK\_JAR

任务级别： 因为要保证在标签任务之后，所以要大于标签任务级别（标签任务级别默认100）就好。

手动调用任务

在【流程任务管理】中，点击手动调度任务。并选择调度日期。



回到【任务进程】中，能够看到除了生成了标签任务的进程，还有合并任务的进程。

调度器（内置15秒）会检查是否有TODO的任务，按照层级进行调度



## 6.8 结果验证

如果看到任务都达到了finish可以取hive中查询每个标签表和标签宽表是否已经生成完成。

