# 报名用户主题看板

## 学习目标

了解报名用户主题看板需求

能够使用Hive进行并行操作

掌握Hive常用的判断函数

掌握Hive小文件合并的用法

掌握矢量化查询的用法

掌握关联优化器的用法

掌握读取零拷贝的用法

能够编写报名用户指标的DWD清洗转换SQL

掌握表连接查询时数据倾斜的优化方法

掌握分组统计时数据倾斜的优化方法

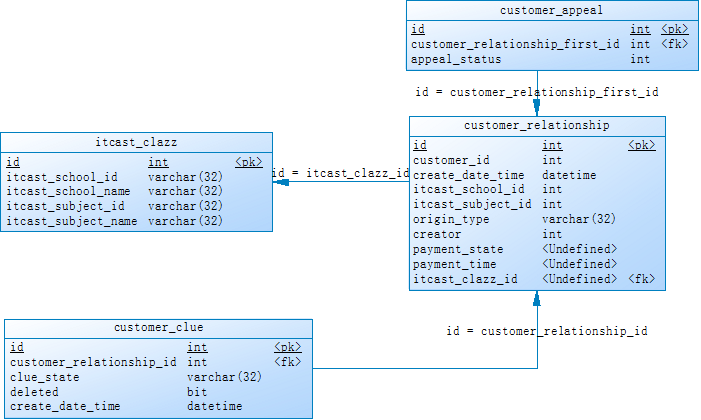
能够编写报名用户指标的DWM中间层SQL

能够编写报名用户指标的DWS业务层SQL

能够编写报名用户指标的APP应用层SQL

## 主题需求

此主题下指标需要能够下钻到小时数据。



### 校区报名柱状图

说明：统计期内，全部报名客户中，各校区报名人数分布。

展现：柱状图

条件：年、月，校区

维度：天区间，按查询条件来定

指标：报名人数

粒度：天/线上线下/校区

数据来源：客户管理系统的customer\_relationship、itcast\_clazz报名课程表

SQL：

1. **SELECT**
2. count( 1 ) '报名数量',
3. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) '时间',
4. **ic.itcast\_school\_name** '校区名称'
5. **FROM**
6. **customer\_relationship** cr
7. LEFT JOIN **itcast\_clazz** ic **ON** **cr.itcast\_clazz\_id = ic.id**
8. **WHERE**
9. **cr.payment\_state =** **'PAID'**
10. AND DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) >= '2019-08-01'
11. AND DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) <= '2019-12-01'
12. **GROUP** **BY**
13. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ),
14. **ic.itcast\_school\_id**
15. **ORDER** **BY**
16. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) **ASC**,
17. count( 1 ) **DESC**;

### 学科报名柱状图

说明：统计期内，全部报名客户中，各学科报名人数分布。

展现：柱状图

条件：年、月，学科

维度：天区间，按查询条件来定

指标：报名人数

粒度：天/线上线下/学科

数据来源：客户管理系统的customer\_relationship、itcast\_clazz报名课程表

SQL：

1. **SELECT**
2. count( 1 ) '报名数量',
3. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) '时间',
4. **ic.itcast\_subject\_name** '学科名称'
5. **FROM**
6. customer\_relationship cr
7. LEFT JOIN itcast\_clazz ic **ON** cr.itcast\_clazz\_id = ic.id
8. **WHERE**
9. cr.payment\_state = 'PAID'
10. AND DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) >= '2019-08-01'
11. AND DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) <= '2019-12-01'
12. **GROUP** **BY**
13. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ),
14. **ic.itcast\_subject\_id**
15. **ORDER** **BY**
16. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) **ASC**,
17. count( 1 ) **DESC**;

### 总报名量

说明：统计期内，已经缴费的报名客户总量。

展现：数值。

条件：年、月

维度：年、月

指标：报名客户总量

粒度：天

数据来源：客户管理系统的customer\_relationship表

SQL：

1. **SELECT**
2. count( 1 ) '报名数量',
3. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) '时间'
4. **FROM**
5. **customer\_relationship cr**
6. **WHERE**
7. cr.payment\_state = 'PAID'
8. AND DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) >= '2019-08-01'
9. AND DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) <= '2019-12-01'
10. **GROUP** **BY**
11. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' )
12. **ORDER** **BY**
13. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) **ASC**,
14. count( 1 ) **DESC**;

### 线上报名量

说明：总报名量中来源渠道为线上访客渠道的报名总量

展现：线状图。

条件：年、月

维度：天区间，按查询条件来定

指标：报名客户总量

粒度：天

数据来源：客户管理系统的customer\_relationship表

SQL：

1. **SELECT**
2. count( 1 ) '报名数量',
3. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) '时间'
4. **FROM**
5. customer\_relationship cr
6. **WHERE**
7. cr.payment\_state = 'PAID'
8. **AND cr.origin\_type IN ('NETSERVICE','PRESIGNUP')**
9. AND DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) >= '2019-08-01'
10. AND DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) <= '2019-12-01'
11. **GROUP** **BY**
12. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' )
13. **ORDER** **BY**
14. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) **ASC**,
15. count( 1 ) **DESC**;

### 意向用户报名转化率

说明：统计期内，新增的意向客户中报名的客户占比。**全部报名人数 / 全部新增的意向人数**

展现：线状图。双轴：全部报名人数、报名转化率。

条件：年、月

维度：天/线上线下

指标：报名转化率=全部报名人数/全部新增的意向人数

粒度：天

数据来源：客户管理系统的customer\_relationship表

SQL：

1. **SELECT**
2. CONCAT(
3. **signup\_num.num / total\_num.num \* 100**,
4. '%'
5. ) '报名转化率',
6. total\_num.date\_day '时间'
7. **FROM**
8. **(**
9. **SELECT**
10. count( 1 ) **AS** num,
11. DATE\_FORMAT( cr.create\_date\_time, '%Y-%m-%d' ) **AS** date\_day
12. **FROM**
13. customer\_relationship cr
14. **GROUP** **BY**
15. DATE\_FORMAT( cr.create\_date\_time, '%Y-%m-%d' )
16. **) total\_num**
17. **INNER** JOIN **(**
18. **SELECT**
19. count( 1 ) **AS** num,
20. DATE\_FORMAT( cr.create\_date\_time, '%Y-%m-%d' ) **AS** date\_day
21. **FROM**
22. customer\_relationship cr
23. **WHERE**
24. **cr.payment\_state = 'PAID'**
25. **GROUP** **BY**
26. DATE\_FORMAT( cr.create\_date\_time, '%Y-%m-%d' )
27. **)** **signup\_num** **ON** total\_num.date\_day = signup\_num.date\_day

### 有效线索报名转化率

说明：**线上报名量 / 线上有效线索量**，与上一个指标类似，此处的线索量需要排除已申诉数据。

展现：线状图。双轴：线上报名人数、线上报名转化率。

条件：年、月

维度：天/线上线下

指标：线上报名转化率=线上报名人数/线上有效线索量

粒度：天

数据来源：客户管理系统的customer\_relationship表、customer\_clue表、customer\_appeal表

SQL：

1. **SELECT**
2. CONCAT(
3. signup\_num.num / total\_num.num \* 100,
4. '%'
5. ) '线上报名转化率',
6. total\_num.date\_day '时间'
7. **FROM**
8. **(**
9. **SELECT**
10. count( 1 ) **AS** num,
11. DATE\_FORMAT( cr.create\_date\_time, '%Y-%m-%d' ) **AS** date\_day
12. **FROM**
13. customer\_clue cc
14. **LEFT** JOIN customer\_relationship cr **ON** cr.id = cc.customer\_relationship\_id
15. **WHERE**
16. cc.clue\_state IN ( 'VALID\_NEW\_CLUES', 'VALID\_PUBLIC\_NEW\_CLUE' )
17. AND !cc.deleted
18. AND cr.id NOT IN ( **SELECT** customer\_relationship\_first\_id **FROM** customer\_appeal **WHERE** appeal\_status = '1' )
19. **GROUP** **BY**
20. DATE\_FORMAT( cr.create\_date\_time, '%Y-%m-%d' )
21. **) total\_num**
22. **INNER** JOIN **(**
23. **SELECT**
24. count( 1 ) **AS** num,
25. DATE\_FORMAT( cr.create\_date\_time, '%Y-%m-%d' ) **AS** date\_day
26. **FROM**
27. customer\_relationship cr
28. **WHERE**
29. cr.payment\_state = 'PAID'
30. **GROUP** **BY**
31. DATE\_FORMAT( cr.create\_date\_time, '%Y-%m-%d' )
32. **) signup\_num** **ON** total\_num.date\_day = signup\_num.date\_day

### 日报名趋势图

说明：统计期内，每天报名人数的趋势图。

展现：线状图。

条件：年、月

维度：天/线上线下

指标：报名人数

粒度：天

数据来源：客户管理系统的customer\_relationship表

SQL：

1. **SELECT**
2. count( 1 ) '报名人数',
3. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' )
4. **FROM**
5. **customer\_relationship cr**
6. **WHERE**
7. cr.payment\_state = 'PAID'
8. AND DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) >= '2019-08-01'
9. AND DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' ) <= '2019-12-01'
10. **GROUP** **BY**
11. DATE\_FORMAT( cr.payment\_time, '%Y-%m-%d' )

### 校区学科的报名学员TOP

说明：统计期内，全部报名学员中，校区学科排行榜，topN。A校区b学科第一，B校区a学科第二等等。

展现：柱状图

条件：年、月，校区，学科，数据量N

维度：天/线上线下

指标：报名学员人数

粒度：各校区各学科的报名人数和

数据来源：客户管理系统的customer\_relationship表、itcast\_clazz表

SQL：

1. **SELECT**
2. count(1) '总数',
3. itc.itcast\_school\_id,
4. itc.itcast\_school\_name,
5. itc.itcast\_subject\_id,
6. itc.itcast\_subject\_name,
7. cr.payment\_state,
8. cr.payment\_time
9. **FROM**
10. customer\_relationship cr
11. LEFT JOIN itcast\_clazz itc **ON** cr.itcast\_clazz\_id = itc.id
12. **WHERE**
13. cr.payment\_state = 'PAID'
14. AND cr.payment\_time >= '2019-10-01'
15. AND cr.payment\_time <= '2020-10-31 23:59:59'
16. **GROUP** **BY**
17. **itc.itcast\_school\_id,**
18. **itc.itcast\_subject\_id**
19. **ORDER** **BY**
20. count(1) **DESC**;

### 来源渠道占比

说明：统计期内，全部报名学员中，不同来源渠道的报名学员占比情况。

展现：饼状图

条件：年、月

维度：天/线上线下/来源渠道

指标：比值

数据来源：客户管理系统的customer\_relationship表

SQL：

1. **SELECT**
2. count( 1 ) '总数',
3. cr.origin\_type,
4. cr.payment\_state,
5. cr.payment\_time
6. **FROM**
7. customer\_relationship cr
8. **WHERE**
9. cr.payment\_state = 'PAID'
10. AND cr.payment\_time >= '2019-10-01'
11. AND cr.payment\_time <= '2019-10-31 23:59:59'
12. **GROUP** **BY**
13. **cr.origin\_type**;

### 咨询中心报名贡献

说明：统计期内，全部报名学员中，各咨询中心的报名学员人数占比情况。

展现：饼状图

条件：年、月，咨询中心

维度：天/线上线下/咨询中心

指标：报名学员人数

粒度：天/报名学员人数

数据来源：客户管理系统的customer\_relationship表、employee表、scrm\_department表

SQL：

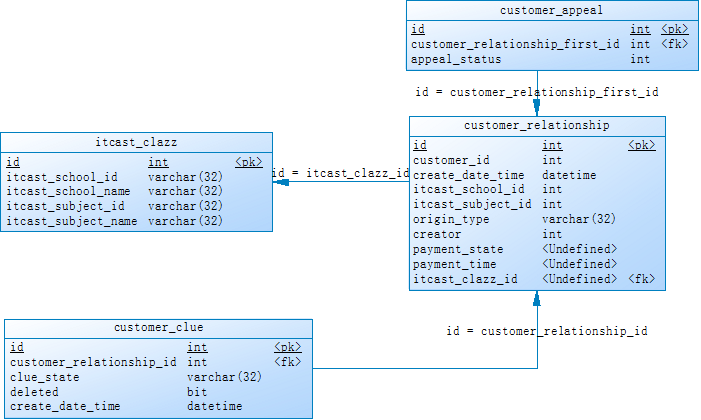
1. **SELECT**
2. count( 1 ),
3. e.tdepart\_id,
4. sd.`**name**`
5. **FROM**
6. customer\_relationship cr
7. LEFT JOIN employee e **ON** cr.creator = e.id
8. LEFT JOIN scrm\_department sd **ON** e.tdepart\_id = sd.id
9. **WHERE**
10. cr.payment\_state = 'PAID'
11. AND cr.payment\_time >= '2019-10-01'
12. AND cr.payment\_time <= '2019-10-31 23:59:59'
13. **GROUP** **BY**
14. **e.tdepart\_id**;

### 原始数据结构

customer\_relationship、employee、scrm\_department三张表，在之前的主题统计中已经实现，可以复用。

itcast\_clazz报名课程表可以获取客户报名的校区和学科。

测试数据：【Home\讲义\第7章 报名用户主题看板\mysql测试数据】



|  |
| --- |
| CREATE TABLE `itcast\_clazz` (  `id` int(11) NOT NULL COMMENT 'ems课程id(非自增)',  `create\_date\_time` datetime NOT NULL DEFAULT *CURRENT\_TIMESTAMP* COMMENT '创建时间',  `update\_date\_time` timestamp NOT NULL DEFAULT *CURRENT\_TIMESTAMP* ON UPDATE *CURRENT\_TIMESTAMP* COMMENT '最后更新时间',  `deleted` bit(1) NOT NULL DEFAULT b'0' COMMENT '是否被删除（禁用）',  `itcast\_school\_id` varchar(32) COLLATE utf8\_bin DEFAULT '' COMMENT 'ems校区ID',  `itcast\_school\_name` varchar(32) COLLATE utf8\_bin DEFAULT '' COMMENT 'ems校区名称',  `itcast\_subject\_id` varchar(32) COLLATE utf8\_bin DEFAULT '' COMMENT 'ems学科ID',  `itcast\_subject\_name` varchar(32) COLLATE utf8\_bin DEFAULT '' COMMENT 'ems学科名称',  `itcast\_brand` varchar(32) COLLATE utf8\_bin DEFAULT NULL COMMENT 'ems品牌',  `clazz\_type\_state` varchar(32) COLLATE utf8\_bin DEFAULT NULL COMMENT '班级类型状态',  `clazz\_type\_name` varchar(32) COLLATE utf8\_bin DEFAULT '' COMMENT '班级类型名称',  `teaching\_mode` varchar(32) COLLATE utf8\_bin DEFAULT NULL COMMENT '授课模式',  `start\_time` datetime DEFAULT NULL COMMENT '开班时间',  `end\_time` datetime DEFAULT NULL COMMENT '毕业时间',  `comment` varchar(255) COLLATE utf8\_bin DEFAULT '' COMMENT '备注',  `detail` varchar(32) COLLATE utf8\_bin DEFAULT NULL COMMENT '详情(比如：27期)',  `uncertain` bit(1) DEFAULT b'0' COMMENT '待定班(0:否,1:是)',  `tenant` int(11) NOT NULL DEFAULT '0',  PRIMARY KEY (`id`) ) ENGINE=InnoDB 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：

共有维度：年、月，以及线上线下的维度。

校区报名柱状图，是将不同校区的报名用户数据进行统计。指标是报名用户量，维度是校区+共有维度。

学科报名柱状图，是将不同学科的报名用户数据进行统计。指标是报名用户量，维度是学科+共有维度。

总报名量，是统计所有的报名用户数据。指标是报名用户量，共有维度。

线上报名量，是在总报名量的基础上，线上的那部分数据，因此此指标的数据和维度被总报名量指标包含。

意向用户报名转化率，等于全部报名人数/全部新增的意向人数。分母新增意向人数，我们在之前的主题中已经统计出结果，可以直接复用。分子全部报名人数，和第三个指标总报名量是一致的。所以此指标的数据和维度都已经被包含了。

有效线索报名转化率，等于线上报名量/线上有效线索量。分母线上有效线索量，在之前的线索主题中已经统计过，此处可以直接复用。分子线上报名量，和第三个指标总报名量是一致的，已被包含。因此此指标的数据已被其他指标所包含。

日报名趋势图，是天维度的总报名数据，在共有维度基础上，多了个天维度。

校区学科报名学员TOP，是统计各个校区各个学科的报名学员数量，取最大的几名数据，指标是报名用户量，维度是校区+学科+共有维度。

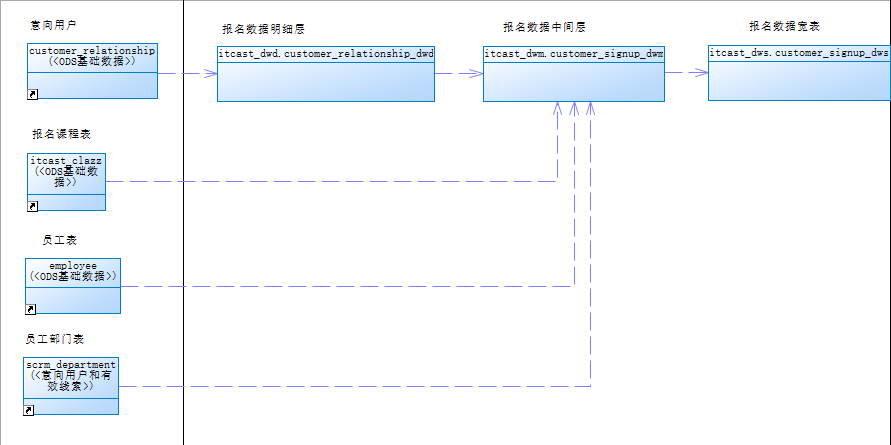
来源渠道报名学员占比，指标是报名用户量，维度是来源渠道+共有维度。

咨询中心报名贡献，指标是报名用户量，维度是咨询中心+共有维度。

由此我们可以推断出，指标是：**报名用户量**、意向客户量、有效线索量。其中意向客户量、有效线索量可以复用前面的看板数据。

### 分层设计

#### 报名用户量



1. 意向用户报名转化率、有效线索报名转化率，使用到的数据和维度都已经被其他指标所包含，因此过程无需再进行重复计算。
2. 需求要求数据能够下钻到小时，因此维度中需要增加小时维度。
3. 维度：年、月、日、小时，线上线下、校区、学科、总量、校区+学科、来源渠道、咨询中心。
4. 其中，校区、学科、校区+学科，三个维度是有重叠的。我们可以只保留校区+学科的维度；分开的校区、学科维度数据可以通过校区+学科的结果二次分组即可得到，处理简单且快速（数据量少），因此交由OLAP应用根据需要自行处理。维度：年、月、日、小时，线上线下、总量、校区+学科、来源渠道、咨询中心。
5. 我们将维度划分为三类：
   1. 时间维度（年、月、日、小时）；
   2. 产品属性维度（1：校区、学科组合分组、2：来源渠道分组、3：咨询中心分组、4:所有）；
   3. 数据来源（线上线下）。
6. ODS层原始数据包括：customer\_relationship（有报名信息）、itcast\_clazz（获取到报名后的校区和学科信息）、employee（内部员工信息）、scrm\_department（部门信息）。
7. **DWD**层清洗删除的、customer不为空的、且是已支付的数据；转换获取到线上线下、年月日字段。
8. 用户不存在多次报名的情况，因此报名数据无需去重。所以可以使用DWM中间层来进行过度，可被其他DWS层调用以提高计算速度；
9. **DWM**层在DWD层的基础上，关联校区、学科和咨询中心，获取到所需要的关联字段。
10. **DWS**在DWM层的基础上，按照产品属性维度进行统计，得到统计宽表。其中时间维度使用最小粒度的小时维度。
11. 为了方便OLAP应用使用，可以在**APP**层按照年月日的时间维度进行二次聚合。因为DWS已经按照产品属性维度统计过一次，数据较少，所以APP的统计效率会比较高。

#### 意向用户索量

复用意向用户主题看板。

#### 有效线索量

复用有效线索主题看板。

## 报名用户量实现

### 建模

#### 指标和维度

指标：报名用户量

维度：

* 时间维度：年、月、天、小时
* 数据来源：线上、线下
* 客户类型：总量、校区+学科、来源渠道、咨询中心

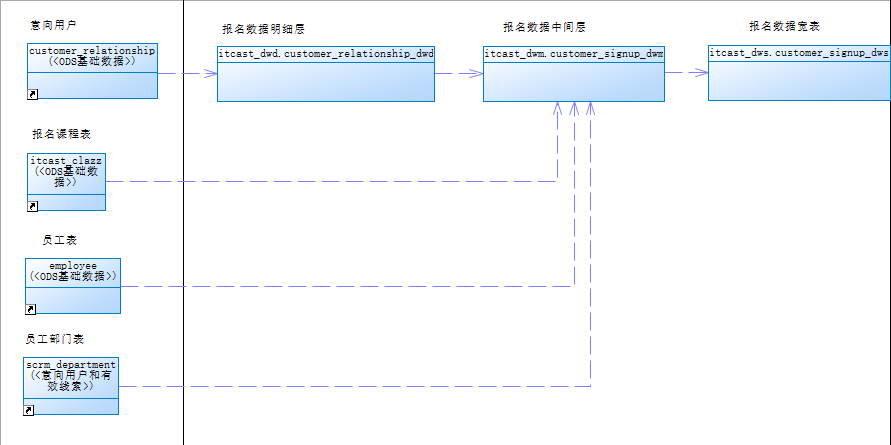
#### 事实表和维度表

对于报名用户指标来说，customer\_relationship表中包含有用户的报名信息，是事实表。

itcast\_clazz表是为了获取客户报名的校区和学科信息，是维度表。

employee、scrm\_department表是为了获取报名所在的咨询中心信息，是维度表。

#### 分层



##### ODS

ODS层的customer\_relationship表可以直接复用。

itcast\_clazz表是维表，放在维表层。

##### Dimension

###### employee、scrm\_department表

可复用意向客户主题ODS层数据。

###### itcast\_clazz报名课程表

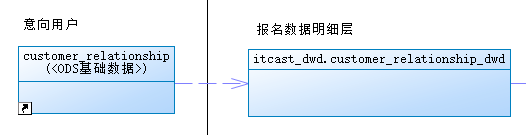
|  |
| --- |
| CREATE EXTERNAL TABLE IF NOT EXISTS itcast\_dimen.itcast\_clazz (  id int COMMENT 'ems课程id(非自增)',  create\_date\_time STRING COMMENT '创建时间',  update\_date\_time STRING COMMENT '最后更新时间',  deleted STRING COMMENT '是否被删除（禁用）',  itcast\_school\_id STRING COMMENT 'ems校区ID',  itcast\_school\_name STRING COMMENT 'ems校区名称',  itcast\_subject\_id STRING COMMENT 'ems学科ID',  itcast\_subject\_name STRING COMMENT 'ems学科名称',  itcast\_brand STRING COMMENT 'ems品牌',  clazz\_type\_state STRING COMMENT '班级类型状态',  clazz\_type\_name STRING COMMENT '班级类型名称',  teaching\_mode STRING COMMENT '授课模式',  start\_time STRING COMMENT '开班时间',  end\_time STRING COMMENT '毕业时间',  comment STRING COMMENT '备注',  detail STRING COMMENT '详情(比如：27期)',  uncertain STRING COMMENT '待定班(0:否,1:是)',  tenant int COMMENT '租户',  ends\_time STRING COMMENT '有效时间') comment '班级信息表' PARTITIONED BY(starts\_time STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' stored as orc TBLPROPERTIES ('orc.compress'='SNAPPY','orc.create.index'='true','orc.bloom.filter.columns'='id'); |

##### DWD

###### customer\_relationship\_dwd表

虽然报名用户的DWD层和意向客户的DWD层都是由customer\_relationship原始表转化而来，但是在报名看板中，统计分析主要是**按照报名（支付）的时间**来进行的，而意向用户看板中则是以**意向时间（创建时间）**进行统计，因此分区的字段也不相同。

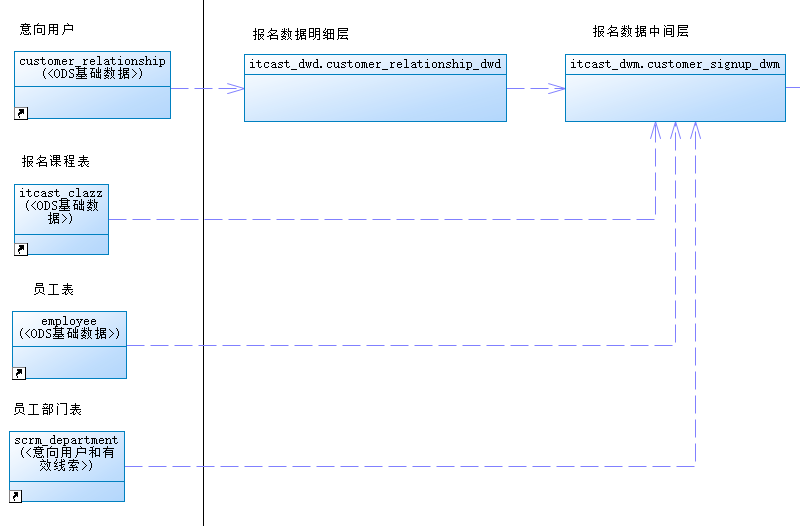
报名看板需要按照报名的年月日来进行分区，才能够提升后续的**增量分析性能**。所以**DWD层需要重新创建，不复用意向用户主题。**



|  |
| --- |
| CREATE TABLE IF NOT EXISTS itcast\_dwd.customer\_relationship\_dwd (  `id` int COMMENT '客户关系id',  `customer\_id` int COMMENT '所属客户id',  `origin\_type` STRING COMMENT '数据来源',  `payment\_time` STRING COMMENT '支付状态变动时间字符串',  `payment\_time\_hour` STRING COMMENT '支付状态变动小时',  `itcast\_clazz\_id` int COMMENT '报名课程id',  `creator` int COMMENT '创建人id',  `origin\_type\_stat` STRING COMMENT '数据来源:0.线下；1.线上' ) comment '客户关系表' PARTITIONED BY (payment\_time\_year String, payment\_time\_month String, payment\_time\_day String) CLUSTERED BY(id) sorted by(id) into 10 buckets ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' stored as orc TBLPROPERTIES ('orc.compress'='SNAPPY','orc.create.index'='true','orc.bloom.filter.columns'='itcast\_clazz\_id,creator'); |

##### DWM

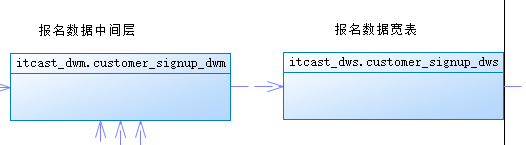
DWM层在DWD基础上，关联出学校信息、学科信息以及咨询中心的维度信息。



|  |
| --- |
| drop table itcast\_dwm.customer\_signup\_dwm; CREATE TABLE IF NOT EXISTS itcast\_dwm.customer\_signup\_dwm (  `customer\_id` int COMMENT '报名客户id',  `itcast\_school\_id` STRING COMMENT '学校id',  `itcast\_school\_name` STRING COMMENT '学校namne',  `itcast\_subject\_id` STRING COMMENT '学科id',  `itcast\_subject\_name` STRING COMMENT '学科name',  `tdepart\_id` int COMMENT '咨询中心id',  `tdepart\_name` STRING COMMENT '咨询中心name',  `origin\_type` STRING COMMENT '来源渠道',  origin\_type\_stat STRING COMMENT '数据来源:0.线下；1.线上',  `payment\_time\_hour` STRING COMMENT '最后更新时间') comment '报名人数统计中间表' PARTITIONED BY (payment\_time\_year String, payment\_time\_month String, payment\_time\_day String) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' stored as orc TBLPROPERTIES ('orc.compress'='SNAPPY'); |

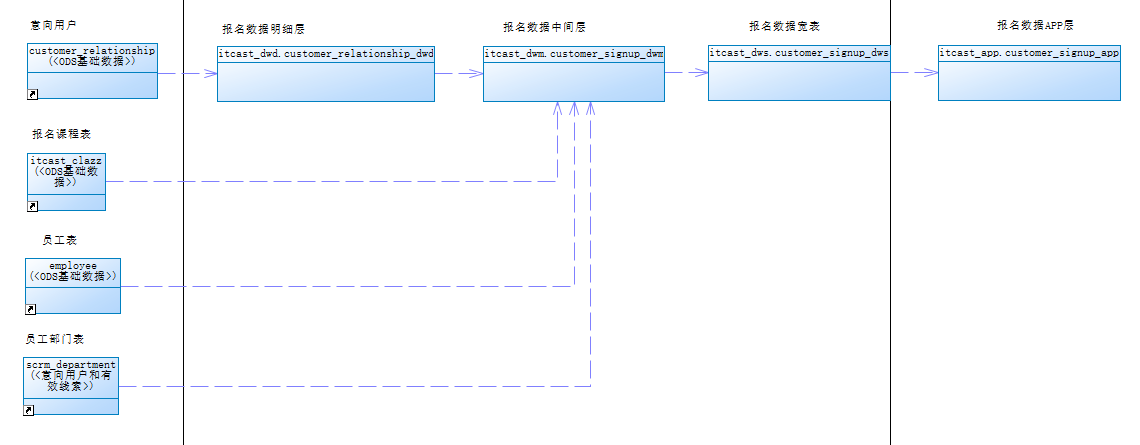
##### DWS

在DWM层基础上，按照业务分组类型进行统计，其中时间维度使用最小粒度的小时维度。因为APP要按照年月日统计，所以DWS按照年月日来分区。



|  |
| --- |
| drop table itcast\_dws.customer\_signup\_dws; CREATE TABLE IF NOT EXISTS itcast\_dws.customer\_signup\_dws (  `signup\_num` int COMMENT '报名人数',  `itcast\_school\_id` STRING COMMENT '学校id',  `itcast\_school\_name` STRING COMMENT '学校namne',  `itcast\_subject\_id` STRING COMMENT '学科id',  `itcast\_subject\_name` STRING COMMENT '学科name',  `tdepart\_id` int COMMENT '咨询中心id',  `tdepart\_name` STRING COMMENT '咨询中心name',  `origin\_type` STRING COMMENT '来源渠道',  origin\_type\_stat STRING COMMENT '数据来源:0.线下；1.线上',  `payment\_time\_hour` STRING COMMENT '最后更新时间',  **groupType** STRING COMMENT '业务分组类型：1：校区、学科组合分组；2：来源渠道分组；3：咨询中心分组；4:所有',  **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。'  ) comment '报名人数统计业务表' PARTITIONED BY (payment\_time\_year String, payment\_time\_month String, payment\_time\_day String) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' stored as orc TBLPROPERTIES ('orc.compress'='SNAPPY'); |

### 全量流程



#### 数据采集

##### customer\_relationship、employee、scrm\_department表

可复用意向客户主题ODS层数据。

##### itcast\_clazz表

|  |
| --- |
| sqoop import \  --connect jdbc:mysql://192.168.52.150:3306/scrm \  --username root \  --password 123456 \  --query 'select \*,FROM\_UNIXTIME(unix\_timestamp(),"%Y-%m-%d")as starts\_time,date\_format("9999-12-31","%Y-%m-%d") as ends\_time from itcast\_clazz where $CONDITIONS' \  --hcatalog-database itcast\_dimen \  --hcatalog-table itcast\_clazz \  -m 100 \  --split-by id |

#### 数据清洗转换

##### 判断函数

###### If函数

if和case差不多，都是处理单个列的查询结果

语法: if(boolean testCondition, T valueTrue, T valueFalseOrNull)

返回值: T

说明: 当条件testCondition为TRUE时，返回valueTrue；否则返回valueFalseOrNull。

示例：if（条件表达式，结果1，结果2）相当于java中的三目运算符,只是if后面的表达式类型可以不一样。

|  |
| --- |
| select if(a=a,’bbbb’,111) fromlxw\_dual;  bbbb  select if(1<2,100,200) fromlxw\_dual;  200 |

**nvl函数**

###### nvl

对于null值的判断。

语法：nvl(T value, T default\_value)

返回值：T

说明：如果value值为NULL就返回default\_value，否则返回value，主要功能就是实现null值替换。

示例：

|  |
| --- |
| select nvl(principal,1) from mydb.aaaaa;  1 |

###### COALESCE

非空查找函数。

语法: COALESCE(T v1, T v2, …)

返回值: T

说明: 返回参数中的第一个非空值；如果所有值都为NULL，那么返回NULL，可以有很多个参数。

示例：

|  |
| --- |
| select COALESCE(null,’aaa’,50) from lxw\_dual;  aaa |

###### CASE WHEN THEN

两种语法结构：

1. 条件判断函数case A when B THEN C ,case和when是分开的

语法：CASE a WHEN b THEN c [WHEN d THEN e]\* [ELSE f] END

说明：如果a等于b，那么返回c；如果a等于d，那么返回e；否则返回f。注意这种when的判断条件可以有很多个。如CASE 4 WHEN 5 THEN 5 WHEN 4 THEN 4 ELSE 3 END 将返回4。

示例：

|  |
| --- |
| select case 100  when 50 then 'tom'  when 100 then 'mary'  else 'tim' end  from lxw\_dual;  mary  select case 200 when 50 then 'tom'when 100 then 'mary' else 'tim' end from lxw\_dual;  tim |

1. 这种语法第一个case when是一起的，用于处理单个列的查询结果，类似于if判断

语法：CASE WHEN a THEN b [WHEN c THEN d]\* [ELSE e] END

说明：如果a为TRUE,则返回b；如果c为TRUE，则返回d；否则返回e 如：CASE WHEN 5>0 THEN 5 WHEN 4>0 THEN 4 ELSE 0 END 将返回5；CASE WHEN 5<0 THEN 5 WHEN 4<0 THEN 4 ELSE 0 END 将返回0。

示例：

|  |
| --- |
| select case when 1=2 then 'tom' when 2=2 then 'mary' else 'tim' end from lxw\_dual;  mary  select case when 1=1 then 'tom' when 2=2 then 'mary' else 'tim' end from lxw\_dual;  tom |

###### isnull

语法：isnull( a )

返回值：boolean

说明：如果a为null就返回true，否则返回false。

###### isnotnull

语法：isnotnull(a)

返回值：boolean

说明：如果a为非null就返回true，否则返回false。

##### 优化

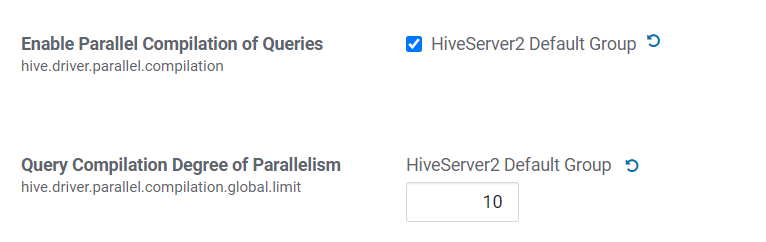
###### Hive并行操作

Hive编译查询限制

Hive默认同时只能编译一段HiveQL，并上锁。

将hive.driver.parallel.compilation设置为true，各个会话可以同时编译查询，提高团队工作效率。否则如果在UDF中执行了一段HiveQL，或者多个用户同时使用的话, 就会锁住。

修改hive.driver.parallel.compilation.global.limit的值，0或负值为无限制，可根据团队人员和硬件进行修改，以保证同时编译查询。



Hive不同阶段任务并行执行

Hive会将一个查询转化为一个或多个阶段，包括：MapReduce阶段、抽样阶段、合并阶段、limit阶段等。默认情况下，一次只执行一个阶段。不过，如果某些阶段不是互相依赖，是可以并行执行的。

set hive.exec.parallel=true,可以开启并发执行，默认为false。

set hive.exec.parallel.thread.number=16; //同一个sql允许的最大并行度，默认为8。

|  |
| --- |
| set hive.exec.parallel=true; set hive.exec.parallel.thread.number=16; |

###### Hive小文件合并

此部分设置，要根据硬件内存来进行调整，个人电脑配置较低，不建议修改。

hive.merge.mapfiles

是否开启合并Map端小文件，在Map-only的任务结束时合并小文件，true是打开。

hive.merge.mapredfiles

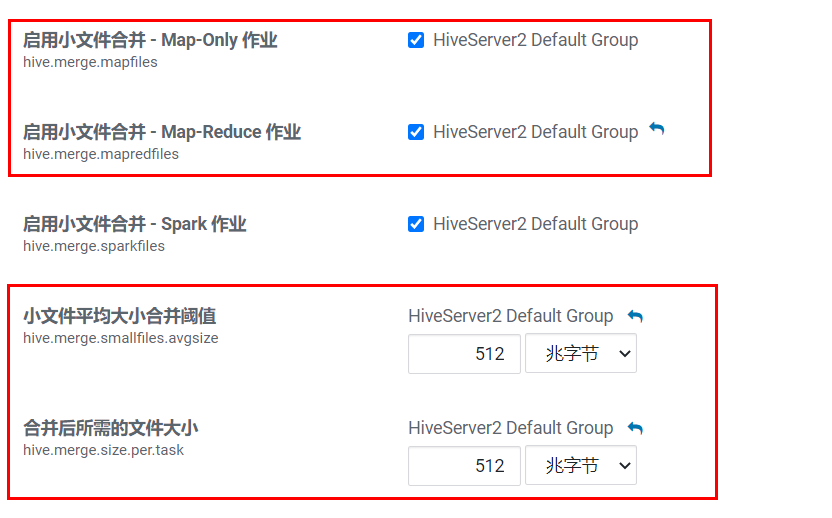
是否开启合并Reduce端小文件，在map-reduce作业结束时合并小文件。true是打开。

hive.merge.size.per.task

合并后MR输出文件的大小，默认为256M。

hive.merge.smallfiles.avgsize

当输出文件的平均大小小于此设置值时，启动一个独立的map-reduce任务进行文件merge，默认值为16M。



###### 矢量化查询

hive的默认查询执行引擎一次处理一行，而矢量化查询执行是一种hive特性，目的是按照每批1024行读取数据，并且一次性对整个记录整合（而不是对单条记录）应用操作，注意：要使用矢量化查询执行，就必须以**ORC格式**存储数据。

|  |
| --- |
| set hive.vectorized.execution.enabled=true; |

###### 读取零拷贝

ORC可以使用新的HDFS缓存API和ZeroCopy读取器来避免在扫描文件时将额外的数据复制到内存中。

|  |
| --- |
| set hive.exec.orc.zerocopy=true; |

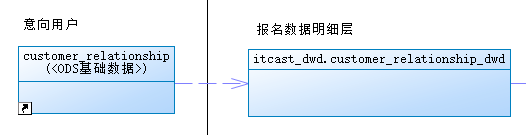
##### 分析

customer\_relationship表中包含有用户的报名信息，ODS复用后，在DWD层进行清洗转换。

清洗：DWD只保留**未被删除的、customer不为空的、并且必须是已支付'PAID'**（已报名）数据。

转换：

1. 如果origin\_type是NETSERVICE和PRESIGNUP类型，即为1线上，否则为0线下。
2. 支付时间切割获取到年月日时的字符值，并分别存为单独的字段方便后续统计。



##### DWD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=8; *--矢量化查询* set hive.vectorized.execution.enabled=true; *--读取零拷贝* set hive.exec.orc.zerocopy=true;  INSERT INTO itcast\_dwd.customer\_relationship\_dwd PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  id,  customer\_id,  *nvl*(origin\_type, -1) origin\_type,  payment\_time,  *substr*(payment\_time, 12, 2) as payment\_time\_hour,  *nvl*(itcast\_clazz\_id, -1) itcast\_clazz\_id,  *nvl*(creator, -1) creator,  *if*(origin\_type='NETSERVICE' or origin\_type='PRESIGNUP', '1', '0') origin\_type\_stat,  *substr*(payment\_time, 1, 4) payment\_time\_year,  *substr*(payment\_time, 6, 2) payment\_time\_month,  *substr*(payment\_time, 9, 2) payment\_time\_day from itcast\_ods.customer\_relationship WHERE deleted = 0 AND customer\_id IS NOT NULL AND payment\_state='PAID'; |

##### 测试

可以采用分桶的方式进行采样测试。

|  |
| --- |
| INSERT INTO itcast\_dwd.customer\_relationship\_dwd PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  id,  deleted,  customer\_id,  *nvl*(origin\_type, -1) origin\_type,  payment\_state,  payment\_time,  *substr*(payment\_time, 12, 2) as payment\_time\_hour,  *nvl*(itcast\_clazz\_id, -1) itcast\_clazz\_id,  *nvl*(creator, -1) creator,  *if*(origin\_type='NETSERVICE' or origin\_type='PRESIGNUP', '1', '0') origin\_type\_stat,  *substr*(payment\_time, 1, 4) payment\_time\_year,  *substr*(payment\_time, 6, 2) payment\_time\_month,  *substr*(payment\_time, 9, 2) payment\_time\_day from itcast\_ods.customer\_relationship tablesample ( bucket 1 out of 10 on id) WHERE deleted = 0 AND customer\_id IS NOT NULL AND payment\_state='PAID' AND payment\_time > '2020-01-01 00:00:00'; |

#### 统计分析

##### 优化

###### 关联优化器

在Hive的一些复杂关联查询中，可能同时还包含有group by等能够触发shuffle的操作，有些时候shuffle操作是可以共享的，通过关联优化器选项，可以尽量减少复杂查询中的shuffle，从而提升性能。

|  |
| --- |
| set hive.optimize.correlation=true; |

###### 数据倾斜Skewin

如果数据量很大或者出现了数据倾斜比较严重的情况，如何来优化呢？

map join、Bucket-Map join、SMB join

详见意向客户主题看板。

表连接数据倾斜(Join skew in)

运行时优化

**set hive.optimize.skewjoin=true;**

默认关闭。

如果大表和大表进行join操作，则可采用skewjoin(倾斜关联)来开启对倾斜数据的优化。

**skewjoin原理：**

1. 对于skewjoin.key，在执行job时，将它们存入临时的HDFS目录，其它数据正常执行
2. 对倾斜数据开启map join操作（多个map并行处理），对非倾斜值采取普通join操作
3. 将倾斜数据集和非倾斜数据集进行合并Union操作。

开启skewin以后，究竟多大的数据才会被认为是倾斜了的数据呢？

**set hive.skewjoin.key=****100000;**

默认值100000。

如果join的key对应的记录条数超过这个值，就认为这个key产生了数据倾斜，则会对其进行分拆优化。

编译时优化

上面的配置项其实应该理解为hive.optimize.skewjoin.runtime，也就是sql运行时来对偏斜信息进行优化；除此之外还有另外一个配置：

**set hive.optimize.skewjoin.****compiletime=true;**

默认关闭。

此参数的用处和上面的hive.optimize.skewjoin一致，但在编译sql时就已经将执行计划优化完毕。但要注意的是，只有在表的元数据中存储的有数据倾斜信息时，才能生效。因此建议runtime和compiletime都设置为true。

可以通过建表语句来指定数据倾斜元数据：

|  |
| --- |
| CREATE TABLE list\_bucket\_single (key STRING, value STRING) *-- 倾斜的字段和需要拆分的key值* SKEWED BY (key) ON (1,5,6) *-- 为倾斜值创建子目录单独存放* [STORED AS DIRECTORIES]; |

Union优化

应用了表连接倾斜优化以后，会在执行计划中插入一个新的union操作，此时建议开启对union的优化配置：

**set hive.optimize.union.remove=true;**

默认关闭。

此项配置减少对Union all子查询中间结果的二次读写，可以避免union输出的额外扫描过程，当我们开启了skewjoin时尤其有用，建议同时开启。

|  |
| --- |
| set hive.optimize.skewjoin=true;  set hive.optimize.skewjoin.compiletime=true;  set hive.optimize.union.remove=true; |

分组统计数据倾斜(Groupby skew in)

Map阶段聚合

**hive.map.aggr=true;**

开启map端combiner。此配置可以在group by语句中提高HiveQL聚合的执行性能。这个设置可以将顶层的聚合操作放在Map阶段执行，从而减轻数据传输和Reduce阶段的执行时间，提升总体性能。默认开启，无需显示声明。



两个MRJob

**hive.groupby.skewindata=true;**

默认关闭。

这个配置项是用于决定group by操作是否支持倾斜数据的负载均衡处理。当数据出现倾斜时，如果该变量设置为true，那么Hive会自动进行负载均衡。

当选项设定为 true，生成的查询计划会有两个 MR Job。

第一个MR Job中，Map 的输出结果集合会随机分布到Reduce中，每个Reduce做部分聚合操作，并输出结果，这样处理的结果是相同的Group By Key有可能被分发到不同的Reduce中，从而达到负载均衡的目的；

第二个MR Job再根据预处理的数据结果按照Group By Key分布到Reduce中（这个过程可以保证相同的Group By Key被分布到同一个Reduce中），最后完成最终的聚合操作。

注意：在多个列上进行的去重操作与hive环境变量hive.groupby.skewindata存在冲突。

当hive.groupby.skewindata=true时，hive不支持多列上的去重操作，并报错：

|  |
| --- |
| Error in semantic analysis: DISTINCT on different columns notsupported with skew in data. |

比如：

|  |
| --- |
| (1) SELECT count(DISTINCT uid) FROM log  (2) SELECT ip, count(DISTINCT uid) FROM log GROUP BY ip  (3) SELECT ip, count(DISTINCT uid, uname) FROMlog GROUP BY ip  (4) SELECT ip, count(DISTINCT uid), count(DISTINCT uname) FROMlog GROUP BY ip |

1、2、3能够正常执行，但是4会报错。

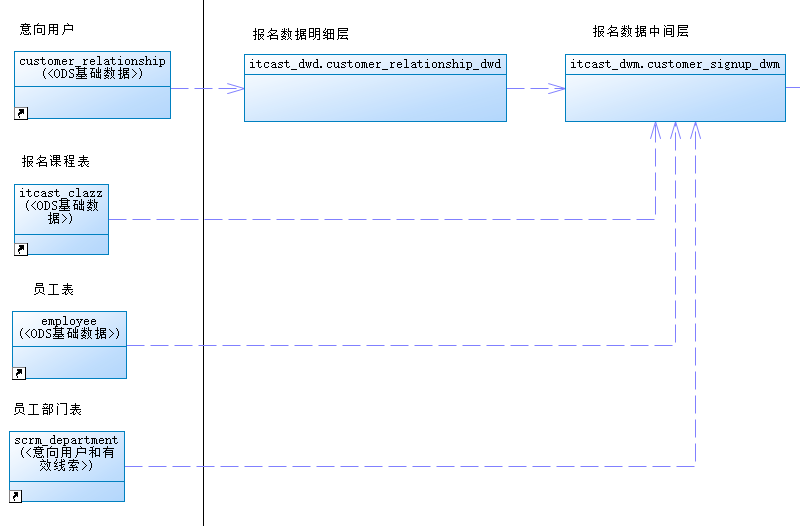
##### DWM

###### 分析

DWM层在DWD基础上，关联出学校信息、学科信息以及咨询中心的维度信息。

通过itcast\_clazz\_id关联itcast\_clazz表的id字段，来获取到学校、学科的信息；

通过creator关联employee表获取tdepart\_id咨询中心单位id；再用employee的department\_id和scrm\_department表id关联获取单位名称name。



###### 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; *--矢量化查询* set hive.vectorized.execution.enabled=true; *--关联优化器* set hive.optimize.correlation=true; *--读取零拷贝* set hive.exec.orc.zerocopy=true; *--join数据倾斜* set hive.optimize.skewjoin=true; *-- set hive.skewjoin.key=100000;* set hive.optimize.skewjoin.compiletime=true; set hive.optimize.union.remove=true;   INSERT into itcast\_dwm.customer\_signup\_dwm PARTITION (payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  dwd.customer\_id,  clazz.itcast\_school\_id,  clazz.itcast\_school\_name,  clazz.itcast\_subject\_id,  clazz.itcast\_subject\_name,  e.tdepart\_id,  dept.name as tdepart\_name,  dwd.origin\_type,  dwd.origin\_type\_stat,  dwd.payment\_time\_hour,  dwd.payment\_time\_year,  dwd.payment\_time\_month,  dwd.payment\_time\_day FROM itcast\_dwd.customer\_relationship\_dwd dwd LEFT JOIN itcast\_dimen.itcast\_clazz clazz on dwd.itcast\_clazz\_id=clazz.id LEFT JOIN itcast\_dimen.employee e on dwd.creator=e.id LEFT JOIN itcast\_dimen.scrm\_department dept on e.tdepart\_id=dept.id; |

##### DWS

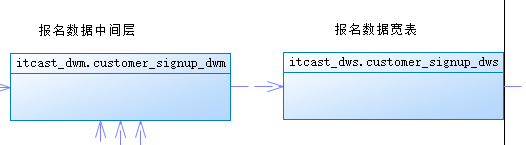
###### 分析

DWS在DWM层的基础上，按照维度进行聚合统计，得到统计宽表。

新增两个字段：产品属性维度标识字段、聚合时间维度标识字段。

产品属性维度字段标识：1、校区、学科组合分组；2、来源渠道分组；3、咨询中心分组；4、所有。

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。



###### 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; *--矢量化查询* set hive.vectorized.execution.enabled=true; *--关联优化器* set hive.optimize.correlation=true; *--读取零拷贝* set hive.exec.orc.zerocopy=true; *--join数据倾斜* set hive.optimize.skewjoin=true; *-- set hive.skewjoin.key=100000;* set hive.optimize.skewjoin.compiletime=true; set hive.optimize.union.remove=true; *-- group倾斜* set hive.groupby.skewindata=true;  *--校区、学科组合分组* *--小时* INSERT INTO itcast\_dws.customer\_signup\_dws PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  *count*(dwm.customer\_id) as signup\_num,  dwm.itcast\_school\_id,  dwm.itcast\_school\_name,  dwm.itcast\_subject\_id,  dwm.itcast\_subject\_name,  -1 as tdepart\_id,  '-1' tdepart\_name,  '-1' origin\_type,  dwm.origin\_type\_stat,  dwm.payment\_time\_hour,  '1' grouptype,   '1' as time\_type,  dwm.payment\_time\_year,  dwm.payment\_time\_month,  dwm.payment\_time\_day from itcast\_dwm.customer\_signup\_dwm dwm GROUP BY dwm.origin\_type\_stat,  dwm.payment\_time\_year, dwm.payment\_time\_month, dwm.payment\_time\_day, dwm.payment\_time\_hour,  dwm.itcast\_school\_id, dwm.itcast\_school\_name, dwm.itcast\_subject\_id, dwm.itcast\_subject\_name;  *--天* INSERT INTO itcast\_dws.customer\_signup\_dws PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  *count*(dwm.customer\_id) as signup\_num,  dwm.itcast\_school\_id,  dwm.itcast\_school\_name,  dwm.itcast\_subject\_id,  dwm.itcast\_subject\_name,  -1 as tdepart\_id,  '-1' tdepart\_name,  '-1' origin\_type,  dwm.origin\_type\_stat,  '-1' payment\_time\_hour,  '1' grouptype,   '2' as time\_type,  dwm.payment\_time\_year,  dwm.payment\_time\_month,  dwm.payment\_time\_day from itcast\_dwm.customer\_signup\_dwm dwm GROUP BY dwm.origin\_type\_stat,  dwm.payment\_time\_year, dwm.payment\_time\_month, dwm.payment\_time\_day,   dwm.itcast\_school\_id, dwm.itcast\_school\_name, dwm.itcast\_subject\_id, dwm.itcast\_subject\_name;  *--月、年省略*  *--来源渠道分组 --小时* INSERT INTO itcast\_dws.customer\_signup\_dws PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  *count*(dwm.customer\_id) as signup\_num,  '-1' itcast\_school\_id,  '-1' itcast\_school\_name,  '-1' itcast\_subject\_id,  '-1' itcast\_subject\_name,  -1 as tdepart\_id,  '-1' tdepart\_name,  dwm.origin\_type,   '1' as time\_type,  dwm.origin\_type\_stat,  dwm.payment\_time\_hour,  '2' grouptype,  dwm.payment\_time\_year,  dwm.payment\_time\_month,  dwm.payment\_time\_day from itcast\_dwm.customer\_signup\_dwm dwm GROUP BY dwm.origin\_type\_stat,  dwm.payment\_time\_year, dwm.payment\_time\_month, dwm.payment\_time\_day, dwm.payment\_time\_hour,  **dwm.origin\_type**; *--天、月、年省略*  *--咨询中心分组 --小时* INSERT INTO itcast\_dws.customer\_signup\_dws PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  *count*(dwm.customer\_id) as signup\_num,  '-1' itcast\_school\_id,  '-1' itcast\_school\_name,  '-1' itcast\_subject\_id,  '-1' itcast\_subject\_name,  dwm.tdepart\_id,  dwm.tdepart\_name,  '-1' origin\_type,  dwm.origin\_type\_stat,  dwm.payment\_time\_hour,  '3' grouptype,   '1' as time\_type,  dwm.payment\_time\_year,  dwm.payment\_time\_month,  dwm.payment\_time\_day from itcast\_dwm.customer\_signup\_dwm dwm GROUP BY dwm.origin\_type\_stat,  dwm.payment\_time\_year, dwm.payment\_time\_month, dwm.payment\_time\_day, dwm.payment\_time\_hour,  **dwm.tdepart\_id, dwm.tdepart\_name**; *--天、月、年省略*  *--总数分组 --小时* INSERT INTO itcast\_dws.customer\_signup\_dws PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  *count*(dwm.customer\_id) as signup\_num,  '-1' itcast\_school\_id,  '-1' itcast\_school\_name,  '-1' itcast\_subject\_id,  '-1' itcast\_subject\_name,  -1 as tdepart\_id,  '-1' tdepart\_name,  '-1' origin\_type,  dwm.origin\_type\_stat,  dwm.payment\_time\_hour,  '4' grouptype,   '1' as time\_type,  dwm.payment\_time\_year,  dwm.payment\_time\_month,  dwm.payment\_time\_day from itcast\_dwm.customer\_signup\_dwm dwm GROUP BY dwm.origin\_type\_stat,  dwm.payment\_time\_year, dwm.payment\_time\_month, dwm.payment\_time\_day, dwm.payment\_time\_hour;  *--天、月、年省略……* |

#### 导出数据

##### 创建Mysql表

|  |
| --- |
| CREATE TABLE `customer\_signup\_app` (  `signup\_num` int(11) COMMENT '报名人数',  `itcast\_school\_id` varchar(32) COMMENT '学校id',  `itcast\_school\_name` varchar(32) COMMENT '学校name',  `itcast\_subject\_id` varchar(32) COMMENT '学科id',  `itcast\_subject\_name` varchar(32) COMMENT '学科name',  `tdepart\_id` int(11) COMMENT '咨询中心id',  `tdepart\_name` varchar(32) COMMENT '咨询中心name',  `origin\_type` varchar(32) COMMENT '来源渠道',  `origin\_type\_stat` varchar(32) COMMENT '数据来源:0.线下；1.线上',  `payment\_time\_month` varchar(32) COMMENT '月信息',  `payment\_time\_day` varchar(32) COMMENT '日信息',  `payment\_time\_hour` varchar(32) COMMENT '最后更新时间',  `groupType` varchar(32) COMMENT '统计分组类型：1：校区、学科组合分组；2：来源渠道分组；3：咨询中心分组；4:所有',  `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。',  `payment\_time\_year` 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 customer\_signup\_app \  --hcatalog-database itcast\_dws \  --hcatalog-table customer\_signup\_dws \  -m 100 |

### 增量流程

#### 数据采集

同全量过程。

#### 数据清洗转换

##### 分析

增量计算时，只需要将新抽取的数据进行清洗转换即可。

##### 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; *--矢量化查询* set hive.vectorized.execution.enabled=true; *--关联优化器* set hive.optimize.correlation=true; *--读取零拷贝* set hive.exec.orc.zerocopy=true;  INSERT INTO itcast\_dwd.customer\_relationship\_dwd PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  id,  deleted,  customer\_id,  *nvl*(origin\_type, -1) origin\_type,  payment\_state,  payment\_time,  *substr*(payment\_time, 12, 2) as payment\_time\_hour,  *nvl*(itcast\_clazz\_id, -1) itcast\_clazz\_id,  *nvl*(creator, -1) creator,  *if*(origin\_type='NETSERVICE' or origin\_type='PRESIGNUP', '1', '0') origin\_type\_stat,  *substr*(payment\_time, 1, 4) payment\_time\_year,  *substr*(payment\_time, 6, 2) payment\_time\_month,  *substr*(payment\_time, 9, 2) payment\_time\_day from itcast\_ods.customer\_relationship WHERE deleted = 0 AND customer\_id IS NOT NULL AND payment\_state='PAID' and *substr*(payment\_time, 1, 10) >='2037-02-01'; |

###### Shell

|  |
| --- |
| #! /bin/bash  #上个月1日  Last\_Month\_DATE=$(date -d "$(date +%Y%m)01 last month" +%Y-%m-01)  out\_put=$(${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;  --矢量化查询  set hive.vectorized.execution.enabled=true;  --关联优化器  set hive.optimize.correlation=true;  --读取零拷贝  set hive.exec.orc.zerocopy=true;  INSERT INTO itcast\_dwd.customer\_relationship\_dwd PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day)  SELECT  id,  deleted,  customer\_id,  nvl(origin\_type, -1) origin\_type,  payment\_state,  payment\_time,  substr(payment\_time, 12, 2) as payment\_time\_hour,  nvl(itcast\_clazz\_id, -1) itcast\_clazz\_id,  nvl(creator, -1) creator,  if(origin\_type='NETSERVICE' or origin\_type='PRESIGNUP', '1', '0') origin\_type\_stat,  substr(payment\_time, 1, 4) payment\_time\_year,  substr(payment\_time, 6, 2) payment\_time\_month,  substr(payment\_time, 9, 2) payment\_time\_day  from itcast\_ods.customer\_relationship  WHERE deleted = 0 AND customer\_id IS NOT NULL AND payment\_state='PAID'  and substr(payment\_time, 1, 10) >='${Last\_Month\_DATE}';  ") |

#### 统计分析

##### 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; *--矢量化查询* set hive.vectorized.execution.enabled=true; *--关联优化器* set hive.optimize.correlation=true; *--读取零拷贝* set hive.exec.orc.zerocopy=true; *--join数据倾斜* set hive.optimize.skewjoin=true; *-- set hive.skewjoin.key=100000;* set hive.optimize.skewjoin.compiletime=true; set hive.optimize.union.remove=true;   INSERT into itcast\_dwm.customer\_signup\_dwm PARTITION (payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  dwd.customer\_id,  clazz.itcast\_school\_id,  clazz.itcast\_school\_name,  clazz.itcast\_subject\_id,  clazz.itcast\_subject\_name,  e.tdepart\_id,  dept.name as tdepart\_name,  dwd.origin\_type,  dwd.origin\_type\_stat,  dwd.payment\_time\_hour,  dwd.payment\_time\_year,  dwd.payment\_time\_month,  dwd.payment\_time\_day FROM itcast\_dwd.customer\_relationship\_dwd dwd LEFT JOIN itcast\_dimen.itcast\_clazz clazz on dwd.itcast\_clazz\_id=clazz.id LEFT JOIN itcast\_dimen.employee e on dwd.creator=e.id LEFT JOIN itcast\_dimen.scrm\_department dept on e.tdepart\_id=dept.id where *substr*(dwd.payment\_time, 1, 10) >= '2037-02-01' ; |

###### Shell脚本

|  |
| --- |
| #! /bin/bash  HIVE\_HOME=/usr/bin/hive  Last\_Month=$(date -d "$(date +%Y%m)01 last month" +%Y-%m-01)  ${HIVE\_HOME} -S -e "  --分区  SET hive.exec.dynamic.partition=true;  SET hive.exec.dynamic.partition.mode=nonstrict;  set hive.exec.max.dynamic.partitions.pernode=10000;  set hive.exec.max.dynamic.partitions=100000;  set hive.exec.max.created.files=150000;  --hive压缩  set hive.exec.compress.intermediate=true;  set hive.exec.compress.output=true;  --写入时压缩生效  set hive.exec.orc.compression.strategy=COMPRESSION;  --分桶  set hive.enforce.bucketing=true;  set hive.enforce.sorting=true;  set hive.optimize.bucketmapjoin = true;  set hive.auto.convert.sortmerge.join=true;  set hive.auto.convert.sortmerge.join.noconditionaltask=true;  --并行执行  set hive.exec.parallel=true;  set hive.exec.parallel.thread.number=8;  --矢量化查询  set hive.vectorized.execution.enabled=true;  --关联优化器  set hive.optimize.correlation=true;  --读取零拷贝  set hive.exec.orc.zerocopy=true;  --join数据倾斜  set hive.optimize.skewjoin=true;  -- set hive.skewjoin.key=100000;  set hive.optimize.skewjoin.compiletime=true;  set hive.optimize.union.remove=true;  INSERT into itcast\_dwm.customer\_signup\_dwm PARTITION (payment\_time\_year, payment\_time\_month, payment\_time\_day)  SELECT  dwd.customer\_id,  clazz.itcast\_school\_id,  clazz.itcast\_school\_name,  clazz.itcast\_subject\_id,  clazz.itcast\_subject\_name,  e.tdepart\_id,  dept.name as tdepart\_name,  dwd.origin\_type,  dwd.origin\_type\_stat,  dwd.payment\_time\_hour,  dwd.payment\_time\_year,  dwd.payment\_time\_month,  dwd.payment\_time\_day  FROM itcast\_dwd.customer\_relationship\_dwd dwd  LEFT JOIN itcast\_dimen.itcast\_clazz clazz on dwd.itcast\_clazz\_id=clazz.id  LEFT JOIN itcast\_dimen.employee e on dwd.creator=e.id  LEFT JOIN itcast\_dimen.scrm\_department dept on e.tdepart\_id=dept.id  where substr(dwd.payment\_time, 1, 10) >= '$Last\_Month' ;  " |

##### DWS

###### 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; *--矢量化查询* set hive.vectorized.execution.enabled=true; *--关联优化器* set hive.optimize.correlation=true; *--读取零拷贝* set hive.exec.orc.zerocopy=true; *--join数据倾斜* set hive.optimize.skewjoin=true; *-- set hive.skewjoin.key=100000;* set hive.optimize.skewjoin.compiletime=true; set hive.optimize.union.remove=true; *-- group倾斜* set hive.groupby.skewindata=true;   *--校区、学科组合分组 --小时* INSERT INTO itcast\_dws.customer\_signup\_dws PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  *count*(dwm.customer\_id) as signup\_num,  dwm.itcast\_school\_id,  dwm.itcast\_school\_name,  dwm.itcast\_subject\_id,  dwm.itcast\_subject\_name,  -1 as tdepart\_id,  '-1' tdepart\_name,  '-1' origin\_type,  dwm.origin\_type\_stat,  dwm.payment\_time\_hour,  '1' grouptype,   '1' as time\_type,  dwm.payment\_time\_year,  dwm.payment\_time\_month,  dwm.payment\_time\_day from itcast\_dwm.customer\_signup\_dwm dwm where *CONCAT\_WS*('-',dwm.payment\_time\_year,dwm.payment\_time\_month,dwm.payment\_time\_day) >= '2037-02-01' GROUP BY dwm.origin\_type\_stat,  dwm.payment\_time\_year, dwm.payment\_time\_month, dwm.payment\_time\_day, dwm.payment\_time\_hour,  dwm.itcast\_school\_id, dwm.itcast\_school\_name, dwm.itcast\_subject\_id, dwm.itcast\_subject\_name; *--天* INSERT INTO itcast\_dws.customer\_signup\_dws PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  *count*(dwm.customer\_id) as signup\_num,  dwm.itcast\_school\_id,  dwm.itcast\_school\_name,  dwm.itcast\_subject\_id,  dwm.itcast\_subject\_name,  -1 as tdepart\_id,  '-1' tdepart\_name,  '-1' origin\_type,  dwm.origin\_type\_stat,  '-1' payment\_time\_hour,  '1' grouptype,   '2' as time\_type,  dwm.payment\_time\_year,  dwm.payment\_time\_month,  dwm.payment\_time\_day from itcast\_dwm.customer\_signup\_dwm dwm where *CONCAT\_WS*('-',dwm.payment\_time\_year,dwm.payment\_time\_month,dwm.payment\_time\_day) >= '2037-02-01' GROUP BY dwm.origin\_type\_stat,  dwm.payment\_time\_year, dwm.payment\_time\_month, dwm.payment\_time\_day,   dwm.itcast\_school\_id, dwm.itcast\_school\_name, dwm.itcast\_subject\_id, dwm.itcast\_subject\_name; *--月、年省略   --来源渠道分组 --小时* INSERT INTO itcast\_dws.customer\_signup\_dws PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  *count*(dwm.customer\_id) as signup\_num,  '-1' itcast\_school\_id,  '-1' itcast\_school\_name,  '-1' itcast\_subject\_id,  '-1' itcast\_subject\_name,  -1 as tdepart\_id,  '-1' tdepart\_name,  dwm.origin\_type,  dwm.origin\_type\_stat,  dwm.payment\_time\_hour,  '2' grouptype,   '1' as time\_type,  dwm.payment\_time\_year,  dwm.payment\_time\_month,  dwm.payment\_time\_day from itcast\_dwm.customer\_signup\_dwm dwm where *CONCAT\_WS*('-',dwm.payment\_time\_year,dwm.payment\_time\_month,dwm.payment\_time\_day) >= '2037-02-01' GROUP BY dwm.origin\_type\_stat,  dwm.payment\_time\_year, dwm.payment\_time\_month, dwm.payment\_time\_day, dwm.payment\_time\_hour,  dwm.origin\_type; *--天、月、年省略   --咨询中心分组 --小时* INSERT INTO itcast\_dws.customer\_signup\_dws PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  *count*(dwm.customer\_id) as signup\_num,  '-1' itcast\_school\_id,  '-1' itcast\_school\_name,  '-1' itcast\_subject\_id,  '-1' itcast\_subject\_name,  dwm.tdepart\_id,  dwm.tdepart\_name,  '-1' origin\_type,  dwm.origin\_type\_stat,  dwm.payment\_time\_hour,  '3' grouptype,   '1' as time\_type,  dwm.payment\_time\_year,  dwm.payment\_time\_month,  dwm.payment\_time\_day from itcast\_dwm.customer\_signup\_dwm dwm where *CONCAT\_WS*('-',dwm.payment\_time\_year,dwm.payment\_time\_month,dwm.payment\_time\_day) >= '2037-02-01' GROUP BY dwm.origin\_type\_stat,  dwm.payment\_time\_year, dwm.payment\_time\_month, dwm.payment\_time\_day, dwm.payment\_time\_hour,  dwm.tdepart\_id, dwm.tdepart\_name; *--天、月、年省略   --总数分组 --小时* INSERT INTO itcast\_dws.customer\_signup\_dws PARTITION(payment\_time\_year, payment\_time\_month, payment\_time\_day) SELECT  *count*(dwm.customer\_id) as signup\_num,  '-1' itcast\_school\_id,  '-1' itcast\_school\_name,  '-1' itcast\_subject\_id,  '-1' itcast\_subject\_name,  -1 as tdepart\_id,  '-1' tdepart\_name,  '-1' origin\_type,  dwm.origin\_type\_stat,  dwm.payment\_time\_hour,  '4' grouptype,   '1' as time\_type,  dwm.payment\_time\_year,  dwm.payment\_time\_month,  dwm.payment\_time\_day from itcast\_dwm.customer\_signup\_dwm dwm where *CONCAT\_WS*('-',dwm.payment\_time\_year,dwm.payment\_time\_month,dwm.payment\_time\_day) >= '2037-02-01' GROUP BY dwm.origin\_type\_stat,  dwm.payment\_time\_year, dwm.payment\_time\_month, dwm.payment\_time\_day, dwm.payment\_time\_hour; *--天、月、年省略……* |

###### Shell脚本

略。

#### 导出数据

##### sqoop语句

|  |
| --- |
| sqoop export \  --connect "jdbc:mysql://172.17.0.202:3306/scrm\_bi?useUnicode=true&characterEncoding=utf-8" \  --username root \  --password 123456 \  --table customer\_signup\_app \  --hcatalog-database itcast\_dws \  --hcatalog-table customer\_signup\_dws \  --hcatalog-partition-keys payment\_time\_year \  --hcatalog-partition-values 2016 \  -m 10 |

##### shell脚本

|  |
| --- |
| #! /bin/bash  SQOOP\_HOME=/usr/bin/sqoop  HOST=172.17.0.202  USERNAME="root"  PASSWORD="123456"  PORT=3306  DBNAME="scrm\_bi"  MYSQL=/usr/local/mysql\_5723/bin/mysql  if [[ $1 == "" ]];then  TD\_DATE=`date -d '1 days ago' "+%Y-%m-%d"`  else  TD\_DATE=$1  fi  ${MYSQL} -h${HOST} -P${PORT} -u${USERNAME} -p${PASSWORD} -D${DBNAME} -e "delete from customer\_signup\_app where payment\_time\_year = '${TD\_DATE:0:4}'"  ${SQOOP\_HOME} export \  --connect "jdbc:mysql://${HOST}:${PORT}/${DBNAME}?useUnicode=true&characterEncoding=utf-8" \  --username ${USERNAME} \  --password ${PASSWORD} \  --table customer\_signup\_app \  --hcatalog-database itcast\_dws \  --hcatalog-table customer\_signup\_dws \  --hcatalog-partition-keys payment\_time\_year \  --hcatalog-partition-values ${TD\_DATE:0:4} \  -m 100 |

## 无投诉线索量实现

复用有效线索主题看板。