# 有效线索主题看板

## 学习目标

了解有效线索主题看板需求

了解Hive索引的用法

掌握Row Group Index的用法

掌握Bloom Filter Index的用法

能够采集有效线索全量数据

能够使用Hive进行并行操作

掌握Hive常用的判断函数

能够编写有效线索指标的DWD清洗转换SQL

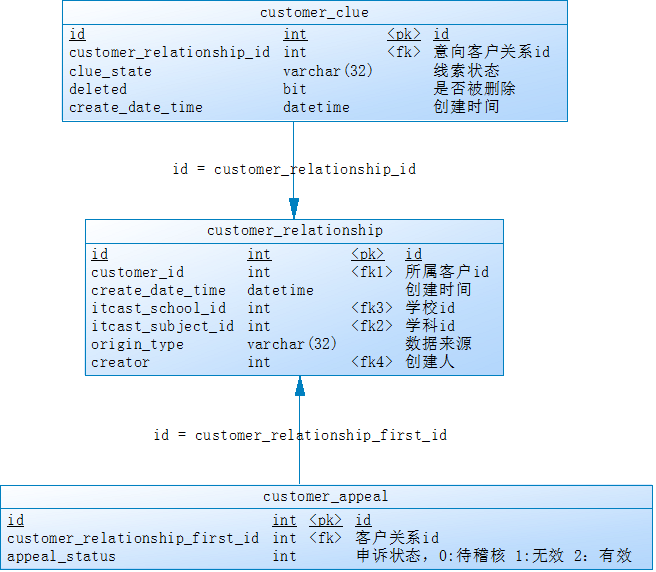
能够编写有效线索指标的DWM中间层SQL

能够编写有效线索指标的DWS业务层SQL

能够导出有效线索指标结果到Mysql

掌握增量数据分析的过程

## 主题需求



### 有效线索转化率

说明：统计期内，访客咨询产生的有效线索的占比。有效线索量 / 咨询量，有效线索指的是拿到电话且电话有效。

展现：线状图。双轴：有效线索量、有效线索转化率。

条件：年、月、线上线下

维度：年、月、线上线下

指标：访客咨询率=有效线索量/咨询量

粒度：天

数据来源：客户管理系统的customer\_clue线索表、customer\_relationship意向表、customer\_appeal申诉表；咨询系统的web\_chat\_ems访问咨询表

SQL：

1. --咨询量(暂时以2019年7月的数据为例)：
2. **SELECT**
3. count(1)
4. **FROM**
5. web\_chat\_ems\_2019\_07
6. **WHERE**
7. msg\_count >= 1
8. AND create\_time >= '2019-07-01'
9. AND create\_time <= '2019-07-15 23:59:59';
10. --有效线索量：
11. **SELECT**
12. count(1)
13. **FROM**
14. *customer\_clue* cc
15. LEFT JOIN *customer\_relationship* cr **ON** cc.customer\_relationship\_id = cr.id
16. **WHERE**
17. cc.clue\_state IN (
18. 'VALID\_NEW\_CLUES', --新客户新线索
19. 'VALID\_PUBLIC\_NEW\_CLUE' --老客户新线索
20. )
21. AND cc.customer\_relationship\_id NOT IN (
22. **SELECT**
23. ca.customer\_relationship\_first\_id
24. **FROM** --投诉表，投诉成功的数据为无效线索
25. customer\_appeal ca
26. **WHERE**
27. ca.appeal\_status = 1 AND ca.customer\_relationship\_first\_id != 0
28. )
29. AND cr.origin\_type IN ('NETSERVICE','PRESIGNUP') --线上（排除挖掘录入量）
30. AND ! cc.deleted
31. AND cc.create\_date\_time <= '2019-07-01'
32. AND cc.create\_date\_time <= '2019-07-15 23:59:59';

### 有效线索转化率时间段趋势

说明：统计期内，1-24h之间，每个时间段的有效线索转化率。横轴：1-24h，间隔为1h，纵轴：每个时间段的有效线索转化率。

展现：线状图

条件：天、线上线下

维度：天、线上线下

指标：某小时的总有效线索转化率

粒度：区间内小时段（区间内同一个时间点的总有效线索转化率）

数据来源：客户管理系统的customer\_clue线索表、customer\_relationship意向表、customer\_appeal申诉表；咨询系统的web\_chat\_ems访问咨询表

SQL：同上

### 有效线索量

说明：统计期内，新增的咨询客户中，有效线索的数量。

展现：线状图。

条件：年、月、线上线下

维度：年、月、线上线下

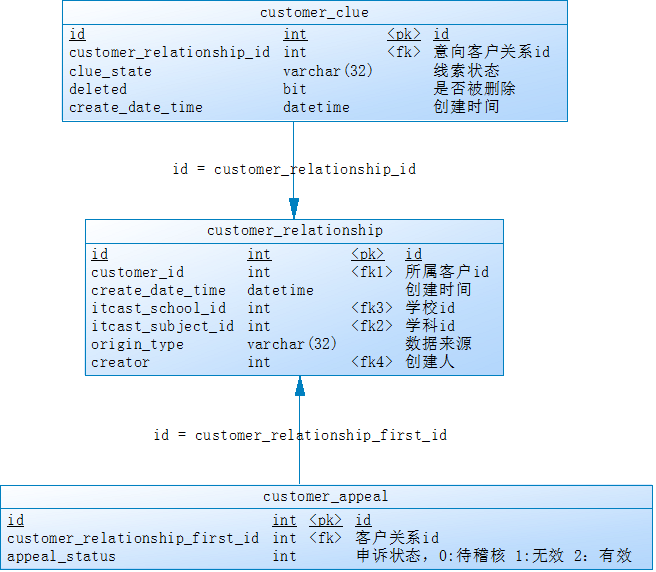
指标：有效线索的数量

粒度：天

数据来源：客户管理系统的customer\_clue线索表、customer\_relationship意向表、customer\_appeal申诉表

SQL：同上

### 原始数据结构



有效线索指标的原始数据为客户管理系统的customer\_clue线索表和customer\_relationship意向客户表。

customer\_clue是线索事实表，customer\_relationship表主要是用来判断数据来源为线上还是线下。

这两张表在意向客户指标的ODS层已经抽取过，此处可以直接复用。

customer\_appeal表是线索申诉表，主要用来判断客户线索被投诉无效。

测试数据：已包含在意向客户主题测试sql中，无需重复导入。

|  |
| --- |
| CREATE TABLE `customer\_appeal` (  `id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '主键',  `customer\_relationship\_first\_id` int(11) NOT NULL COMMENT '第一条客户关系id',  `employee\_id` int(11) DEFAULT NULL COMMENT '申诉人',  `employee\_name` varchar(64) COLLATE utf8\_bin DEFAULT NULL COMMENT '申诉人姓名',  `employee\_department\_id` int(11) DEFAULT NULL COMMENT '申诉人部门',  `employee\_tdepart\_id` int(11) DEFAULT NULL COMMENT '申诉人所属部门',  `appeal\_status` int(1) NOT NULL COMMENT '申诉状态，0:待稽核 1:无效 2：有效',  `audit\_id` int(11) DEFAULT NULL COMMENT '稽核人id',  `audit\_name` varchar(255) COLLATE utf8\_bin DEFAULT NULL COMMENT '稽核人姓名',  `audit\_department\_id` int(11) DEFAULT NULL COMMENT '稽核人所在部门',  `audit\_department\_name` varchar(255) COLLATE utf8\_bin DEFAULT NULL COMMENT '稽核人部门名称',  `audit\_date\_time` datetime DEFAULT NULL COMMENT '稽核时间',  `create\_date\_time` datetime DEFAULT *CURRENT\_TIMESTAMP* COMMENT '创建时间（申诉时间）',  `update\_date\_time` timestamp NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP COMMENT '更新时间',  `deleted` bit(1) NOT NULL DEFAULT b'0' COMMENT '删除标志位',  `tenant` int(11) NOT NULL DEFAULT '0',  PRIMARY KEY (`id`),  KEY `id` (`id`,`appeal\_status`) USING BTREE,  KEY `idx\_customer\_relationship\_first\_id` (`customer\_relationship\_first\_id`) USING BTREE ) ENGINE=InnoDB AUTO\_INCREMENT=2012358 DEFAULT CHARSET=utf8 COLLATE=utf8\_bin; |

## 建模分析

### 指标和维度

根据主题需求，我们来进行指标和维度的提取：

有效线索转化率，公式：有效线索量/咨询量。其中分母咨询量，在我们之前的指标中已经统计过，在这里可以直接使用。因此只需要统计分子有效线索量即可。有效线索指的是拿到电话且电话有效。维度虽然只有年、月、线上线下，但是因为数据粒度具体到天，所以维度统计时也要增加天。维度：年、月、天、线上线下。

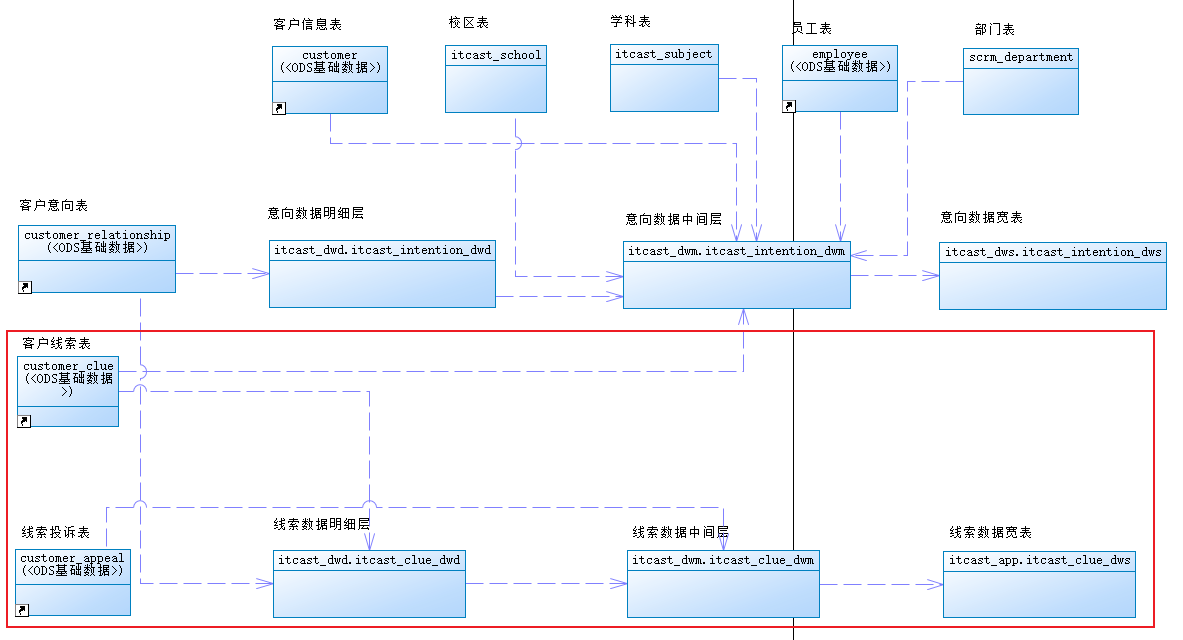
有效线索转化率时间段趋势，主要是针对小时段区间内的数据进行统计，跨天数据无需去重。因此维度统计时需要统计到小时维度。维度：天、小时、线上线下。

有效线索量，即有效线索转化率的分子，数据粒度为天。但此处特别指明是新增客户的有效线索，因此统计维度中需要包含此线索所属的客户是新客户还是老客户。维度：年、月、天、线上线下、新旧客户。

总结：

* 指标：有效线索量；
* 维度：年、月、天、小时、线上线下、新旧客户。

### 分层设计



1. 最终需要统计的数据维度：年、月、天、小时、线上线下、新旧客户；
2. 将维度分为三类：时间维度（年、月、天、小时）、数据来源（线上线下）和新旧客户类型；
3. ODS层原始数据包括：主表有效线索、意向客户表（用来判断是新客户还是老客户）；
4. 有效线索数据同样的数据只能录入一次，因此不存在去重的问题，所以可以使用DWM中间层来进行维度关联，并做少量聚合，可被DWS层调用以提高计算速度；
5. DWS层在DWM层的基础上进行统计，得出数据集市；
6. 因为使用的是帆软BI自定义可视化展现，所以不再提供细分的APP层，直接将DWS数据集市导出到OLAP应用的mysql中即可。

## 实现

### 建模

#### 指标和维度

指标：有效线索量；

维度：

* 时间维度：年、月、天、小时
* 数据来源：线上、线下
* 客户类型：新客户线索、老客户线索

#### 事实表和维度表

事实表：customer\_clue线索表

维表：

1. customer\_relationship意向客户表，主要为了判断数据来源为线上还是线下，也是意向客户指标的事实表。
2. customer\_appeal线索申诉表，主要为了判断线索数据是否有效。

#### Hive索引

Hive支持索引，但是Hive的索引与关系型数据库中的索引并不相同，比如，Hive不支持主键或者外键。

Hive索引可以建立在表中的某些列上，以提升一些操作的效率，例如减少MapReduce任务中需要读取的数据块的数量。

在可以预见到分区数据非常庞大的情况下，分桶和索引常常是优于分区的。而分桶由于SMB Join对关联键要求严格，所以并不是总能生效。

##### Hive原始索引

Hive的索引目的是提高Hive表指定列的查询速度。

没有索引时，类似'WHERE tab1.col1 = 10' 的查询，Hive会加载整张表或分区，然后处理所有的rows，但是如果在字段col1上面存在索引时，那么只会加载和处理文件的一部分。

。在每次建立、更新数据后，Hive索引不会自动更新，需要手动进行更新（重建索引以构建索引表），会触发一个mr job

Hive索引使用过程繁杂，而且性能一般，在Hive3.0中已被删除，在工作环境中不推荐优先使用，在分区数量过多或查询字段不是分区字段时，索引可以作为补充方案同时使用。推荐使用ORC文件格式的索引类型进行查询。

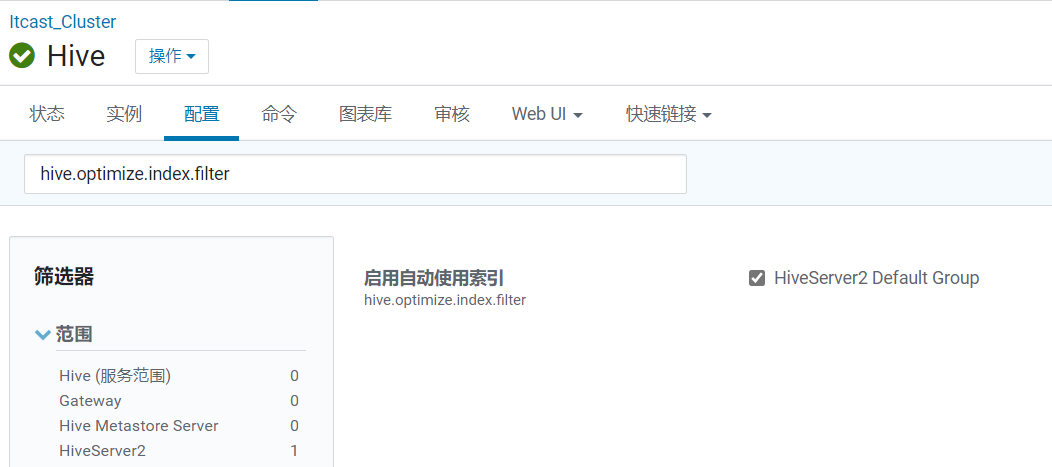
##### Row Group Index

一个ORC文件包含一个或多个stripes(groups of row data)，每个stripe中包含了每个column的min/max值的索引数据，当查询中有<,>,=的操作时，会根据min/max值，跳过扫描不包含的stripes。

而其中为每个stripe建立的包含min/max值的索引，就称为Row Group Index行组索引，也叫min-max Index大小对比索引，或者Storage Index。

在建立ORC格式表时，指定表参数’orc.create.index’=’true’之后，便会建立Row Group Index，需要注意的是，为了使Row Group Index有效利用，向表中加载数据时，必须对需要使用索引的字段进行排序，否则，min/max会失去意义。另外，这种索引主要用于**数值型字段**的查询过滤优化上。

设置**hive.optimize.index.filter**为true，并重启hive



创建表/插入数据：

|  |
| --- |
| CREATE TABLE lxw1234\_orc2 stored AS ORC TBLPROPERTIES (  'orc.compress'='SNAPPY', *-- 开启行组索引* 'orc.create.index'='true' ) AS  SELECT *CAST*(siteid AS INT) AS id,  pcid  FROM lxw1234\_text *-- 插入的数据保持排序* DISTRIBUTE BY id sort BY id; |

查询：

|  |
| --- |
| set hive.optimize.index.filter=true; SELECT *COUNT*(1) FROM lxw1234\_orc1 WHERE id >= 1382 AND id <= 1399; |

##### Bloom Filter Index

在建表时候，通过表参数”orc.bloom.filter.columns”=”pcid”来指定为那些字段建立BloomFilter索引，这样，在生成数据的时候，会在每个stripe中，为该字段建立BloomFilter的数据结构，当查询条件中包含对该字段的**=号过滤**时候，先从BloomFilter中获取以下是否包含该值，如果不包含，则跳过该stripe。

创建：

|  |
| --- |
| CREATE TABLE lxw1234\_orc2 stored AS ORC TBLPROPERTIES (  'orc.compress'='SNAPPY',  'orc.create.index'='true', *-- pcid字段开启BloomFilter索引* "orc.bloom.filter.columns"="pcid" ) AS SELECT *CAST*(siteid AS INT) AS id, pcid FROM lxw1234\_text DISTRIBUTE BY id sort BY id; |

查询：

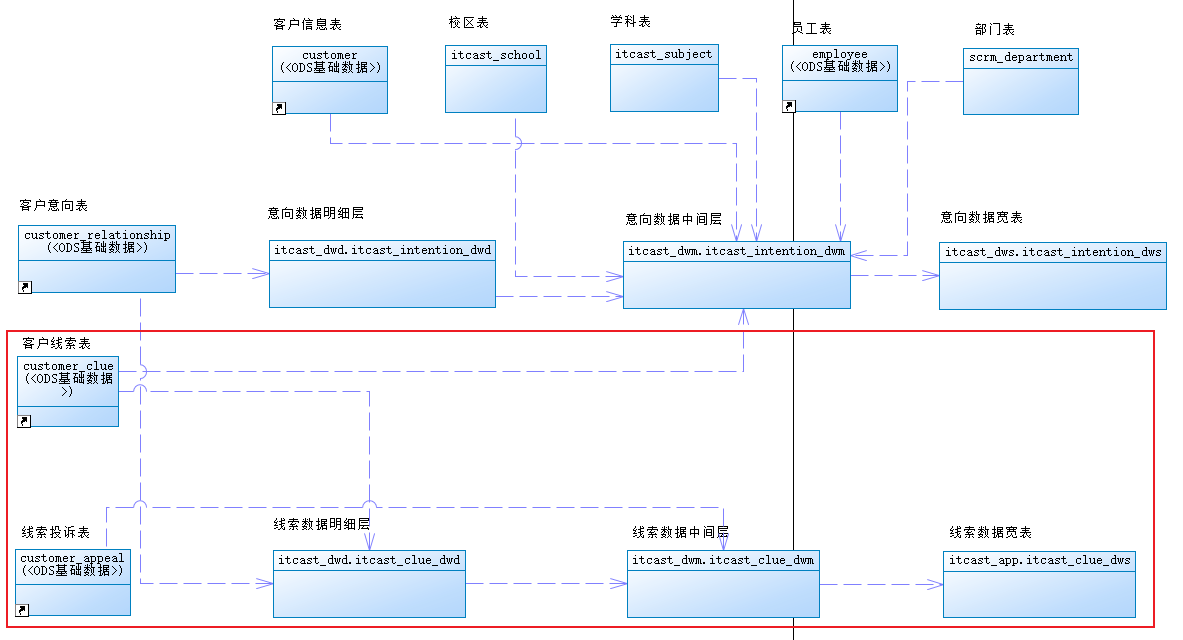
|  |
| --- |
| SET hive.optimize.index.filter=true; SELECT *COUNT*(1) FROM lxw1234\_orc1 WHERE id >= 0 AND id <= 1000  AND pcid IN ('0005E26F0DCCDB56F9041C','A'); |

只有在数据量较大时，使用索引才能带来性能优势。

#### 分层

ODS层可以复用意向客户指标，无需重复创建。

因为线索数据不会重复（不用distinct），所以可以采用DWM中间层过度。



##### ODS

customer\_clue线索表和customer\_relationship意向客户表复用意向客户指标的ODS层。

写入时压缩生效

|  |
| --- |
| set hive.exec.orc.compression.strategy=COMPRESSION; |

###### customer\_appeal线索申诉表

分桶表，sqoop抽取数据是不支持的，但是索引表是支持的。

|  |
| --- |
| CREATE EXTERNAL TABLE IF NOT EXISTS itcast\_ods.`customer\_appeal` (  `id` int COMMENT 'customer\_appeal\_id',  `customer\_relationship\_first\_id` int COMMENT '第一条客户关系id',  `employee\_id` int COMMENT '申诉人',  `employee\_name` STRING COMMENT '申诉人姓名',  `employee\_department\_id` int COMMENT '申诉人部门',  `employee\_tdepart\_id` int COMMENT '申诉人所属部门',  `appeal\_status` int COMMENT '申诉状态，0:待稽核 1:无效 2：有效',  `audit\_id` int COMMENT '稽核人id',  `audit\_name` STRING COMMENT '稽核人姓名',  `audit\_department\_id` int COMMENT '稽核人所在部门',  `audit\_department\_name` STRING COMMENT '稽核人部门名称',  `audit\_date\_time` STRING COMMENT '稽核时间',  `create\_date\_time` STRING COMMENT '创建时间（申诉时间）',  `update\_date\_time` STRING COMMENT '更新时间',  `deleted` STRING COMMENT '删除标志位',  `tenant` int COMMENT '租户id')  comment '客户申诉表' PARTITIONED BY(start\_time STRING) ROW FORMAT DELIMITED  FIELDS TERMINATED BY '\t' stored as orc TBLPROPERTIES ('orc.compress'='SNAPPY','orc.create.index'='true','orc.bloom.filter.columns'='appeal\_status,customer\_relationship\_first\_id'); |

##### DWD

过滤已删除数据(deleted)、线索状态clue\_state为空的数据；

此处回顾SMB Join的用法：关联意向表的customer\_relationship\_id如果为空则转换为-1。

customer\_clue线索表、customer\_relationship意向表可以获取并转换得到新老客户、线上线下。在意向客户主题已经建立过这两个ODS表，类似的关联查询可以通过分桶提高查询效率，我们在这两个字段上分别建立分桶，而且要保证他们的排序和分桶的数量是一致的。

|  |
| --- |
| CREATE TABLE IF NOT EXISTS itcast\_dwd.itcast\_clue\_dwd (  `id` STRING COMMENT '线索id',  `customer\_relationship\_id` int COMMENT '客户关系id',  `origin\_type\_stat` STRING COMMENT '数据来源:0.线下；1.线上',  `for\_new\_user` STRING COMMENT '0:未知；1：新客户线索；2：旧客户线索',  `deleted` STRING COMMENT '是否删除',  `create\_date\_time` BIGINT COMMENT '创建时间',  `hourinfo` STRING COMMENT '小时信息' ) comment '客户申诉dwd表'  PARTITIONED BY(yearinfo STRING,monthinfo STRING,dayinfo STRING)  clustered by(customer\_relationship\_id) sorted by(customer\_relationship\_id) into 10 buckets ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' stored as orcfile TBLPROPERTIES ('orc.compress'='SNAPPY','orc.create.index'='true','orc.bloom.filter.columns'='customer\_relationship\_id'); |

##### DWM

DWM层过滤投诉数据，并以小时进行统计。

分区并不是越多越好，统计后数据量变小，可以年作为分区。

|  |
| --- |
| CREATE TABLE IF NOT EXISTS itcast\_dwm.itcast\_clue\_dwm (  `clue\_nums` STRING COMMENT '根据id聚合',  `origin\_type\_stat` STRING COMMENT '数据来源:0.线下；1.线上',  `for\_new\_user` STRING COMMENT '0:未知；1：新客户线索；2：旧客户线索',  `hourinfo` STRING COMMENT '小时信息',  `dayinfo`STRING COMMENT '天信息',  `monthinfo` STRING COMMENT '月信息' ) comment '客户申诉dwm表' PARTITIONED BY(yearinfo STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' stored as orcfile TBLPROPERTIES ('orc.compress'='SNAPPY'); |

##### DWS

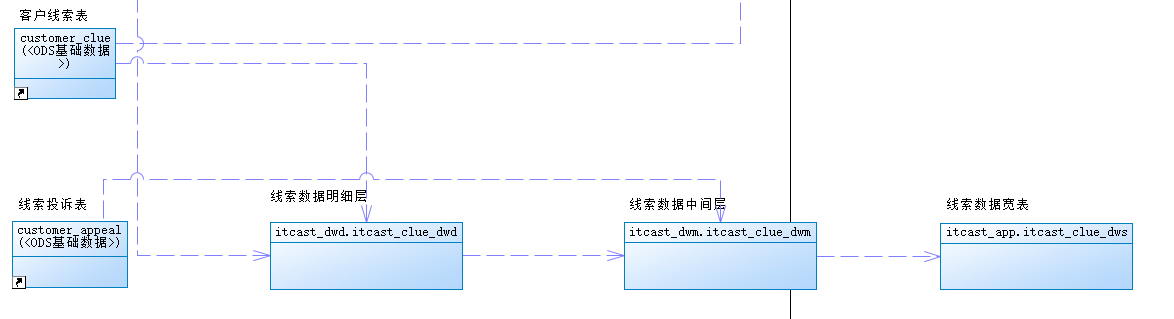
针对不同的时间维度进行统计，方便OLAP系统使用。

|  |
| --- |
| CREATE TABLE IF NOT EXISTS itcast\_dws.itcast\_clue\_dws (  `clue\_nums` INT COMMENT '根据id聚合',  `origin\_type\_stat` STRING COMMENT '数据来源:0.线下；1.线上',  `for\_new\_user` STRING COMMENT '0:未知；1：新客户线索；2：旧客户线索',  `hourinfo` STRING COMMENT '小时信息',  `dayinfo`STRING COMMENT '天信息',  `monthinfo` STRING COMMENT '月信息',  `time\_type` STRING COMMENT '聚合时间类型：1、按小时聚合；2、按天聚合；3、按周聚合；4、按月聚合；5、按年聚合；',  `time\_str` STRING COMMENT '时间明细' ) comment '客户申诉app表' PARTITIONED BY(yearinfo STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' stored as orcfile TBLPROPERTIES ('orc.compress'='SNAPPY'); |

##### APP

如果用户需要具体的报表展示，可以针对不同的报表页面设计APP层结构，然后导出至OLAP系统的mysql中。此系统使用FineReport，需要通过宽表来进行灵活的展现。因此APP层不再进行细化。直接将DWS层导出至mysql即可。

### 全量流程



#### 数据采集

##### Customer\_clue线索表、customer\_relationship表

ODS复用意向客户指标，不需要重复采集数据。

##### customer\_appeal表

SQL：

|  |
| --- |
| select `id`,  `customer\_relationship\_first\_id`,  `employee\_id`,  `employee\_name`,  `employee\_department\_id`,  `employee\_tdepart\_id`,  `appeal\_status`,  `audit\_id`,  `audit\_name`,  `audit\_department\_id`,  `audit\_department\_name`,  `audit\_date\_time`,  `create\_date\_time`,  `update\_date\_time`,  `deleted`,  `tenant`,  DATE\_SUB(curdate(),INTERVAL 1 DAY) as start\_time  from customer\_appeal |

Sqoop：

|  |
| --- |
| sqoop import \  --connect jdbc:mysql://192.168.52.150:3306/scrm \  --username root \  --password 123456 \  --query 'select `id`,`customer\_relationship\_first\_id`,`employee\_id`,`employee\_name`,`employee\_department\_id`,`employee\_tdepart\_id`,`appeal\_status`,`audit\_id`,`audit\_name`,`audit\_department\_id`,`audit\_department\_name`,`audit\_date\_time`,`create\_date\_time`,`update\_date\_time`,`deleted`,`tenant`,DATE\_SUB(curdate(),INTERVAL 1 DAY) as start\_time from customer\_appeal where $CONDITIONS' \  --hcatalog-database itcast\_ods \  --hcatalog-table customer\_appeal \  -m 100 \  --split-by id |

#### 数据清洗转换

##### 分析

**一、清洗：**

过滤已删除数据(deleted)、线索状态clue\_state为空的数据；

**二、转换：**

关联意向表的customer\_relationship\_id字段如果为空则转换为-1。

customer\_clue线索表、customer\_relationship意向表可以获取并转换得到新老客户、线上线下。

customer\_clue线索表获取clue\_state信息，将clue\_state状态转换为新老客户：如果clue\_state状态为VALID\_NEW\_CLUES，则为新客户，为VALID\_PUBLIC\_NEW\_CLUE，则为老客户，否则为无效数据。

此处回顾SMB Join的用法，customer\_clue线索表的customer\_relationship\_id字段与customer\_relationship表的id字段进行关联。customer\_relationship意向客户主表，将origin\_type来源渠道字段转换为线上/线下：如果origin\_type是NETSERVICE和PRESIGNUP类型，即为1线上，否则为0线下。

在意向客户主题已经建立过这两个ODS表，类似的关联查询可以通过分桶提高查询效率，我们在这两个字段上分别建立分桶，而且要保证他们的排序和分桶的数量是一致的。

类似的关联查询可以通过分桶提高查询效率，我们在这两个字段上分别建立分桶，而且要保证他们的排序和分桶的数量是一致的。在意向客户主题已经建立过这两个ODS表。

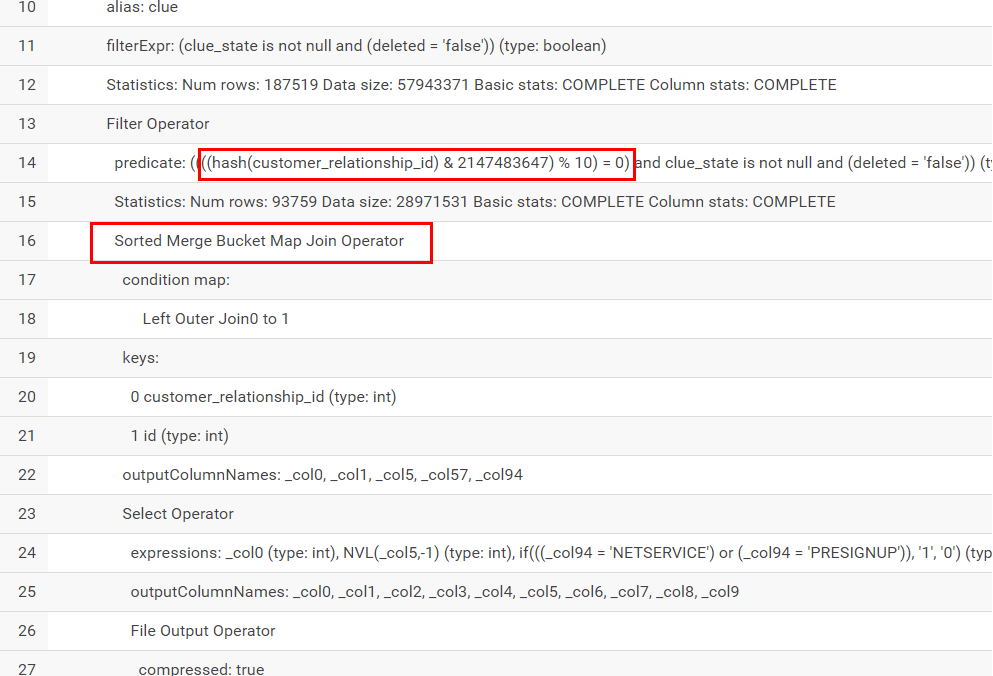
##### 代码

|  |
| --- |
| *--分区* SET hive.exec.dynamic.partition=true; SET hive.exec.dynamic.partition.mode=nonstrict; set hive.exec.max.dynamic.partitions.pernode=10000; set hive.exec.max.dynamic.partitions=100000; set hive.exec.max.created.files=150000; *--hive压缩* set hive.exec.compress.intermediate=true; set hive.exec.compress.output=true; *--写入时压缩生效* set hive.exec.orc.compression.strategy=COMPRESSION; *--分桶* set hive.enforce.bucketing=true; set hive.enforce.sorting=true; set hive.optimize.bucketmapjoin = true; set hive.auto.convert.sortmerge.join=true; set hive.auto.convert.sortmerge.join.noconditionaltask=true; *--并行执行* set hive.exec.parallel=true; set hive.exec.parallel.thread.number=8;  INSERT INTO itcast\_dwd.itcast\_clue\_dwd PARTITION (yearinfo, monthinfo, dayinfo) SELECT  clue.id,  ***nvl*(**clue.customer\_relationship\_id, -1**)** customer\_relationship\_id,  *if*(rs.origin\_type='NETSERVICE' or rs.origin\_type='PRESIGNUP', '1', '0') as origin\_type\_stat,  ***if*(**clue.clue\_state='VALID\_NEW\_CLUES', 1, ***if*(**clue.clue\_state='VALID\_PUBLIC\_NEW\_CLUE', 2, 0**))** as for\_new\_user,  clue.deleted,  clue.create\_date\_time,  *substr*(clue.create\_date\_time, 12, 2) as hourinfo,  *substr*(clue.create\_date\_time, 1, 4) as yearinfo,  *substr*(clue.create\_date\_time, 6, 2) as monthinfo,  *substr*(clue.create\_date\_time, 9, 2) as dayinfo FROM itcast\_ods.customer\_clue TABLESAMPLE(BUCKET 1 OUT OF 10 ON customer\_relationship\_id) clue LEFT JOIN itcast\_ods.customer\_relationship rs on clue.customer\_relationship\_id = rs.id where **clue.clue\_state is not null AND clue.deleted = 'false'**; |

##### 测试

customer\_relationship可以采用分桶采样的方式进行测试，以提升执行效率。注意tablesample关键字所在的位置。

通过执行计划，可以看到分桶后的Join查询，使用了SMB Join进行优化。



|  |
| --- |
| INSERT INTO itcast\_dwd.itcast\_clue\_dwd PARTITION (yearinfo, monthinfo, dayinfo) SELECT   ……  FROM itcast\_ods.customer\_clue TABLESAMPLE(BUCKET 1 OUT OF 10 ON customer\_relationship\_id) clue LEFT JOIN itcast\_ods.customer\_relationship rs on clue.customer\_relationship\_id = rs.id where clue.clue\_state is not null AND clue.deleted = 'false'; |

#### 统计分析

##### 分析

在DWD层通过关联转换获取到客户线索是否有效，以及线索的来源是线上还是线下。

DWM层会在DWD层的基础之上，判断线索是否被客服投诉；

因为不涉及去重问题，此处简单的按照最细粒度维度打包进行统计，便于上层的数据集市按需取数。其中时间维度使用最小粒度的小时维度。

DWS根据需要的维度，分别从DWM获取数据后进行二次聚合，因为DWM已经打包统计过一次，数据较少，所以DWS的统计效率会比较高。

##### 代码

###### DWM

实现

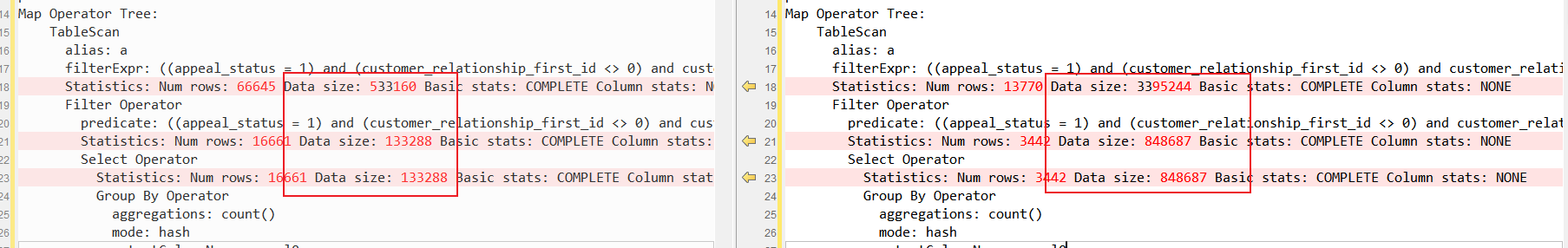
|  |
| --- |
| *--分区* SET hive.exec.dynamic.partition=true; SET hive.exec.dynamic.partition.mode=nonstrict; set hive.exec.max.dynamic.partitions.pernode=10000; set hive.exec.max.dynamic.partitions=100000; set hive.exec.max.created.files=150000; *--hive压缩；* set hive.exec.compress.intermediate=true; set hive.exec.compress.output=true; *--写入时压缩生效* set hive.exec.orc.compression.strategy=COMPRESSION; *--分桶* set hive.enforce.bucketing=true; set hive.enforce.sorting=true; set hive.optimize.bucketmapjoin = true; set hive.auto.convert.sortmerge.join=true; set hive.auto.convert.sortmerge.join.noconditionaltask=true; *--并行执行* set hive.exec.parallel=true; set hive.exec.parallel.thread.number=16;  INSERT into itcast\_dwm.itcast\_clue\_dwm partition (yearinfo) SELECT  *count*(dwd.id) as clue\_nums,  dwd.origin\_type\_stat,  dwd.for\_new\_user,  dwd.hourinfo,  dwd.dayinfo,  dwd.monthinfo,  dwd.yearinfo from itcast\_dwd.itcast\_clue\_dwd dwd WHERE  dwd.customer\_relationship\_id NOT IN (  SELECT a.customer\_relationship\_first\_id from itcast\_ods.customer\_appeal a  WHERE a.appeal\_status = 1 and a.customer\_relationship\_first\_id != 0 ) GROUP BY dwd.yearinfo, dwd.monthinfo, dwd.dayinfo, dwd.hourinfo, dwd.origin\_type\_stat, dwd.for\_new\_user; |

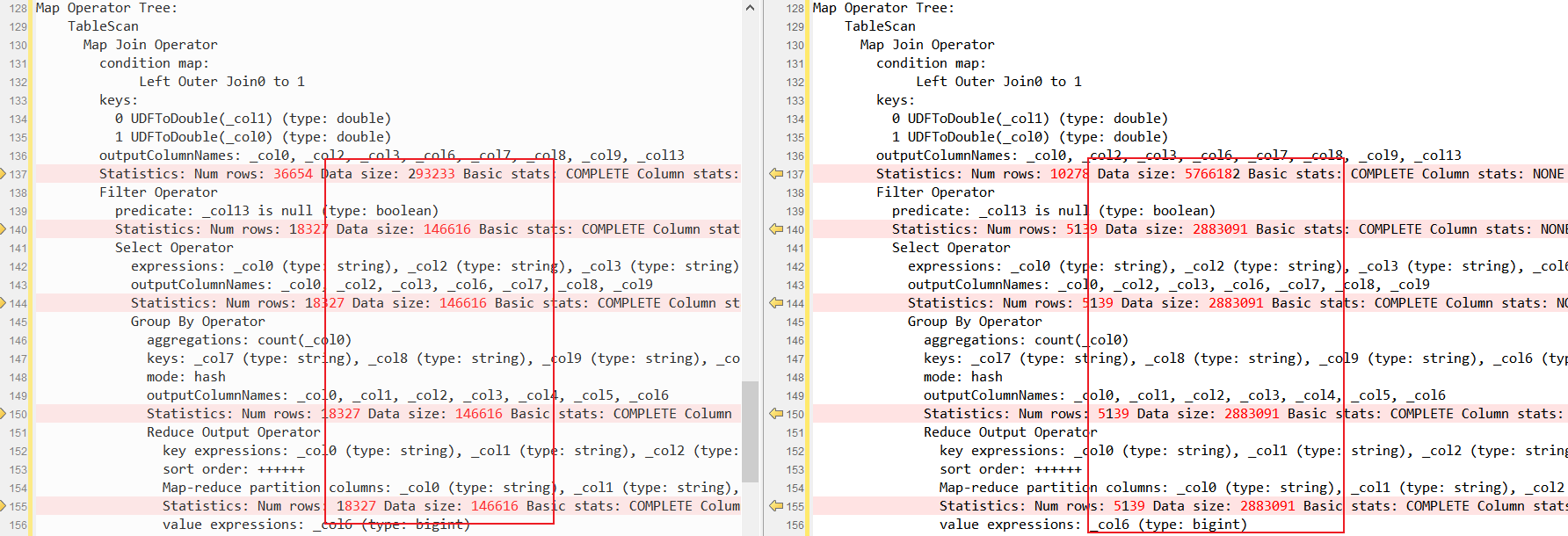
验证索引性能

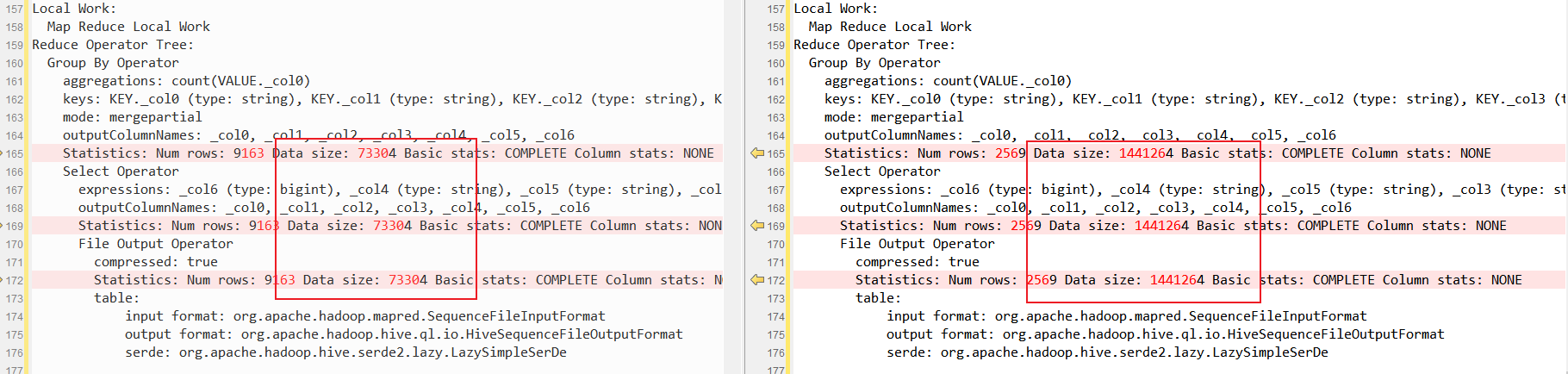
创建无索引表并导入数据：

|  |
| --- |
| CREATE TABLE IF NOT EXISTS itcast\_dwd.itcast\_clue\_dwd\_test (  `id` STRING COMMENT '线索id',  `customer\_relationship\_id` STRING COMMENT '客户关系id',  `origin\_type\_stat` STRING COMMENT '数据来源:0.线下；1.线上',  `for\_new\_user` STRING COMMENT '0:未知；1：新客户线索；2：旧客户线索',  `deleted` STRING COMMENT '是否删除',  `create\_date\_time` BIGINT COMMENT '创建时间',  `hourinfo` STRING COMMENT '小时信息' ) comment '客户申诉dwd表' PARTITIONED BY(yearinfo STRING,monthinfo STRING,dayinfo STRING) clustered by(customer\_relationship\_id) sorted by(customer\_relationship\_id) into 10 buckets ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' stored as orcfile TBLPROPERTIES ('orc.compress'='SNAPPY');   CREATE EXTERNAL TABLE IF NOT EXISTS itcast\_ods.`customer\_appeal\_test` (  `id` int COMMENT 'customer\_appeal\_id',  `customer\_relationship\_first\_id` int COMMENT '第一条客户关系id',  `employee\_id` int COMMENT '申诉人',  `employee\_name` STRING COMMENT '申诉人姓名',  `employee\_department\_id` int COMMENT '申诉人部门',  `employee\_tdepart\_id` int COMMENT '申诉人所属部门',  `appeal\_status` int COMMENT '申诉状态，0:待稽核 1:无效 2：有效',  `audit\_id` int COMMENT '稽核人id',  `audit\_name` STRING COMMENT '稽核人姓名',  `audit\_department\_id` int COMMENT '稽核人所在部门',  `audit\_department\_name` STRING COMMENT '稽核人部门名称',  `audit\_date\_time` STRING COMMENT '稽核时间',  `create\_date\_time` STRING COMMENT '创建时间（申诉时间）',  `update\_date\_time` STRING COMMENT '更新时间',  `deleted` STRING COMMENT '删除标志位',  `tenant` int COMMENT '租户id')  comment '客户申诉表' PARTITIONED BY(start\_time STRING) ROW FORMAT DELIMITED  FIELDS TERMINATED BY '\t' stored as orc TBLPROPERTIES ('orc.compress'='SNAPPY');  INSERT INTO itcast\_ods.customer\_appeal\_test PARTITION (start\_time) SELECT *\** from itcast\_ods.customer\_appeal;   INSERT INTO itcast\_dwd.itcast\_clue\_dwd\_test PARTITION (yearinfo, monthinfo, dayinfo) SELECT   clue.id,  clue.customer\_relationship\_id,  *if*(rs.origin\_type='NETSERVICE' or rs.origin\_type='PRESIGNUP', '1', '0') as origin\_type\_stat,   *if*(clue.clue\_state='VALID\_NEW\_CLUES', 1, *if*(clue.clue\_state='VALID\_PUBLIC\_NEW\_CLUE', 2, 0)) as for\_new\_user,   clue.deleted,  clue.create\_date\_time,  *substr*(clue.create\_date\_time, 12, 2) as hourinfo,   *substr*(clue.create\_date\_time, 1, 4) as yearinfo,   *substr*(clue.create\_date\_time, 6, 2) as monthinfo,   *substr*(clue.create\_date\_time, 9, 2) as dayinfo  FROM itcast\_ods.customer\_clue TABLESAMPLE(BUCKET 1 OUT OF 10 ON customer\_relationship\_id) clue LEFT JOIN itcast\_ods.customer\_relationship rs on clue.customer\_relationship\_id = rs.id where clue.clue\_state is not null AND clue.deleted = 'false';   explain SELECT  *count*(dwd.id) as clue\_nums,  dwd.origin\_type\_stat,  dwd.for\_new\_user,  dwd.hourinfo,  dwd.dayinfo,  dwd.monthinfo,  dwd.yearinfo from itcast\_dwd.itcast\_clue\_dwd\_test dwd WHERE  dwd.customer\_relationship\_id NOT IN (  SELECT a.customer\_relationship\_first\_id from itcast\_ods.customer\_appeal\_test a  WHERE a.appeal\_status = 1 and a.customer\_relationship\_first\_id != 0 ) GROUP BY dwd.yearinfo, dwd.monthinfo, dwd.dayinfo, dwd.hourinfo, dwd.origin\_type\_stat, dwd.for\_new\_user; |

通过执行计划，可以明显看到，加上索引以后，查询所读取的数据大小缩小了10倍以上，数据量越大提升越大：







###### DWS

DWS根据需要的维度，分别从DWM获取数据后进行二次聚合，因为DWM已经打包统计过一次，数据较少，所以DWS的统计效率会比较高。

小时数据和DWM层的数据是一致的，可以直接拿来使用。而年月日数据，则需要group by以后执行sum求和操作。

小时：

|  |
| --- |
| *--分区* SET hive.exec.dynamic.partition=true; SET hive.exec.dynamic.partition.mode=nonstrict; set hive.exec.max.dynamic.partitions.pernode=10000; set hive.exec.max.dynamic.partitions=100000; set hive.exec.max.created.files=150000; *--分桶* set hive.enforce.bucketing=true; set hive.enforce.sorting=true; set hive.optimize.bucketmapjoin = true; set hive.auto.convert.sortmerge.join=true; set hive.auto.convert.sortmerge.join.noconditionaltask=true; *--并行执行* set hive.exec.parallel=true; set hive.exec.parallel.thread.number=16;  *--小时* INSERT INTO itcast\_clue\_dws PARTITION(yearinfo) SELECT   clue\_nums,  origin\_type\_stat,   for\_new\_user,   hourinfo,   dayinfo,   monthinfo,   '1' as time\_type,   *concat*(dwm.yearinfo,'-',dwm.monthinfo,'-',dwm.dayinfo,' ', dwm.hourinfo) as time\_str,   yearinfo from itcast\_dwm.itcast\_clue\_dwm dwm; |

年月日数据：

|  |
| --- |
| *--天* INSERT INTO itcast\_clue\_dws PARTITION(yearinfo) SELECT   *sum*(clue\_nums) as clue\_nums,  origin\_type\_stat,   for\_new\_user,   '-1' as hourinfo,   dayinfo,   monthinfo,   '2' as time\_type,   *concat*(dwm.yearinfo,'-',dwm.monthinfo,'-',dwm.dayinfo) as time\_str,   yearinfo from itcast\_dwm.itcast\_clue\_dwm dwm **GROUP BY** dwm.yearinfo, dwm.monthinfo, dwm.dayinfo, dwm.origin\_type\_stat, dwm.for\_new\_user;   *--月* INSERT INTO itcast\_clue\_dws PARTITION(yearinfo) SELECT   *sum*(clue\_nums) as clue\_nums,  origin\_type\_stat,   for\_new\_user,   '-1' as hourinfo,   '-1' asdayinfo,   monthinfo,   '4' as time\_type,   *concat*(dwm.yearinfo,'-',dwm.monthinfo) as time\_str,   yearinfo from itcast\_dwm.itcast\_clue\_dwm dwm GROUP BY dwm.yearinfo, dwm.monthinfo, dwm.origin\_type\_stat, dwm.for\_new\_user;    *--年* INSERT INTO itcast\_clue\_dws PARTITION(yearinfo) SELECT   *sum*(clue\_nums) as clue\_nums,  origin\_type\_stat,   for\_new\_user,   '-1' as hourinfo,   '-1' as dayinfo,   '-1' as monthinfo,   '5' as time\_type,   dwm.yearinfo as time\_str,   yearinfo from itcast\_dwm.itcast\_clue\_dwm dwm GROUP BY dwm.yearinfo, dwm.origin\_type\_stat, dwm.for\_new\_user; |

#### 导出数据

##### 创建Mysql表

|  |
| --- |
| CREATE TABLE `itcast\_clue` (  `clue\_nums` INT(11) COMMENT '有效线索量',  `origin\_type\_stat` varchar(32) COMMENT '数据来源:0.线下；1.线上',  `for\_new\_user` varchar(32) COMMENT '0:未知；1：新客户线索；2：旧客户线索',  `hourinfo` varchar(32) COMMENT '小时信息',  `dayinfo` varchar(32) COMMENT '日信息',  `monthinfo` varchar(32) COMMENT '月信息',  `time\_type` varchar(32) COMMENT '聚合时间类型：1、按小时聚合；2、按天聚合；3、按周聚合；4、按月聚合；5、按年聚合；',  `time\_str` varchar(32) COMMENT '时间明细',  `yearinfo` varchar(32) COMMENT '年信息' ); |

##### Sqoop导出脚本

|  |
| --- |
| sqoop export \  --connect "jdbc:mysql://192.168.52.150:3306/scrm\_bi?useUnicode=true&characterEncoding=utf-8" \  --username root \  --password 123456 \  --table itcast\_clue \  --hcatalog-database itcast\_dws \  --hcatalog-table itcast\_clue\_dws \  -m 100 |

### 增量流程

#### 数据采集

##### customer\_relationship表

ODS复用意向客户指标，不需要重复采集数据。

##### customer\_appeal表

此表数据较少，因此可以直接全部覆盖同步，同全量过程。

#### 数据清洗转换

##### 分析

customer\_clue表是一个拉链表，会保存数据的历史状态。因为业务方将更新周期限制在30天内，所以只需查询更新30天内的数据即可。

因此在统计时，我们只需要将上个月1日至今的数据进行统计。

##### 代码

###### SQL：

|  |
| --- |
| *--分区* SET hive.exec.dynamic.partition=true; SET hive.exec.dynamic.partition.mode=nonstrict; set hive.exec.max.dynamic.partitions.pernode=10000; set hive.exec.max.dynamic.partitions=100000; set hive.exec.max.created.files=150000; *--hive压缩* set hive.exec.compress.intermediate=true; set hive.exec.compress.output=true; *--写入时压缩生效* set hive.exec.orc.compression.strategy=COMPRESSION; *--分桶* set hive.enforce.bucketing=true; set hive.enforce.sorting=true; set hive.optimize.bucketmapjoin = true; set hive.auto.convert.sortmerge.join=true; set hive.auto.convert.sortmerge.join.noconditionaltask=true; *--并行执行* set hive.exec.parallel=true; set hive.exec.parallel.thread.number=8;  INSERT INTO itcast\_dwd.itcast\_clue\_dwd PARTITION (yearinfo, monthinfo, dayinfo) SELECT  clue.id,  clue.customer\_relationship\_id,  *if*(rs.origin\_type='NETSERVICE' or rs.origin\_type='PRESIGNUP', '1', '0') as origin\_type\_stat,  *if*(clue.clue\_state='VALID\_NEW\_CLUES', 1, *if*(clue.clue\_state='VALID\_PUBLIC\_NEW\_CLUE', 2, 0)) as for\_new\_user,  clue.deleted,  clue.create\_date\_time,  *substr*(clue.create\_date\_time, 12, 2) as hourinfo,  *substr*(clue.create\_date\_time, 1, 4) as yearinfo,  *substr*(clue.create\_date\_time, 6, 2) as monthinfo,  *substr*(clue.create\_date\_time, 9, 2) as dayinfo FROM itcast\_ods.customer\_clue TABLESAMPLE(BUCKET 1 OUT OF 10 ON customer\_relationship\_id) clue LEFT JOIN itcast\_ods.customer\_relationship rs on clue.customer\_relationship\_id = rs.id where clue.clue\_state is not null AND clue.deleted = 'false' AND ***start\_time* = '${TD\_DATE}'**;*--2019-11-01* |

###### Shell脚本：

|  |
| --- |
| #! /bin/bash  #采集日期  if [[ $1 == "" ]];then  TD\_DATE=`date -d '1 days ago' "+%Y-%m-%d"`  else  TD\_DATE=$1  fi  ${HIVE\_HOME} -S -e "  --分区  SET hive.exec.dynamic.partition=true;  SET hive.exec.dynamic.partition.mode=nonstrict;  set hive.exec.max.dynamic.partitions.pernode=10000;  set hive.exec.max.dynamic.partitions=100000;  set hive.exec.max.created.files=150000;  --hive压缩  set hive.exec.compress.intermediate=true;  set hive.exec.compress.output=true;  --写入时压缩生效  set hive.exec.orc.compression.strategy=COMPRESSION;  --分桶  set hive.enforce.bucketing=true;  set hive.enforce.sorting=true;  set hive.optimize.bucketmapjoin = true;  set hive.auto.convert.sortmerge.join=true;  set hive.auto.convert.sortmerge.join.noconditionaltask=true;  --并行执行  set hive.exec.parallel=true;  set hive.exec.parallel.thread.number=8;  INSERT INTO itcast\_dwd.itcast\_clue\_dwd PARTITION (yearinfo, monthinfo, dayinfo)  SELECT  clue.id,  clue.customer\_relationship\_id,  if(rs.origin\_type='NETSERVICE' or rs.origin\_type='PRESIGNUP', '1', '0') as origin\_type\_stat,  if(clue.clue\_state='VALID\_NEW\_CLUES', 1, if(clue.clue\_state='VALID\_PUBLIC\_NEW\_CLUE', 2, 0)) as for\_new\_user,  clue.deleted,  clue.create\_date\_time,  substr(clue.create\_date\_time, 12, 2) as hourinfo,  substr(clue.create\_date\_time, 1, 4) as yearinfo,  substr(clue.create\_date\_time, 6, 2) as monthinfo,  substr(clue.create\_date\_time, 9, 2) as dayinfo  FROM itcast\_ods.customer\_clue TABLESAMPLE(BUCKET 1 OUT OF 10 ON customer\_relationship\_id) clue  LEFT JOIN itcast\_ods.customer\_relationship rs on clue.customer\_relationship\_id = rs.id  where clue.clue\_state is not null AND clue.deleted = 'false' substr(clue.create\_date\_time, 1, 10) >= '${TD\_DATE}';--2019-11-01  " |

#### 统计分析

增量统计时，只需要统计上个月1日至今的数据。

##### DWM

###### SQL

|  |
| --- |
| *--分区* SET hive.exec.dynamic.partition=true; SET hive.exec.dynamic.partition.mode=nonstrict; set hive.exec.max.dynamic.partitions.pernode=10000; set hive.exec.max.dynamic.partitions=100000; set hive.exec.max.created.files=150000; *--hive压缩* set hive.exec.compress.intermediate=true; set hive.exec.compress.output=true; *--写入表时压缩生效* set hive.exec.orc.compression.strategy=COMPRESSION; *--分桶* set hive.enforce.bucketing=true; set hive.enforce.sorting=true; set hive.optimize.bucketmapjoin = true; set hive.auto.convert.sortmerge.join=true; set hive.auto.convert.sortmerge.join.noconditionaltask=true; *--并行执行* set hive.exec.parallel=true; set hive.exec.parallel.thread.number=16;  INSERT into itcast\_dwm.itcast\_clue\_dwm partition (yearinfo) SELECT  *count*(dwd.id) as clue\_nums,  dwd.origin\_type\_stat,  dwd.for\_new\_user,  dwd.hourinfo,  dwd.dayinfo,  dwd.monthinfo,  dwd.yearinfo from itcast\_dwd.itcast\_clue\_dwd dwd WHERE  dwd.customer\_relationship\_id NOT IN (  SELECT a.customer\_relationship\_first\_id from itcast\_ods.customer\_appeal a  WHERE a.appeal\_status = 1 and a.customer\_relationship\_first\_id != 0 ) AND *concat\_ws*('-',dwd.yearinfo,dwd.monthinfo,dwd.dayinfo)>= '2019-11-01' GROUP BY dwd.yearinfo, dwd.monthinfo, dwd.dayinfo, dwd.hourinfo, dwd.origin\_type\_stat, dwd.for\_new\_user; |

###### Shell脚本

|  |
| --- |
| #! /bin/bash  #上个月1日  Last\_Month\_DATE=$(date -d "$(date +%Y%m)01 last month" +%Y-%m-01)  ${HIVE\_HOME} -S -e "  --分区  SET hive.exec.dynamic.partition=true;  SET hive.exec.dynamic.partition.mode=nonstrict;  set hive.exec.max.dynamic.partitions.pernode=10000;  set hive.exec.max.dynamic.partitions=100000;  set hive.exec.max.created.files=150000;  --hive压缩  set hive.exec.compress.intermediate=true;  set hive.exec.compress.output=true;  --写入表时压缩生效  set hive.exec.orc.compression.strategy=COMPRESSION;  --分桶  set hive.enforce.bucketing=true;  set hive.enforce.sorting=true;  set hive.optimize.bucketmapjoin = true;  set hive.auto.convert.sortmerge.join=true;  set hive.auto.convert.sortmerge.join.noconditionaltask=true;  --并行执行  set hive.exec.parallel=true;  set hive.exec.parallel.thread.number=16;  INSERT into itcast\_dwm.itcast\_clue\_dwm partition (yearinfo)  SELECT  count(dwd.id) as clue\_nums,  dwd.origin\_type\_stat,  dwd.for\_new\_user,  dwd.hourinfo,  dwd.dayinfo,  dwd.monthinfo,  dwd.yearinfo  from itcast\_dwd.itcast\_clue\_dwd dwd  WHERE  dwd.customer\_relationship\_id NOT IN  (  SELECT a.customer\_relationship\_first\_id from itcast\_ods.customer\_appeal a  WHERE a.appeal\_status = 1 and a.customer\_relationship\_first\_id != 0  )  AND concat\_ws('-',dwd.yearinfo,dwd.monthinfo,dwd.dayinfo)>= '$Last\_Month\_DATE'  GROUP BY dwd.yearinfo, dwd.monthinfo, dwd.dayinfo, dwd.hourinfo, dwd.origin\_type\_stat, dwd.for\_new\_user;  " |

##### DWS

统计上个月1日至今的数据。

###### SQL

|  |
| --- |
| *--分区* SET hive.exec.dynamic.partition=true; SET hive.exec.dynamic.partition.mode=nonstrict; set hive.exec.max.dynamic.partitions.pernode=10000; set hive.exec.max.dynamic.partitions=100000; set hive.exec.max.created.files=150000; *--分桶* set hive.enforce.bucketing=true; set hive.enforce.sorting=true; set hive.optimize.bucketmapjoin = true; set hive.auto.convert.sortmerge.join=true; set hive.auto.convert.sortmerge.join.noconditionaltask=true; *--并行执行* set hive.exec.parallel=true; set hive.exec.parallel.thread.number=16; *--小时* INSERT INTO itcast\_dws.itcast\_clue\_dws PARTITION(yearinfo) SELECT  clue\_nums,  origin\_type\_stat,  for\_new\_user,  hourinfo,  dayinfo,  monthinfo,  '1' as time\_type,  *concat*(dwm.yearinfo,'-',dwm.monthinfo,'-',dwm.dayinfo,' ', dwm.hourinfo) as time\_str,  yearinfo from itcast\_dwm.itcast\_clue\_dwm dwm where ***CONCAT\_WS*('-', dwm.yearinfo, dwm.monthinfo, dwm.dayinfo)>='${Last\_Month\_DATE}'**;  *--天* INSERT INTO itcast\_dws.itcast\_clue\_dws PARTITION(yearinfo) SELECT  *sum*(clue\_nums) as clue\_nums,  origin\_type\_stat,  for\_new\_user,  '-1' as hourinfo,  dayinfo,  monthinfo,  '2' as time\_type,  *concat*(dwm.yearinfo,'-',dwm.monthinfo,'-',dwm.dayinfo) as time\_str,  yearinfo from itcast\_dwm.itcast\_clue\_dwm dwm where ***CONCAT\_WS*('-', dwm.yearinfo, dwm.monthinfo, dwm.dayinfo)>='${Last\_Month\_DATE}'** GROUP BY dwm.yearinfo, dwm.monthinfo, dwm.dayinfo, dwm.origin\_type\_stat, dwm.for\_new\_user;   *--月* INSERT INTO itcast\_dws.itcast\_clue\_dws PARTITION(yearinfo) SELECT  *sum*(clue\_nums) as clue\_nums,  origin\_type\_stat,  for\_new\_user,  '-1' as hourinfo,  '-1' asdayinfo,  monthinfo,  '4' as time\_type,  *concat*(dwm.yearinfo,'-',dwm.monthinfo) as time\_str,  yearinfo from itcast\_dwm.itcast\_clue\_dwm dwm where ***CONCAT\_WS*('-', dwm.yearinfo, dwm.monthinfo)>='${V\_Month}'** GROUP BY dwm.yearinfo, dwm.monthinfo, dwm.origin\_type\_stat, dwm.for\_new\_user;    *--年* INSERT INTO itcast\_dws.itcast\_clue\_dws PARTITION(yearinfo) SELECT  *sum*(clue\_nums) as clue\_nums,  origin\_type\_stat,  for\_new\_user,  '-1' as hourinfo,  '-1' as dayinfo,  '-1' as monthinfo,  '5' as time\_type,  dwm.yearinfo as time\_str,  yearinfo from itcast\_dwm.itcast\_clue\_dwm dwm where **dwm.yearinfo>='${V\_Year}'** GROUP BY dwm.yearinfo, dwm.origin\_type\_stat, dwm.for\_new\_user; |

###### Shell脚本

|  |
| --- |
| #! /bin/bash  #上个月1日  Last\_Month\_DATE=$(date -d "$(date +%Y%m)01 last month" +%Y-%m-01)  #根据TD\_DATE计算年季度月日  V\_PARYEAR=`date --date="$Last\_Month\_DATE" +%Y`  V\_PARMONTH=`date --date="$Last\_Month\_DATE" +%m`  V\_PARDAY=`date --date="$Last\_Month\_DATE" +%d`  V\_month\_for\_quarter=`date --date="$Last\_Month\_DATE" +%-m`  V\_PARQUARTER=$(((${V\_month\_for\_quarter}-1)/3+1))  #计算所需要的日期字符串  V\_Month="${V\_PARYEAR}"\_"${V\_PARMONTH}"  V\_QUARTER="${V\_PARYEAR}"\_Q"${V\_PARQUARTER}"  V\_Year="${V\_PARYEAR}"  ${HIVE\_HOME} -S -e "  --分区  SET hive.exec.dynamic.partition=true;  SET hive.exec.dynamic.partition.mode=nonstrict;  set hive.exec.max.dynamic.partitions.pernode=10000;  set hive.exec.max.dynamic.partitions=100000;  set hive.exec.max.created.files=150000;  --分桶  set hive.enforce.bucketing=true;  set hive.enforce.sorting=true;  set hive.optimize.bucketmapjoin = true;  set hive.auto.convert.sortmerge.join=true;  set hive.auto.convert.sortmerge.join.noconditionaltask=true;  --并行执行  set hive.exec.parallel=true;  set hive.exec.parallel.thread.number=16;  --小时  INSERT INTO itcast\_dws.itcast\_clue\_dws PARTITION(yearinfo)  SELECT  clue\_nums,  origin\_type\_stat,  for\_new\_user,  hourinfo,  dayinfo,  monthinfo,  '1' as time\_type,  concat(dwm.yearinfo,'-',dwm.monthinfo,'-',dwm.dayinfo,' ', dwm.hourinfo) as time\_str,  yearinfo  from itcast\_dwm.itcast\_clue\_dwm dwm  where CONCAT\_WS('-', dwm.yearinfo, dwm.monthinfo, dwm.dayinfo)>='${Last\_Month\_DATE}';  --天  INSERT INTO itcast\_dws.itcast\_clue\_dws PARTITION(yearinfo)  SELECT  sum(clue\_nums) as clue\_nums,  origin\_type\_stat,  for\_new\_user,  '-1' as hourinfo,  dayinfo,  monthinfo,  '2' as time\_type,  concat(dwm.yearinfo,'-',dwm.monthinfo,'-',dwm.dayinfo) as time\_str,  yearinfo  from itcast\_dwm.itcast\_clue\_dwm dwm  where CONCAT\_WS('-', dwm.yearinfo, dwm.monthinfo, dwm.dayinfo)>='${Last\_Month\_DATE}'  GROUP BY dwm.yearinfo, dwm.monthinfo, dwm.dayinfo, dwm.origin\_type\_stat, dwm.for\_new\_user;  --月  INSERT INTO itcast\_dws.itcast\_clue\_dws PARTITION(yearinfo)  SELECT  sum(clue\_nums) as clue\_nums,  origin\_type\_stat,  for\_new\_user,  '-1' as hourinfo,  '-1' asdayinfo,  monthinfo,  '4' as time\_type,  concat(dwm.yearinfo,'-',dwm.monthinfo) as time\_str,  yearinfo  from itcast\_dwm.itcast\_clue\_dwm dwm  where CONCAT\_WS('-', dwm.yearinfo, dwm.monthinfo)>='${V\_Month}'  GROUP BY dwm.yearinfo, dwm.monthinfo, dwm.origin\_type\_stat, dwm.for\_new\_user;    --年  INSERT INTO itcast\_dws.itcast\_clue\_dws PARTITION(yearinfo)  SELECT  sum(clue\_nums) as clue\_nums,  origin\_type\_stat,  for\_new\_user,  '-1' as hourinfo,  '-1' as dayinfo,  '-1' as monthinfo,  '5' as time\_type,  dwm.yearinfo as time\_str,  yearinfo  from itcast\_dwm.itcast\_clue\_dwm dwm  where dwm.yearinfo>='${V\_Year}'  GROUP BY dwm.yearinfo, dwm.origin\_type\_stat, dwm.for\_new\_user;  " |

#### 导出数据

|  |
| --- |
| #! /bin/bash SQOOP\_HOME=/usr/bin/sqoop HOST=192.168.52.150 USERNAME="root"  PASSWORD="123456" PORT=3306 DBNAME="scrm\_bi" MYSQL=/usr/local/mysql\_5723/bin/mysql  #上个月1日  if [[ $1 == "" ]];then  Last\_Month\_DATE=$(date -d "$(date +%Y%m)01 last month" +%Y-%m-01) else  Last\_Month\_DATE=$1 fi ${MYSQL} -h${HOST} -P${PORT} -u${USERNAME} -p${PASSWORD} -D${DBNAME} -e "delete from itcast\_clue where yearinfo = '${Last\_Month\_DATE:0:4}'" ${SQOOP\_HOME} export \ *--connect "jdbc:mysql://${HOST}:${PORT}/${DBNAME}?useUnicode=true&characterEncoding=utf-8" \ --username ${USERNAME} \ --password ${PASSWORD} \ --table itcast\_clue \ --hcatalog-database itcast\_dws \ --hcatalog-table itcast\_clue\_dws \ --hcatalog-partition-keys yearinfo \ --hcatalog-partition-values ${TD\_DATE:0:4} \* -m 100 |