尚硅谷电商用户画像五

版本：V 1.0

张晨

## 

# 第八章 开发任务四：

# 在clickhouse中宽表转换为Bitmap表

## 8.1 为什么用Bitmap?

### 8.1.1 存储成本

假设有个1,2,5的数字集合，如果常规的存储方法，要用3个Int32空间。其中一个Int32就是32位的空间。三个就是3\*32Bit，相当于12个字节。

如果用Bitmap怎么存储呢，只用8Bit（1个字节）就够了。每一位代表一个数，位号就是数值，1标识有，0标识无。如下图：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

这样的一个字节可以存8个整数，每一个数的存储成本实质上是1Bit。

也就是说Bitmap的存储成本是Array[Int32]的1/32,是Array[Int64]的1/64。

好处一： 如果有一个超大的无序且不重复的整数集合，用Bitmap的存储成本是非常低的。

### 8.1.2 天然去重

好处二： 因为每个值都只对应唯一的一个位置，不能存储两个值，所以Bitmap结构可以天然去重。

### 8.1.3 快速定位

如果我有一个需求，比如想判断数字“3”是否存在于该集合中。若是传统的数字集合存储，那就要逐个遍历每个元素进行判断，时间复杂度为O(N)。

但是若是Bitmap存储只要查看对应的下标数的值是0还是1即可，时间复杂度为O(1)。

查询3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | →3 | 2 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

好处三：非常方便快速的查询某个元素是否在集合中。

### 8.1.4 集合间计算

如果我有另一个集合2、3、7，我想查询这两个集合的交集。

传统方式[1,2,5]与[2,3,7] 取交集就要两层循环遍历。

而Bitmap只要把00100110和10001100进行与操作就行了。而计算机做与、或、非、异或 等等操作是非常快的。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

&

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

=

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

好处四：集合与集合之间的运行非常快。

### 8.1.5 优势场景

综上，Bitmap非常适合的场景：

1. 海量数据的压缩存储
2. 去重存储
3. 判断值存在于集合
4. 集合之间的交并差

### 8.1.6 局限性

当然这种方式也有局限性：

1. 只能存储正整数字而不是字符串
2. 存储的值必须是无序不重复
3. 不适合存储稀疏的集合，比如一个集合存了三个数[5,1230000,88880000] 这三个数，用Bitmap存储的话其实就不太划算。（但是clickhouse使用的RoaringBitmap，优化了这个稀疏问题。）

## 8.2 Bitmap在用户分群中的应用

### 现状：

首先，如下是用户的标签宽表，

|  |  |  |  |
| --- | --- | --- | --- |
| 用户 | 性别 | 年龄 | 偏好 |
| 1 | 男 | 90后 | 数码 |
| 2 | 男 | 70后 | 书籍 |
| 3 | 男 | 90后 | 美食 |
| 4 | 女 | 80后 | 书籍 |
| 5 | 女 | 90后 | 美食 |

如果想根据标签划分人群，比如：90后 + 偏好美食。

### 传统解决方案：

那么无非对列值进行遍历筛选，如果优化也就是列上建立索引，但是当这张表有1000个标签列时，如果要索引生效并不是每列有索引就行，要每种查询组合建一个索引才能生效，索引数量相当于1000个列排列组合的个数，这显然是不可能的。

### 更好的方案

那么更好的办法是按字段重组成Bitmap。

|  |  |  |
| --- | --- | --- |
| 年龄 | Array | Bitmap |
| 90后 | 1,3,5 | 00101010 |
| 80后 | 4 | 00010000 |
| 70后 | 2 | 00000100 |

|  |  |
| --- | --- |
| 性别 | Bitmap |
| 男 | 1,2,3 |
| 女 | 4,5 |

|  |  |  |
| --- | --- | --- |
| 偏好 | Bitmap |  |
| 数码 | 1 |  |
| 美食 | 3,5 | 00101000 |
| 书籍 | 2,4 |  |

如果能把数据调整成这样的结构，想进行条件组合,那就简单了.

比如: **[美食] + [90后] = Bitmap[3,5] & Bitmap[1,3,5] = 3,5** 这个计算速度相比宽表条件筛选是非常非常快的。

## 8.3 在clickhouse中使用Bitmap表

还是以上面的表举例：

### 8.3.1 建表和数据

|  |
| --- |
| create table user\_tag\_merge  ( uid UInt64,  gender String,  agegroup String,  favor String  )engine=MergeTree()  order by (uid); |

模拟数据

|  |
| --- |
| insert into user\_tag\_merge values(1,'M','90后','sm');  insert into user\_tag\_merge values(2,'M','70后','sj');  insert into user\_tag\_merge values(3,'M','90后','ms');  insert into user\_tag\_merge values(4,'F','80后','sj');  insert into user\_tag\_merge values(5,'F','90后','ms'); |

Bitmap表

|  |
| --- |
| create table user\_tag\_value\_string  (  tag\_code String,  tag\_value String ,  us AggregateFunction(groupBitmap,UInt64)  )engine=AggregatingMergeTree()  partition by (tag\_code)  order by (tag\_value); |

Bitmap表必须选择AggregatingMergeTree引擎。

对应的Bitmap字段，必须是AggregateFunction(groupBitmap,UInt64)，groupBitmap标识数据的聚合方式，UInt64标识最大可存储的数字长度。

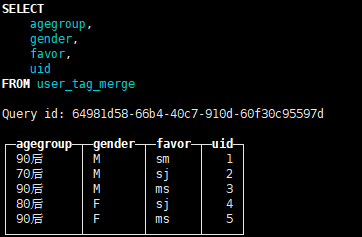
业务结构上，稍作了调整。把不同的标签放在了同一张表中，但是因为根据tag\_code进行了分区，所以不同的标签实质上还是物理分开的。

|  |  |  |
| --- | --- | --- |
| 标签 | 标签值 | Bitmap |
| 年龄段 | 90后 | 1,3,5 |
| 年龄段 | 80后 | 4 |
| 年龄段 | 70后 | 2 |
| 性别 | 男 | 1,2,3 |
| 性别 | 女 | 4,5 |
| 偏好 | 数码 | 1 |
| 偏好 | 美食 | 3,5 |
| 偏好 | 书籍 | 2,4 |

### 8.3.2处理步骤

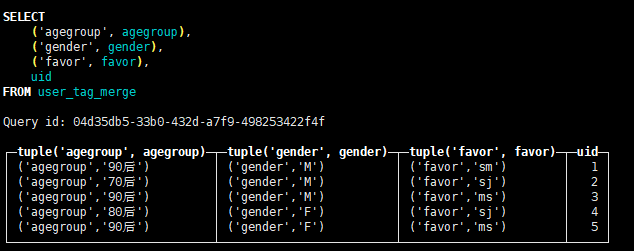
1）首先第一步要把已有的宽表数据保存进

|  |
| --- |
| select agegroup , gender , favor ,uid  from user\_tag\_merge |



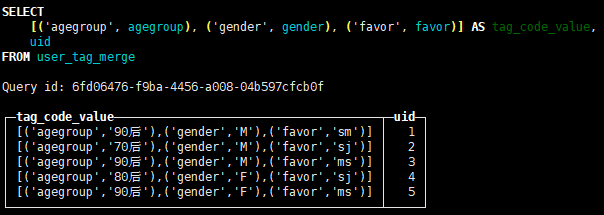
2）每个值前面，补上字段名,用()组合成元组

|  |
| --- |
| select ('agegroup', agegroup ) ,  ('gender',gender ) , ('favor',favor ) ,uid  from user\_tag\_merge |



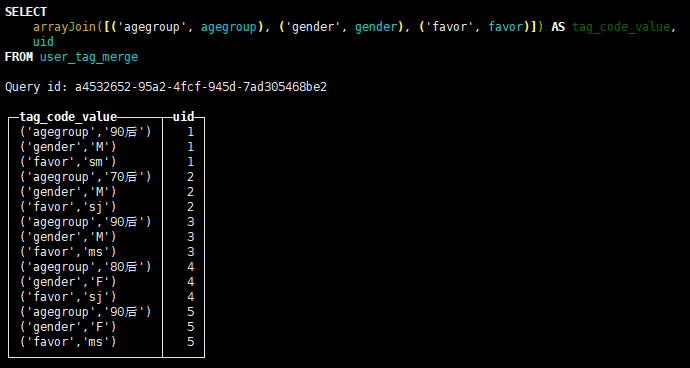
3）每个列用[]拼接成数组

|  |
| --- |
| select [ ('agegroup', agegroup ) ,  ('gender',gender ) ,  ('favor',favor )] tag\_code\_value ,uid  from user\_tag\_merge |



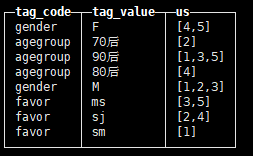
4）用arrayJoin炸开，类似于hive中的explode

|  |
| --- |
| SELECT  arrayJoin([('agegroup', agegroup), ('gender', gender), ('favor', favor)]) AS tag\_code\_value,  uid  FROM user\_tag\_merge |



5） 把元组中的字段名和字段值拆开，并用这两个作为维度聚合uid

|  |
| --- |
| SELECT  tag\_code\_value.1 AS tag\_code,  tag\_code\_value.2 AS tag\_value,  **groupArray**(uid) AS us  FROM  (  SELECT  arrayJoin([('agegroup', agegroup), ('gender', gender), ('favor', favor)]) AS tag\_code\_value,  uid  FROM user\_tag\_merge  ) AS tv  GROUP BY  tag\_code\_value.1,  tag\_code\_value.2 |

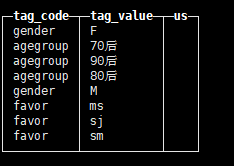


6） 这已经和我们要求的结果非常接近了。只不过关于用户id的聚合，groupArray实现的是拼接成用户id的数组，而我们想要的聚合是，聚合成一个Bitmap。

那只要改一下聚合函数就可以了。

把**groupArray 替换成 groupBitmapState**

|  |
| --- |
| SELECT  tag\_code\_value.1 AS tag\_code,  tag\_code\_value.2 AS tag\_value,  **groupBitmapState** (uid) AS us  FROM  (  SELECT  arrayJoin([('agegroup', agegroup), ('gender', gender), ('favor', favor)]) AS tag\_code\_value,  uid  FROM user\_tag\_merge  ) AS tv  GROUP BY  tag\_code\_value.1,  tag\_code\_value.2 |

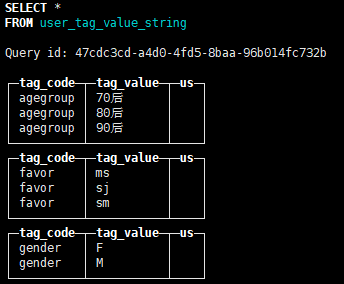


这里聚合成bitmap的列没有显示是正常的，因为bitmap的结构本身无法用正常文本显示。

8）接下来我们可以插入到bitmap表中。

|  |
| --- |
| insert into user\_tag\_value\_string  select tag\_code\_value.1 as tag\_code,tag\_code\_value.2 as tag\_value ,  groupBitmapState( uid ) us  from (  SELECT  arrayJoin([('agegroup', agegroup), ('gender', gender), ('favor', favor)]) AS tag\_code\_value,  uid  FROM user\_tag\_merge  )tv  group by tag\_code\_value.1,tag\_code\_value.2 |

插入完成后，查询依旧不会正常显示



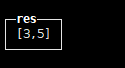
### 8.3.4 对Bitmap进行查询

如何使用这个Bitmap表查询呢？

#### 条件组合查询

比如我想查询[90后]+[美食]的用户

|  |
| --- |
| select **bitmapToArray**(  **bitmapAnd**( ( select us from user\_tag\_value\_string  where tag\_value='ms' and tag\_code='favor' )  , ( select us from user\_tag\_value\_string  where tag\_value='90后' and tag\_code='agegroup' )  )  )as res |



这里首先用条件筛选出us, 每个代表一个Bitmap结构的uid集合，找到两个Bitmap后用bitmapAnd函数求**交集**。 然后为了观察结果用bitmapToArray函数转换成可见的数组。

#### 范围值查询

比如要取 [90后]或者[80后] + [美食]

或者 消费金额大于1000 + [女性]

|  |
| --- |
| select bitmapToArray(  bitmapAnd( ( select groupBitmapMergeState(us) us from user\_tag\_value\_string  where tag\_value='ms' and tag\_code='favor' )  , ( select groupBitmapMergeState(us) from user\_tag\_value\_string  where tag\_value in ('90后','80后') and tag\_code='agegroup' )  )  )as res |

因为查询时，有可能需要针对某一个标签，取多个值，甚至是一个区间范围，那就会涉及多个值的userId集合，因此需要在子查询内部用groupBitmapMergeState进行一次合并，其实就多个集合取并集。

### 8.3.5 函数总结

|  |  |
| --- | --- |
| 函数 |  |
| arrayJoin | 宽表转Bitmap表需要行转列，要用arrayJoin把多列数组炸成行。 |
| groupBitmapState | 把聚合列的数字值聚合成Bitmap的聚合函数 |
| bitmapAnd | 求两个Bitmap值的交集 |
| bitmapOr | 求两个Bitmap值的并集 |
| bitmapXor | 求两个Bitmap值的差集(异或) |
| bitmapToArray | 把Bitmap转换成数值数组 |
| groupBitmapMergeState | 把一列中多个bitmap值进行并集聚合。 （连续值） |

## 8.4 任务目标

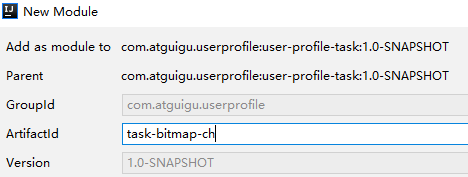
为了能够更快的进行条件查询群体，把clickhouse中的宽表，通过行转列、聚合能操作转入Bitmap表。

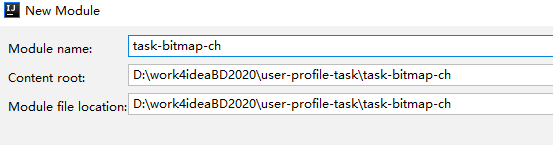
## 8.5 步骤分析

1. 读取标签定义表，用于获得标签名称和标签类型。
2. 建立四种不同数据类型的标签值表
3. 根据标签类型的不同要写入到四种不同的Bitmap表中。

## 8.6 代码实现

### 8.6.1 搭建模块





### 8.6.2 pom.xml

|  |
| --- |
| <**dependencies**>   <**dependency**>  <**groupId**>com.atguigu.userprofile</**groupId**>  <**artifactId**>task-common</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>    </**dependencies**>  <**build**>  <**plugins**>  *<!-- 该插件用于将Scala代码编译成class文件 -->* <**plugin**>  <**groupId**>net.alchim31.maven</**groupId**>  <**artifactId**>scala-maven-plugin</**artifactId**>  <**version**>3.4.6</**version**>  <**executions**>  <**execution**>  *<!-- 声明绑定到maven的compile阶段 -->* <**goals**>  <**goal**>compile</**goal**>  <**goal**>testCompile</**goal**>  </**goals**>  </**execution**>  </**executions**>  </**plugin**>   <**plugin**>  <**groupId**>org.apache.maven.plugins</**groupId**>  <**artifactId**>maven-assembly-plugin</**artifactId**>  <**version**>3.0.0</**version**>  <**configuration**>  <**descriptorRefs**>  <**descriptorRef**>jar-with-dependencies</**descriptorRef**>  </**descriptorRefs**>  </**configuration**>  <**executions**>  <**execution**>  <**id**>make-assembly</**id**>  <**phase**>package</**phase**>  <**goals**>  <**goal**>single</**goal**>  </**goals**>  </**execution**>  </**executions**>  </**plugin**>  </**plugins**> </**build**> |

### 8.6.3 建立数据库

|  |
| --- |
| create table user\_tag\_value\_string  (  tag\_code String,  tag\_value String ,  us AggregateFunction(groupBitmap,UInt64),  dt String  )engine=AggregatingMergeTree()  partition by (dt, tag\_code)  order by ( tag\_value);        create table user\_tag\_value\_long  (  tag\_code String,  tag\_value UInt64 ,  us AggregateFunction(groupBitmap,UInt64),  dt String  )engine=AggregatingMergeTree()  partition by (dt, tag\_code)  order by ( tag\_value);      create table user\_tag\_value\_decimal  (  tag\_code String,  tag\_value Decimal64(2) ,  us AggregateFunction(groupBitmap,UInt64),  dt String  )engine=AggregatingMergeTree()  partition by (dt, tag\_code)  order by ( tag\_value);    create table user\_tag\_value\_date  (  tag\_code String,  tag\_value Date ,  us AggregateFunction(groupBitmap,UInt64),  dt String  )engine=AggregatingMergeTree()  partition by (dt, tag\_code)  order by ( tag\_value); |

### 8.6.4 主程序

|  |
| --- |
| **object** TaskBitmapChApp {   **def** main(args: Array[String]): Unit = {  ***// 全称只使用driver操作  // 读取tag的元数据  // 根据数据类型进行划分 并准备存储在不同的标签结构表中* val** taskId: String = args(0)  **val** taskDate: String = args(1)  **val** tableName=**"up\_tag\_merge\_"** + taskDate.replace(**"-"**,**""**)   **val** sparkConf: SparkConf = **new** SparkConf().setAppName(**"tag\_bitmap\_app"**).setMaster(**"local[\*]"**)  **val** sparkSession: SparkSession = SparkSession.*builder*().config(sparkConf).enableHiveSupport().getOrCreate()  **val** properties: Properties = MyPropertiesUtil.*load*(**"config.properties"**)   **val** dbNameCH = properties.getProperty(**"user-profile.dbname.ch"**)  **val** clickhouseUrl = properties.getProperty(**"clickhouse.url"**)   **val** tagInfoList: List[TagInfo] = TagInfoDAO.*getTagListOnTask*()   **val** stringTagList =**new** ListBuffer[TagInfo]  **val** decimalTagList =**new** ListBuffer[TagInfo]  **val** longTagList =**new** ListBuffer[TagInfo]  **val** dateTagList =**new** ListBuffer[TagInfo]   **for** (tagInfo <- tagInfoList ) {  tagInfo.tagValueType **match** {  **case** ConstCode.*TAG\_VALUE\_TYPE\_STRING* => stringTagList.append(tagInfo)  **case** ConstCode.*TAG\_VALUE\_TYPE\_DECIMAL* => decimalTagList.append(tagInfo)  **case** ConstCode.*TAG\_VALUE\_TYPE\_LONG* => longTagList.append(tagInfo)  **case** ConstCode.*TAG\_VALUE\_TYPE\_DATE* => dateTagList.append(tagInfo)  }  }   ***//不同类型的数据进入不同的表*** *insertTags*(tableName,**"user\_tag\_value\_string"**,  stringTagList.toList,clickhouseUrl,taskDate)  *insertTags*(tableName,**"user\_tag\_value\_long"**,  longTagList.toList,clickhouseUrl,taskDate)  *insertTags*(tableName,**"user\_tag\_value\_decimal"** ,decimalTagList.toList,clickhouseUrl,taskDate)  *insertTags*(tableName,**"user\_tag\_value\_date"**,  dateTagList.toList,clickhouseUrl,taskDate)   }   ***// 列转行  // insert into user\_tag\_value\_string  // select tag\_code\_value.1 as tag\_code,  // tag\_code\_value.2 as tag\_value ,  // groupBitmapState( uid ) us  // from (  // SELECT  // arrayJoin([('agegroup', agegroup),  // ('gender', gender),  // ('favor', favor)]) AS tag\_code\_value,  // uid  // FROM up\_user\_tag\_merge  // )tv  // group by tag\_code\_value.1,tag\_code\_value.2* def** insertTags(fromTableName:String ,toTableName:String,tagList:List[TagInfo],clickhouseUrl:String,taskDate:String): Unit ={  **if**(tagList.size>0){  **val** deleteSql=**s" ALTER TABLE $**toTableName **DELETE WHERE dt='$**taskDate**'"** MyClickHouseUtil.*executeSql*(deleteSql,clickhouseUrl )   **val** tagCodeValueList: List[String] = tagList.map{tagInfo=> **" ('"**+tagInfo.tagCode+**"',"**+tagInfo.tagCode.toLowerCase+**" ) "** }  **val** tagCodeValueSql: String = tagCodeValueList.mkString(**","**)  **val** selectSql=**s"""| select tag\_code\_value.1 tag\_code,  | tag\_code\_value.2 tag\_value,  | groupBitmapState(uid) us ,'$**taskDate**'  | from  | (  | select arrayJoin( [$**tagCodeValueSql**] ) tag\_code\_value ,  | uid from $**fromTableName **| )tagCV  | where tag\_code\_value.2 <>''  | group by tag\_code\_value.1, tag\_code\_value.2  """**.stripMargin  **val** insertSql=**s"insert into $**toTableName **"** +selectSql  *println*(insertSql)  MyClickHouseUtil.*executeSql*(insertSql,clickhouseUrl )  }  }   } |

## 8.7 打包发布

把程序打包

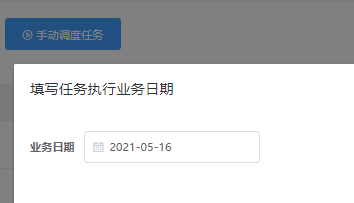
在【流程任务管理】中增加任务





## 8.8 调度任务测试





执行【手动调度任务】，选择数仓的业务日期

## 8.9 验证结果

带任务执行完成后查看结果

|  |
| --- |
| SELECT  tag\_code,  tag\_value,  bitmapToArray(us)  FROM user\_tag\_value\_string |

