# 访问咨询主题看板

## 学习目标

了解访问咨询主题看板需求

能够提取出需求中的指标和维度

了解访问客户量指标的分层结构

了解常见的数据格式和压缩格式

理解全量和增量

掌握Hive静态分区和动态分区的用法

了解如何配置HDFS副本数

了解Yarn的基础优化配置

了解Hive的基础优化配置

了解MR与Hive的压缩配置

能够使用Sqoop抽取全量数据

了解Hive常用的时间函数

了解Hive常用的字符串截取函数

能够编写访问量指标的清洗转换SQL

能够编写访问量指标的统计分析SQL

能够使用Sqoop导出全量数据到Mysql

掌握Shell的date命令

掌握Shell的变量替换、命令替换与数学运算语法

掌握Shell的串行与并行

能够编写sqoop导入的Shell脚本

能够使用oozie定时调度sqoop任务

理解增量清洗转换和全量的区别

理解增量统计分析和全量的区别

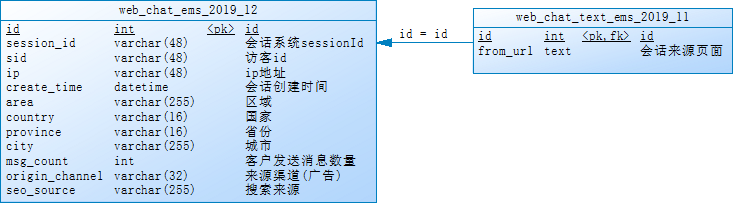
理解增量Sqoop导出和全量的区别

能够使用shell脚本删除mysql数据

## 主题需求

客户访问和咨询主题，顾名思义，分析的数据主要是客户的访问数据和咨询数据。但是经过需求调研，这里的访问数据，实际指的是**访问的客户量**，而不是客户访问量。原始数据来源于咨询系统的mysql业务数据库。

用户关注的核心指标有：1、总访问客户量、2、地区独立访客热力图、3、访客咨询率趋势、4、客户访问量和访客咨询率双轴趋势、5、时间段访问客户量趋势、6、来源渠道访问量占比、7、搜索来源访问量占比、8、活跃页面排行榜。



### 总访问客户量

说明：统计指定时间段内，访问客户的总数量。能够下钻到小时数据。

展现：线状图

指标：访问客户量

维度：年、季度、月

粒度：天

条件：年、季度、月

数据来源：咨询系统的web\_chat\_ems\_2019\_12等月表

按年：显示指定年范围内每天的客户访问量

按季度：显示指定季度范围内每天的客户访问量

按月：显示指定月份范围内每天的客户访问量

SQL：

1. **SELECT**
2. count( **DISTINCT** ( wce.sid ) ) '总数', DATE\_FORMAT( wce.create\_time, '%Y-%m-%d' ) '时间'
3. **FROM**
4. web\_chat\_ems\_2019\_12 wce
5. **GROUP** **BY**
6. DATE\_FORMAT( wce.create\_time, '%Y-%m-%d' )
7. **ORDER** **BY**
8. DATE\_FORMAT( wce.create\_time, '%Y-%m-%d' ) **ASC**

### 地区独立访客热力图

说明：统计指定时间段内，访问客户中各区域人数热力图。能够下钻到小时数据。

展现：地图热力图

指标：按照地区聚合访问的客户数量

维度：年、季度、月

粒度：天

条件：年、季度、月

数据来源：咨询系统的web\_chat\_ems\_2019\_12等月表

按年：显示指定年范围内每天的客户访问量

按季度：显示指定季度范围内每天的客户访问量

按月：显示指定月份范围内每天的客户访问量

SQL：

1. **SELECT**
2. wce.area '区域',
3. count(**DISTINCT**(wce.sid)) '总数',
4. wce.country '国家',
5. wce.province '省份',
6. wce.city '城市',
7. DATE\_FORMAT(wce.create\_time,'%Y-%m-%d') '时间'
8. **FROM**
9. web\_chat\_ems\_2019\_12 wce
10. **GROUP** **BY**
11. DATE\_FORMAT(wce.create\_time,'%Y-%m-%d'),
12. wce.area
13. **ORDER** **BY**
14. DATE\_FORMAT(wce.create\_time,'%Y-%m-%d') **ASC**, count(**DISTINCT**(wce.sid)) **DESC**;

### 访客咨询率趋势

说明：统计指定时间段内，不同地区（省、市）访问的客户中发起咨询的人数占比；

咨询率=发起咨询的人数/访问客户量；客户与网咨有说一句话的称为有效咨询。

展现：线状图

指标：访客咨询率

维度：年、月、城市

粒度：天

条件：年、季度、月、省、市

数据来源：咨询系统的web\_chat\_ems\_2019\_12等月表

按年：显示指定年范围内每天的客户访问量

按季度：显示指定季度范围内每天的客户访问量

按月：显示指定月份范围内每天的客户访问量

SQL:

1. **SELECT**
2. CONCAT(msgNumber.num / totalNumber.num \* 100, '%')
3. **FROM**
4. (
5. **SELECT**
6. count( **DISTINCT** ( sid ) ) num
7. **FROM**
8. web\_chat\_ems\_2019\_12
9. **WHERE**
10. msg\_count >= 1
11. ) msgNumber,
12. (
13. **SELECT**
14. count( **DISTINCT** ( sid ) ) num
15. **FROM**
16. web\_chat\_ems\_2019\_12
17. ) totalNumber

### 客户访问量和访客咨询率双轴趋势

说明：统计指定时间段内，每日客户访问量/咨询率双轴趋势图。能够下钻到小时数据。

每日客户访问量可以复用指标2数据；

咨询率可以复用指标3的数据。

按年：显示指定年范围内每天的客户访问量

按季度：显示指定季度范围内每天的客户访问量

按月：显示指定月份范围内每天的客户访问量

### 时间段访问客户量趋势

说明：统计指定时间段内，1-24h之间，每个时间段的访问客户量。

横轴：1-24h，间隔为一小时，纵轴：指定时间段内同一小时内的总访问客户量。

展现：线状图、柱状图、饼状图

指标：某小时的总访问客户量

维度：天

粒度：区间内小时段

条件：天

数据来源：咨询系统的web\_chat\_ems\_2019\_12等月表

SQL：

1. **SELECT**
2. DATE\_FORMAT(wce.create\_time,'%H') '时间',
3. count(**DISTINCT**(wce.sid)) '总数'
4. **FROM**
5. web\_chat\_ems\_2019\_12 wce
6. **GROUP** **BY**
7. DATE\_FORMAT(wce.create\_time,'%H')
8. **ORDER** **BY**
9. DATE\_FORMAT(wce.create\_time,'%H');

### 来源渠道访问量占比

说明：统计指定时间段内，不同来源渠道的访问客户量占比。能够下钻到小时数据。

展现：饼状图

指标：比值

维度：年、季度、月

粒度：天

条件：年、季度、月

数据来源：咨询系统的web\_chat\_ems\_2019\_12等月表

按年：显示指定年范围内每天的客户访问量

按季度：显示指定季度范围内每天的客户访问量

按月：显示指定月份范围内每天的客户访问量

SQL：

1. **SELECT**
2. count( **DISTINCT** ( wce.sid ) ) '总数',
3. wce.origin\_channel
4. **FROM**
5. web\_chat\_ems\_2019\_12 wce
6. **GROUP** **BY**
7. wce.origin\_channel;

### 搜索来源访问量占比

说明：统计指定时间段内，不同搜索来源的访问客户量占比。能够下钻到小时数据。

展现：饼状图

指标：比值

维度：年、季度、月

粒度：天

条件：年、季度、月

数据来源：咨询系统的web\_chat\_ems\_2019\_12等月表

按年：显示指定年范围内每天的客户访问量

按季度：显示指定季度范围内每天的客户访问量

按月：显示指定月份范围内每天的客户访问量

SQL：

1. **SELECT**
2. count( **DISTINCT** ( wce.sid ) ) '总数',
3. wce.seo\_source
4. **FROM**
5. web\_chat\_ems\_2019\_12 wce
6. **GROUP** **BY**
7. wce.seo\_source;

### 活跃页面排行榜

说明：统计指定时间段内，产生访问客户量最多的页面排行榜TOPN。能够下钻到小时数据。

展现：柱状图

指标：访问客户量

维度：页面、年、季度、月

粒度：天

条件：年、季度、月、Top数量

数据来源：咨询系统的 web\_chat\_text\_ems\_2019\_11等月表

按年：显示指定年范围内每天的客户访问量

按季度：显示指定季度范围内每天的客户访问量

按月：显示指定月份范围内每天的客户访问量

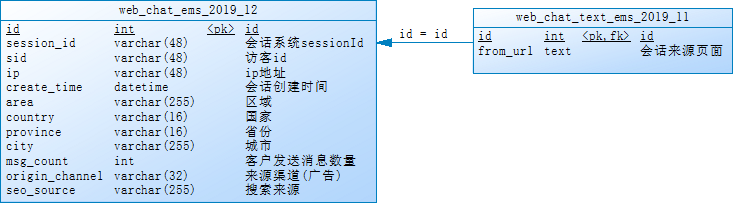
SQL：

1. **SELECT**
2. count( 1 ),
3. wcte.from\_url
4. **FROM**
5. web\_chat\_text\_ems\_2019\_11 wcte
6. **GROUP** **BY**
7. wcte.from\_url
8. LIMIT 20

### 原始数据结构

访问客户量的数据来源于咨询系统的访问会话月表web\_chat\_ems，表名的格式为web\_chat\_ems\_年\_月，年份为4位数字，月份为二位数字，如果为单数时，前面会用0来补全，比如web\_chat\_ems\_2019\_07。

web\_chat\_text\_ems表是访问附属月表，表名的格式和web\_chat\_ems相同。web\_chat\_ems和web\_chat\_text\_ems是一一对应的，通过主键id进行关联。



**建库：**

|  |
| --- |
| create database nev default character set utf8mb4 collate utf8mb4\_unicode\_ci; |

**web\_chat\_ems表结构：**

|  |
| --- |
| create table web\_chat\_ems\_2019\_07  (  id int auto\_increment comment '主键' primary key,  create\_date\_time timestamp null comment '数据创建时间',  session\_id varchar(48) default '' not null comment '会话系统sessionId',  sid varchar(48) collate utf8\_bin default '' not null comment '访客id',  create\_time datetime null comment '会话创建时间',  seo\_source varchar(255) collate utf8\_bin default '' null comment '搜索来源',  seo\_keywords varchar(512) collate utf8\_bin default '' null comment '关键字',  ip varchar(48) collate utf8\_bin default '' null comment 'IP地址',  area varchar(255) collate utf8\_bin default '' null comment '地域',  country varchar(16) collate utf8\_bin default '' null comment '所在国家',  province varchar(16) collate utf8\_bin default '' null comment '省',  city varchar(255) collate utf8\_bin default '' null comment '城市',  origin\_channel varchar(32) collate utf8\_bin default '' null comment '来源渠道(广告)',  user varchar(255) collate utf8\_bin default '' null comment '所属坐席',  manual\_time datetime null comment '人工开始时间',  begin\_time datetime null comment '坐席领取时间 ',  end\_time datetime null comment '会话结束时间',  last\_customer\_msg\_time\_stamp datetime null comment '客户最后一条消息的时间',  last\_agent\_msg\_time\_stamp datetime null comment '坐席最后一下回复的时间',  reply\_msg\_count int(12) default 0 null comment '客服回复消息数',  msg\_count int(12) default 0 null comment '客户发送消息数',  browser\_name varchar(255) collate utf8\_bin default '' null comment '浏览器名称',  os\_info varchar(255) collate utf8\_bin default '' null comment '系统名称'  ); |

**web\_chat\_text\_ems表结构：**

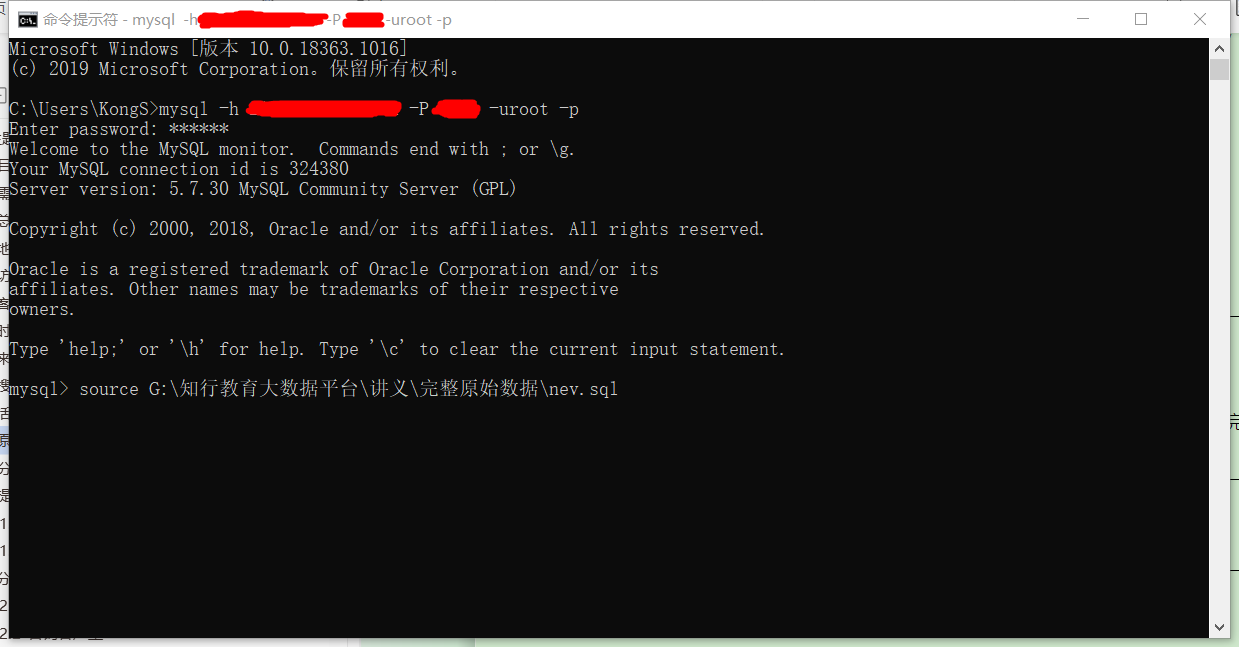
|  |
| --- |
| create table web\_chat\_text\_ems\_2019\_07  (  id int not null comment '主键' primary key,  referrer text collate utf8\_bin null comment '上级来源页面',  from\_url text collate utf8\_bin null comment '会话来源页面',  landing\_page\_url text collate utf8\_bin null comment '访客着陆页面',  url\_title text collate utf8\_bin null comment '咨询页面title',  platform\_description text collate utf8\_bin null comment '客户平台信息',  other\_params text collate utf8\_bin null comment '扩展字段中数据',  history text collate utf8\_bin null comment '历史访问记录'  ); |

**测试数据**

Mysql测试数据可以通过导入已准备好的sql文件进行创建：【Home\讲义\完整原始数据\nev.sql】。可以通过mysql脚本导入：

可以上传到linux后执行，也可以在windows执行（需要有mysql环境变量C:\Program Files\MySQL\MySQL Server 5.7\bin）。

|  |
| --- |
| mysql -h 192.168.52.150 -P 3306 -uroot -p  source G:\知行教育大数据平台\讲义\完整原始数据\nev.sql |



## 建模分析

### 提取指标维度

根据主题的需求，我们可以看出，包含的指标有一些是可以提取合并的：

1. 地区独立访客热力图、总访问客户量、时间段访问客户量趋势、来源渠道访问量占比、搜索来源访问量占比、活跃页面排行榜的指标都可以合并为一个：访问客户量。
2. 合并后的访问客户量指标，维度不同，而且数据来源也不同。
3. 访客咨询率趋势、客户访问量和访客咨询率双轴趋势，都包含了访客咨询率指标。
4. 访客咨询率=发起咨询的人数/访问客户量，分母访问客户量，可以复用前面的指标。因此只需要计算出分子：咨询客户量。

由此我们可以推断出，指标有两个：**访问客户量和咨询客户量**。

#### 访问客户量

虽然访问客户量的时间维度只有年、季度、月，但是展示粒度要具体到天，所以统计时也要包含日维度。同时要求能够下钻到小时数据，所以维度中也要包含小时。

来源渠道访客量占比，虽然最终要的是占比的比值，但是这个比值是可以通过具体的访客量计算出来，所以我们只需要提供不同来源渠道的访客量数据，柱状图前端就能够自动的计算比值。因此这里的指标也归类于访问客户量，维度为来源渠道。

活跃页面排行榜，字面看是和页面相关的指标，实际统计的却是每个页面的访问客户量，然后再进行排序后得出的排行榜。所以这里的指标也归类于访问客户量，统计的维度是具体的页面。但是要注意这里的数据来源有变化。

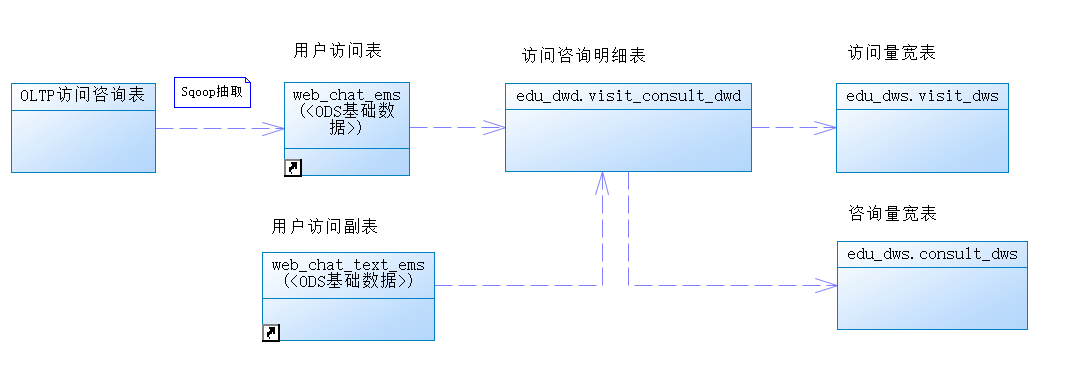
**维度包括：年、季度、月、天、小时（天区间内小时段）、地区、来源渠道、搜索来源、会话来源页面、总访问量。**

#### 咨询客户量

访客咨询率趋势统计中，访客咨询率=发起咨询的人数/访问客户量，分母访问客户量我们可以复制上面的指标数据，而分子咨询客户量则是我们要统计的新指标。

**维度包括：年、季度、月、日、地区、来源渠道。**

### 分层设计



#### 访问客户量

抽取咨询原始数据到ODS层以后，如何分析出对应维度的数据呢？

我们可以采取结果导向的方式来进行倒推：

1. 最终的数据维度：年、季度、月、日、小时、天区间内小时、地区、来源渠道、搜索来源、会话来源页面、总访问量；
2. 首先要有DWD层对ODS原始数据进行清洗和转换，作为我们的明细数据；
3. 维度我们可以分为两类：时间维度（年、季度、月、日、小时）和产品属性维度（地区、来源渠道、搜索来源、会话来源页面、总访问量）；
4. 我们可以将产品属性维度和最小的时间粒度（小时）来统计，作为共享表，放置在DWM层；
5. 在DWM层小时数据的基础上，进行上卷sum统计（年、季度、月、日、小时），即可得到DWS层的数据集市；
6. 注意：最终的数据要求在统计之前，要先根据客户进行去重，这也对我们的中间层进行了限制，不能简单的先按天去重count，然后再按月和年sum，因为不同天的客户可能存在重复，直接sum会导致结果不正确；
7. 所以DWM层不能进行count，那么如果将DWD客户去重后的数据，直接保存为中间表呢？也会存在问题，因为如果DWM层把全表的用户去重后，在DWS层数据会存在丢失减少的情况，比如小时数据和天数据。
8. DWS层直接根据DWD的数据进行统计，得出数据集市；
9. 将宽表数据导出到mysql，由FineBI灵活选择APP数据字段进行展示。
10. ODS——》DWD——》DWS。

#### 咨询客户量

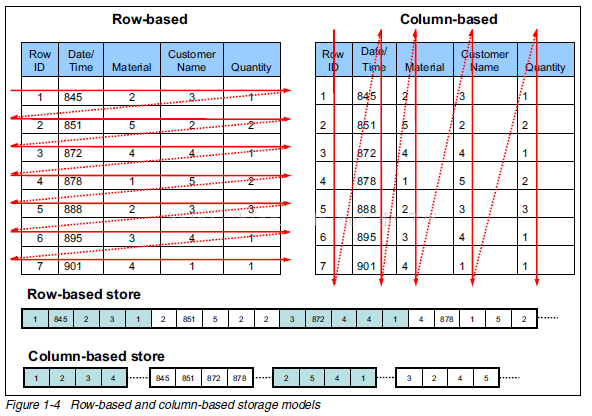
1. 最终的数据维度：年、季度、月、日、地区、来源渠道；
2. 统计的数据和访问客户量指标相似，唯一的不同点是，多了一个条件：和客服有聊天信息；
3. 因为咨询客户量的数据来源和访问客户量相同，所以ODS层可以复用；
4. DWD层对ODS原始数据进行清洗和转换，可以复用；
5. DWM层先去重，再在DWS中sum的结果是不正确的；所以跳过DWM层；
6. DWS层直接在DWD层的基础上，加上聊天信息的条件后，按照维度进行统计。

## 访问客户量实现

### 数据格式和压缩格式

#### 数据格式

##### 列式存储和行式存储



**行存储的特点：** 查询满足条件的一整行（所有列）数据的时候，列存储则需要去每个聚集的字段找到对应的每个列的值，行存储只需要找到其中一个值，其余的值都在相邻地方，所以此时行存储查询的速度更快。

**列存储的特点：** 因为每个字段的数据聚集存储，在查询只需要少数几个字段的时候，能大大减少读取的数据量；每个字段的数据类型一定是相同的，列式存储可以针对性的设计更好的设计压缩算法。

##### TEXTFILE

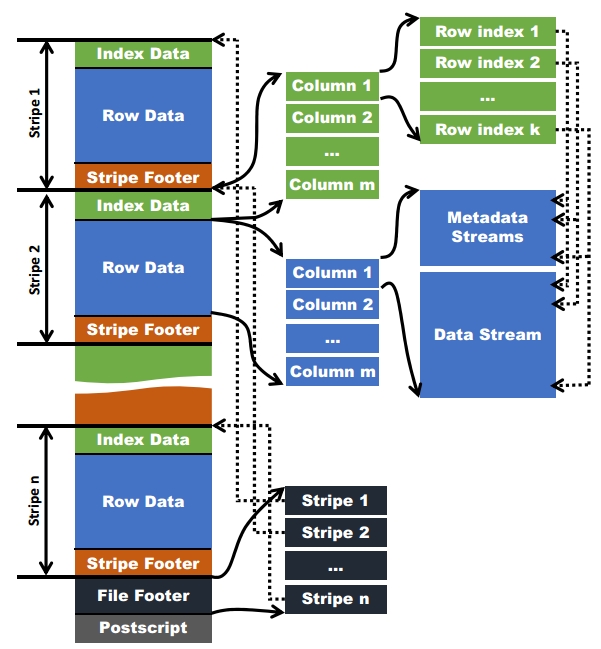
默认格式，行式存储。可结合Gzip、Bzip2使用(系统自动检查，执行查询时自动解压)，但使用这种方式，hive不会对数据进行切分，从而无法对数据进行并行操作。并且反序列化过程中，必须逐个字符判断是不是分隔符和行结束符，性能较差。

##### ORCFILE

使用ORC文件格式可以提高hive读、写和处理数据的能力。ORCFile是RCFile的升级版。

在ORC格式的hive表中，数据按行分块，每块按列存储。结合了行存储和列存储的优点。记录首先会被横向的切分为多个stripes，然后在每一个stripe内数据以列为单位进行存储，所有列的内容都保存在同一个文件中。

每个stripe的默认大小为256MB，相对于RCFile每个4MB的stripe而言，更大的stripe使ORC可以支持索引，数据读取更加高效。



#### 存储和压缩结合

##### zlib压缩

优点：压缩率比较高；hadoop本身支持，在应用中处理gzip格式的文件就和直接处理文本一样。

缺点：压缩性能一般。

##### snappy压缩

优点：高速压缩速度和合理的压缩率。

缺点：压缩率比zlib要低；hadoop本身不支持，需要安装（CDH版本已自动支持，可忽略）。

#### 系统采用的格式

因为ORCFILE的压缩快、存取快，而且拥有特有的查询优化机制，所以系统采用ORCFILE存储格式（RCFILE升级版），压缩算法采用orc支持的ZLIB和SNAPPY。

在ODS数据源层，因为数据量较大，可以采用orcfile+ZLIB的方式，以节省磁盘空间；

而在计算的过程中（DWD、DWM、DWS、APP），为了不影响执行的速度，可以浪费一点磁盘空间，采用orcfile+SNAPPY的方式，提升hive的执行速度。

存储空间足够的情况下，推荐采用SNAPPY压缩。

### 全量和增量

开发步骤共包含两大过程：全量过程和增量过程。

##### 全量过程

全量过程是在首次建库时，需要对OLTP应用中的全量数据进行采集、清洗和统计计算。历史数据量可能会非常大，远远超出了增量过程。在执行时需要进行针对性的优化配置并采用分批执行。

##### 增量过程

增量过程是在全量过程之后进行的，大多采用的是T+1模式。

全量执行完毕后，对OLTP每天的新增数据和更新数据要进行同步，如果还是对全量数据进行分析，效率会非常低下。增量数据只有一天的量，采集、清洗和统计的效率相对于全量过程会有很大提升。

什么是T+1？

这种说法来源于股票交易：

T+0，是国际上普遍使用的一种证劵度（或期货）交易制度。凡在证劵（或期货）成交日当天办理好证劵（或期货）和价款清算交割手续的交易制度，就称为T+0交易。通俗说，就是当天买入的证道劵（或期货）在当天就可以卖出。

T+1是一种股票交易制度，即当日买进的股票，要到下一个交易日才能卖出。“T”指交易登记日，“T+1”指登记日的次日。

### Hive的分区

我们知道传统的OLTP数据库一般都具有索引和表分区的功能，通过表分区能够在特定的区域检索数据，减少扫描成本，在一定程度上提高查询效率，我们还可以通过建立索引进一步提升查询效率。在Hive数仓中也有索引和分区的概念。

为了对表进行合理的管理以及提高查询效率，Hive可以将表组织成“分区”。

分区是表的部分列的集合，可以为频繁使用的数据建立分区，这样查找分区中的数据时就不需要扫描全表，这对于提高查找效率很有帮助。

分区是一种根据“分区列”（partition column）的值对表进行粗略划分的机制。Hive中每个分区对应着表很多的子目录，将所有的数据按照分区列放入到不同的子目录中去。

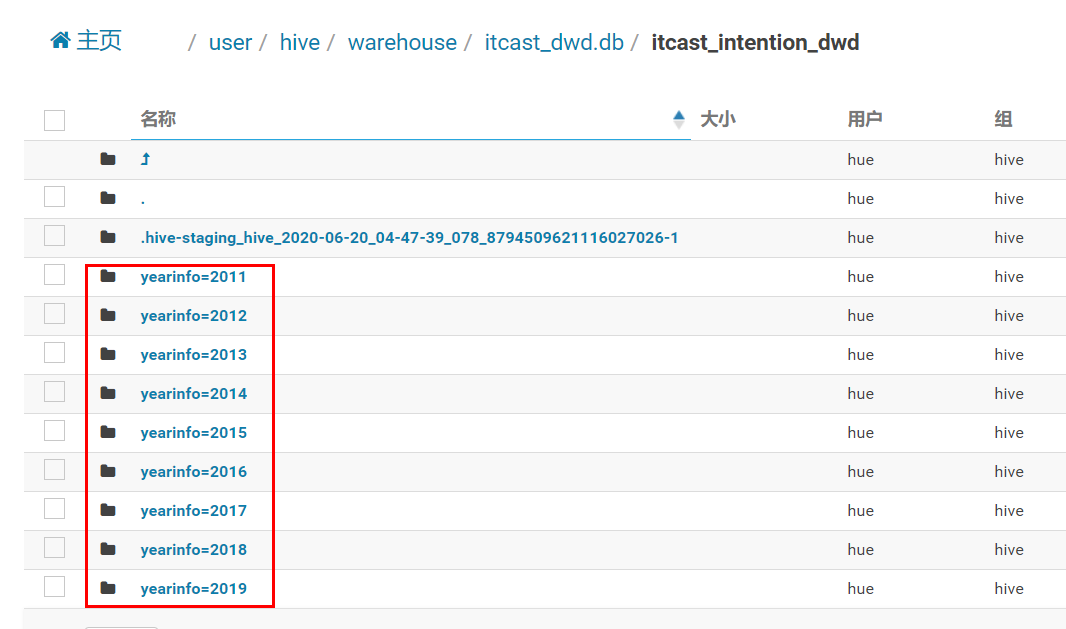
##### 为什么要分区

庞大的数据集可能需要耗费大量的时间去处理。在许多场景下，可以通过分区的方法减少每一次扫描总数据量，这种做法可以显著地改善性能。

数据会依照单个或多个列进行分区，通常按照时间、地域或者是商业维度进行分区。

比如电影表，分区的依据可以是电影的种类和评级，另外，按照拍摄时间划分可能会得到均匀的结果。

为了达到性能表现的一致性，对不同列的划分应该让数据尽可能均匀分布。最好的情况下，分区的划分条件总是能够对应where语句的部分查询条件，这样才能充分利用分区带来的性能优势。



Hive的分区使用HDFS的子目录功能实现。每一个子目录包含了分区对应的列名和每一列的值。但是由于HDFS并不支持大量的子目录，这也给分区的使用带来了限制。我们有必要对表中的分区数量进行预估，从而避免因为分区数量过大带来一系列问题。

Hive查询通常使用分区的列作为查询条件。这样的做法可以指定MapReduce任务在HDFS中指定的子目录下完成扫描的工作。HDFS的文件目录结构可以像索引一样高效利用。

Hive(Inceptor)分区包括静态分区和动态分区。

##### 静态分区

根据插入时是否需要手动指定分区可以分为：**静态分区**：导入数据时需要手动指定分区。**动态分区**：导入数据时，系统可以动态判断目标分区。

1. **创建静态分区**

直接在 PARTITIONED BY 后面跟上分区键、类型即可。（分区键不能和任何列重名）

**语法：**

|  |
| --- |
| CREATE [EXTERNAL] TABLE <table\_name>  (<col\_name> <data\_type> [, <col\_name> <data\_type> ...])  -- 指定分区键和数据类型  PARTITIONED BY (<partition\_key> <data\_type>, ...)  [ROW FORMAT <row\_format>]  [STORED AS TEXTFILE|ORC|CSVFILE]  [LOCATION '<file\_path>']  [TBLPROPERTIES ('<property\_name>'='<property\_value>', ...)]; |

**栗子：**

|  |
| --- |
| --分区字段主要是时间，按年分区  CREATE TABLE device\_open (  deviceid varchar(50),  ...  )  **PARTITIONED BY (year varchar(50))**  ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'; |

1. **写入数据**

**语法：**

|  |
| --- |
| -- 覆盖写入  INSERT OVERWRITE TABLE <table\_name>  PARTITION (<partition\_key>**=**<partition\_value>[, <partition\_key>=<partition\_value>, ...]**)**  SELECT <select\_statement>;    -- 追加写入  INSERT INTO TABLE <table\_name>  PARTITION (<partition\_key>=<partition\_value>[, <partition\_key>=<partition\_value>, ...]**)**  SELECT <select\_statement>; |

**栗子：**

|  |
| --- |
| insert overwrite table device\_open partition(year=’2020’)  select  ...,  original\_device\_open.month as month,  original\_device\_open.day as day,  original\_device\_open.hour as hour  FROM original\_device\_open |

##### 动态分区

1. **创建**

创建方式与静态分区表完全一样。

**语法：**

|  |
| --- |
| --分区字段主要是时间，分为年，月，日，时  CREATE TABLE device\_open (  deviceid varchar(50),  ...  )  PARTITIONED BY (year varchar(50), month varchar(50), day varchar(50), hour varchar(50))  ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'; |

1. **写入**

动态分区只需要给出分区键名称。

语法：

|  |
| --- |
| -- 开启动态分区支持，并开启非严格模式  set hive.exec.dynamic.partition=true;  set hive.exec.dynamic.partition.mode=nonstrict;  insert overwrite table device\_open partition(year,month,day,hour)  select  ...,  original\_device\_open.year as year,  original\_device\_open.month as month,  original\_device\_open.day as day,  original\_device\_open.hour as hour  FROM original\_device\_open |

set hive.exec.dynamic.partition=true; 是开启动态分区

set hive.exec.dynamic.partition.mode=nonstrict; 这个属性默认值是strict，就是要求分区字段必须有一个是静态的分区值。全部动态分区插入，需要设置为nonstrict非严格模式。

代码中标红的部分，partition(year,month,day,hour) 就是要动态插入的分区。对于大批量数据的插入分区，动态分区相当方便。

##### 静态分区和动态分区混用

一张表可同时被静态和动态分区键分区，只是动态分区键需要放在静态分区键的后面（因为HDFS上的动态分区目录下不能包含静态分区的子目录）。

静态分区键要用 <spk>=<value> 指定分区值；动态分区只需要给出分区键名称 <dpk>。

spk 即静态分区static partition key， dpk 即动态分区dynamic partition key。

比如：

|  |
| --- |
| insert overwrite table device\_open partition(year='2017',month='05',day,hour)  select  ...,  original\_device\_open.day as day,  original\_device\_open.hour as hour  FROM original\_device\_open  where original\_device\_open.year='2017' and original\_device\_open.month='05' |

partition(year='2017', month='05', day, hour)，year和month是静态分区字段，day和hour是动态分区字段，这里指将2017年5月份的数据插入分区表，对应底层的物理操作就是将2017年5月份的数据load到hdfs上对应2017年5月份下的所有day和hour目录中去。

注意混用的情况下，静态分区的上层必须也是静态分区，如果partition(year, month, day=’05’, hour=’08’)，则会报错：FAILED: SemanticException [Error 10094]: Line 1:50 Dynamic partition cannot be the parent of a static partition ''day''。

##### 有序动态分区

注意，如果个人电脑性能不好，出现因为动态分区而导致的内存溢出问题，可以设置hive.optimize.sort.dynamic.partition进行避免：



设置为true后，当启用动态分区时，reducer仅随时保持一个记录写入程序，从而降低对 reducer产生的内存压力。但同时也会使查询性能变慢。

动态分区其他相关属性设置：



### 建模

#### 指标和维度

指标：访问客户量是单位时间内访问网站的去重后客户数量，以天为单位显示访问客户。

维度：

* 时间维度：年、季度、月、天、小时
* 业务属性维度：地区、来源渠道、搜索来源、会话来源页面、总访问量。

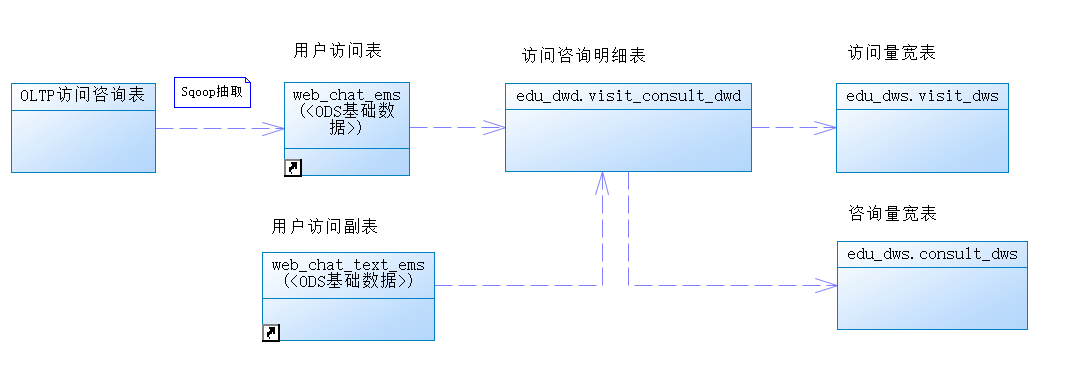
#### 事实表和维度表

事实表的数据就是指标数据，访问客户量指标的事实表就是我们的客户访问表。而维度数据都包含在事实表中，没有需要额外关联的维度表。

#### 分层

数据库命名统一加上前缀itcast，在实际场景中，将此前缀替换为系统的简称即可。比如：edu\_ods、edu\_dwd、edu\_dws等。

ODS层是原始数据，一般不允许修改，所以使用外部表保证数据的安全性，避免误删除；DW和APP层是统计数据，为了使覆盖插入等操作更方便，满足业务需求的同时，提高开发和测试效率，推荐使用内部表。



##### ODS层

从咨询系统OLTP数据库的web\_chat\_ems\_20XX\_XX等月表中抽取的原始数据；

离线数仓大多数的场景都是T+1，为了便于后续的DW层清洗数据时，快速获取昨天的数据，ODS模型要在原始mysql表的基础之上增加starts\_time抽取日期字段，并且可以使用starts\_time字段分区以提升查询的性能。

###### 建库

|  |
| --- |
| CREATE DATABASE IF NOT EXISTS `itcast\_ods`; |

###### 建表web\_chat\_ems

建表时，要注意字段名不要采用关键字，比如原始mysql表中有一个user字段，我们需要将它修改为user\_match。

**注意**，设置ORC压缩格式前一定要先设置hive.exec.orc.compression.strategy，否则压缩不生效：

|  |
| --- |
| *--写入时压缩生效* set hive.exec.orc.compression.strategy=COMPRESSION; |

|  |
| --- |
| CREATE EXTERNAL TABLE IF NOT EXISTS itcast\_ods.web\_chat\_ems (  id INT comment '主键',  create\_date\_time STRING comment '数据创建时间',  session\_id STRING comment '七陌sessionId',  sid STRING comment '访客id',  create\_time STRING comment '会话创建时间',  seo\_source STRING comment '搜索来源',  seo\_keywords STRING comment '关键字',  ip STRING comment 'IP地址',  area STRING comment '地域',  country STRING comment '所在国家',  province STRING comment '省',  city STRING comment '城市',  origin\_channel STRING comment '投放渠道',  **user\_match** STRING comment '所属坐席',  manual\_time STRING comment '人工开始时间',  begin\_time STRING comment '坐席领取时间 ',  end\_time STRING comment '会话结束时间',  last\_customer\_msg\_time\_stamp STRING comment '客户最后一条消息的时间',  last\_agent\_msg\_time\_stamp STRING comment '坐席最后一下回复的时间',  reply\_msg\_count INT comment '客服回复消息数',  msg\_count INT comment '客户发送消息数',  browser\_name STRING comment '浏览器名称',  os\_info STRING comment '系统名称') comment '访问会话信息表' **PARTITIONED BY(starts\_time STRING)** ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' stored as **orc** location '/user/hive/warehouse/itcast\_ods.db/web\_chat\_ems\_ods' TBLPROPERTIES ('orc.compress'='**ZLIB**'); |

###### 建表web\_chat\_text\_ems

|  |
| --- |
| CREATE EXTERNAL TABLE IF NOT EXISTS itcast\_ods.web\_chat\_text\_ems (  id INT COMMENT '主键来自MySQL',  referrer STRING comment '上级来源页面',  from\_url STRING comment '会话来源页面',  landing\_page\_url STRING comment '访客着陆页面',  url\_title STRING comment '咨询页面title',  platform\_description STRING comment '客户平台信息',  other\_params STRING comment '扩展字段中数据',  history STRING comment '历史访问记录' ) comment 'EMS-PV测试表' **PARTITIONED BY(start\_time STRING)** ROW FORMAT DELIMITED  FIELDS TERMINATED BY '\t' stored as **orc** location '/user/hive/warehouse/itcast\_ods.db/web\_chat\_text\_ems\_ods' TBLPROPERTIES ('orc.compress'=**'ZLIB'**); |

##### DWD层

维度：

* 时间维度：年、季度、月、天、小时
* 业务属性维度：地区、来源渠道、搜索来源、会话来源页面、总访问量。

建库：

|  |
| --- |
| CREATE DATABASE IF NOT EXISTS `itcast\_dwd` WITH DBPROPERTIES ( 'creator' = 'jiale', 'create\_date' = '2020-05-05'); |

将ODS层数据，进行清洗转换，并且将web\_chat\_ems主表和web\_chat\_text\_ems附表的内容根据id合并在一起。数据粒度保持不变。

数据清洗：空数据、不满足业务需求的数据处理。

数据转换：数据格式和数据形式的转换，比如时间类型可以转换为同样的展现形式“yyyy-MM-dd HH:mm:ss”或者时间戳类型，金钱类型的数据可以统一转换为以元为单位或以分为单位的数值。

|  |
| --- |
| create table if not exists itcast\_dwd.visit\_consult\_dwd(  session\_id STRING comment '七陌sessionId',  sid STRING comment '访客id',  create\_time bigint comment '会话创建时间',  seo\_source STRING comment '搜索来源',  ip STRING comment 'IP地址',  area STRING comment '地域',  msg\_count int comment '客户发送消息数',  origin\_channel STRING COMMENT '来源渠道',  referrer STRING comment '上级来源页面',  from\_url STRING comment '会话来源页面',  landing\_page\_url STRING comment '访客着陆页面',  url\_title STRING comment '咨询页面title',  platform\_description STRING comment '客户平台信息',  other\_params STRING comment '扩展字段中数据',  history STRING comment '历史访问记录',  hourinfo string comment '小时',  quarterinfo string comment '季度' ) comment '访问咨询DWD表' partitioned by(yearinfo String, monthinfo String, dayinfo string) row format delimited fields terminated by '\t' stored as orc location '/user/hive/warehouse/itcast\_dwd.db/visit\_consult\_dwd' tblproperties ('orc.compress'='SNAPPY'); |

##### DWS层

在DWD层的基础上，按照业务的要求进行统计分析；时间和业务属性三个维度分类，可以在模型中增加对应的属性标识：

* 时间维度：1.年、2.季度、3.月、4.天、5.小时
* 业务属性维度：1.地区、2.来源渠道、3.搜索来源、4.会话来源页面、5.总访问量

建库：

|  |
| --- |
| CREATE DATABASE IF NOT EXISTS `itcast\_dws` WITH DBPROPERTIES ( 'creator' = 'kongshuai', 'create\_date' = '2020-05-05'); |

|  |
| --- |
| CREATE TABLE IF NOT EXISTS itcast\_dws.visit\_dws (  sid\_total INT COMMENT '根据sid去重求count',  sessionid\_total INT COMMENT '根据sessionid去重求count',  ip\_total INT COMMENT '根据IP去重求count',  area STRING COMMENT '区域信息',  seo\_source STRING COMMENT '搜索来源',  origin\_channel STRING COMMENT '来源渠道',  hourinfo STRING COMMENT '创建时间，统计至小时',  quarterinfo STRING COMMENT '季度',  time\_str STRING COMMENT '时间明细',  from\_url STRING comment '会话来源页面',  groupType STRING COMMENT '产品属性类型：1.地区；2.搜索来源；3.来源渠道；4.会话来源页面；5.总访问量',  time\_type STRING COMMENT '时间聚合类型：1、按小时聚合；2、按天聚合；3、按月聚合；4、按季度聚合；5、按年聚合；') comment 'EMS访客日志dws表' PARTITIONED BY(yearinfo STRING,monthinfo STRING,dayinfo STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' stored as orc location '/user/hive/warehouse/itcast\_dws.db/visit\_dws' TBLPROPERTIES ('orc.compress'='SNAPPY'); |

##### APP层

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FineBI，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。

### Hive参数优化（基础）

此课程中关于Hive的优化，皆是基于Hive2.x的版本，对于Hive1.x旧版本的优化机制不再复述（新版本已改善或变更）。另外新版本中默认为开启状态的优化配置项，在工作中无需修改，也不再复述。

#### HDFS副本数

**dfs.replication（HDFS）**

文件副本数，通常设为3，不推荐修改。

如果测试环境只有二台虚拟机（2个datanode节点），此值要修改为2。



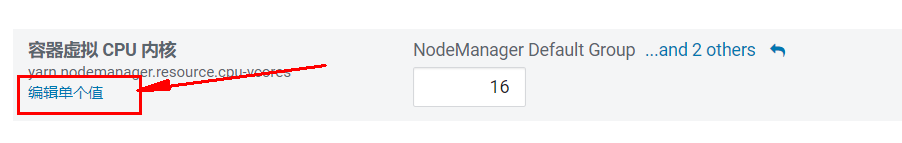
#### Yarn基础配置

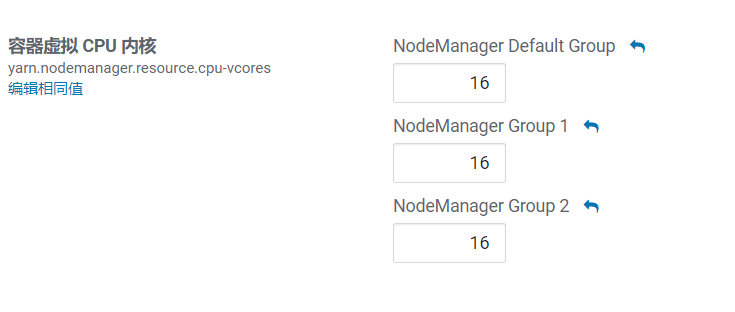
##### NodeManager配置

###### CPU配置

配置项：yarn.nodemanager.resource.cpu-vcores

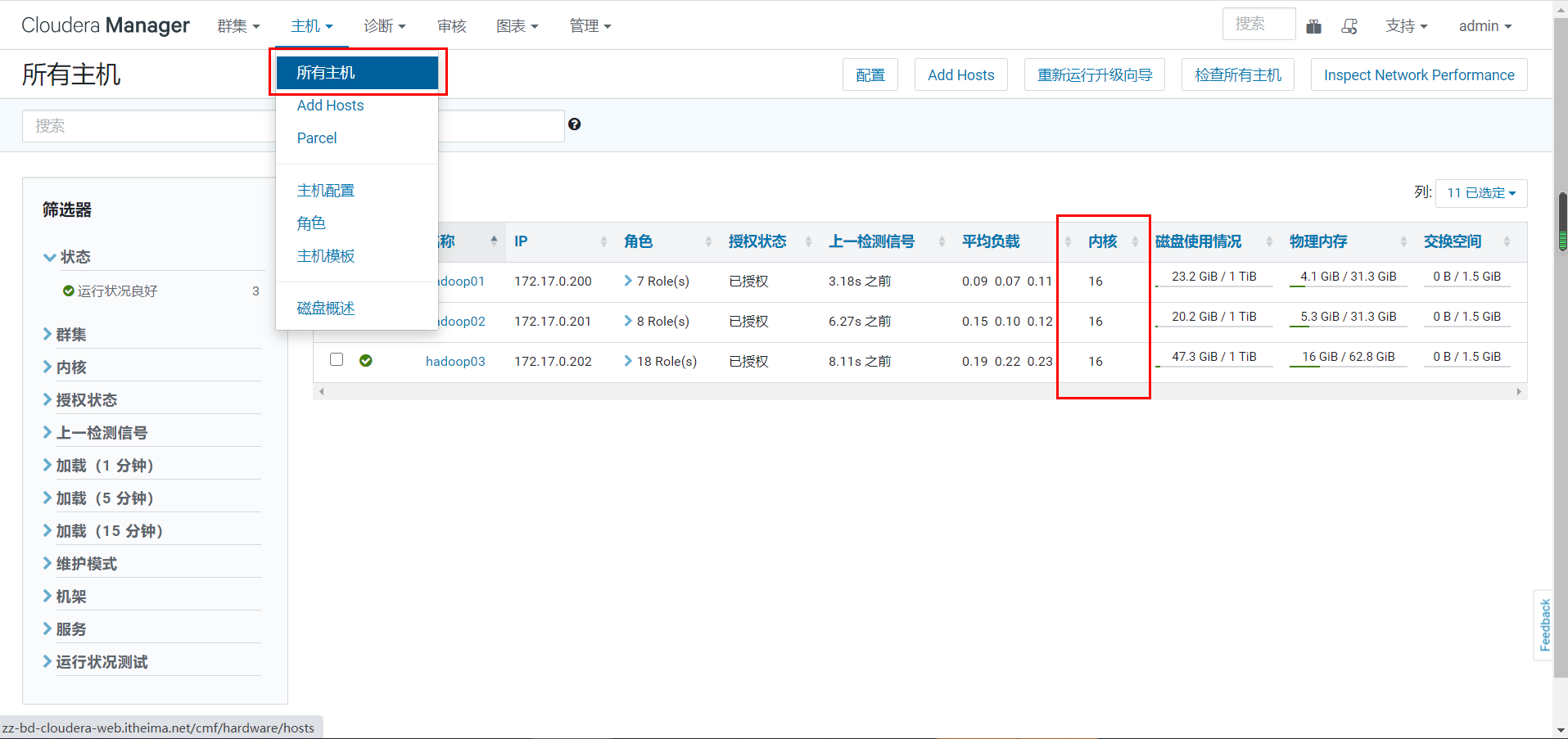
表示该节点服务器上yarn可以使用的虚拟CPU个数，默认值是8，推荐将值配置与物理CPU线程数相同，如果节点CPU核心不足8个，要调小这个值，yarn不会智能的去检测物理核心数。





查看 CPU 线程数：

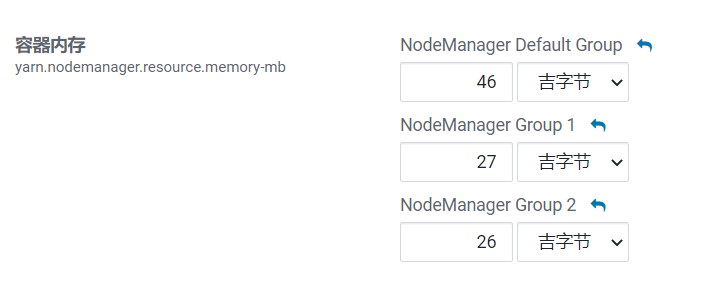
|  |
| --- |
| grep 'processor' /proc/cpuinfo | sort -u | wc -l |

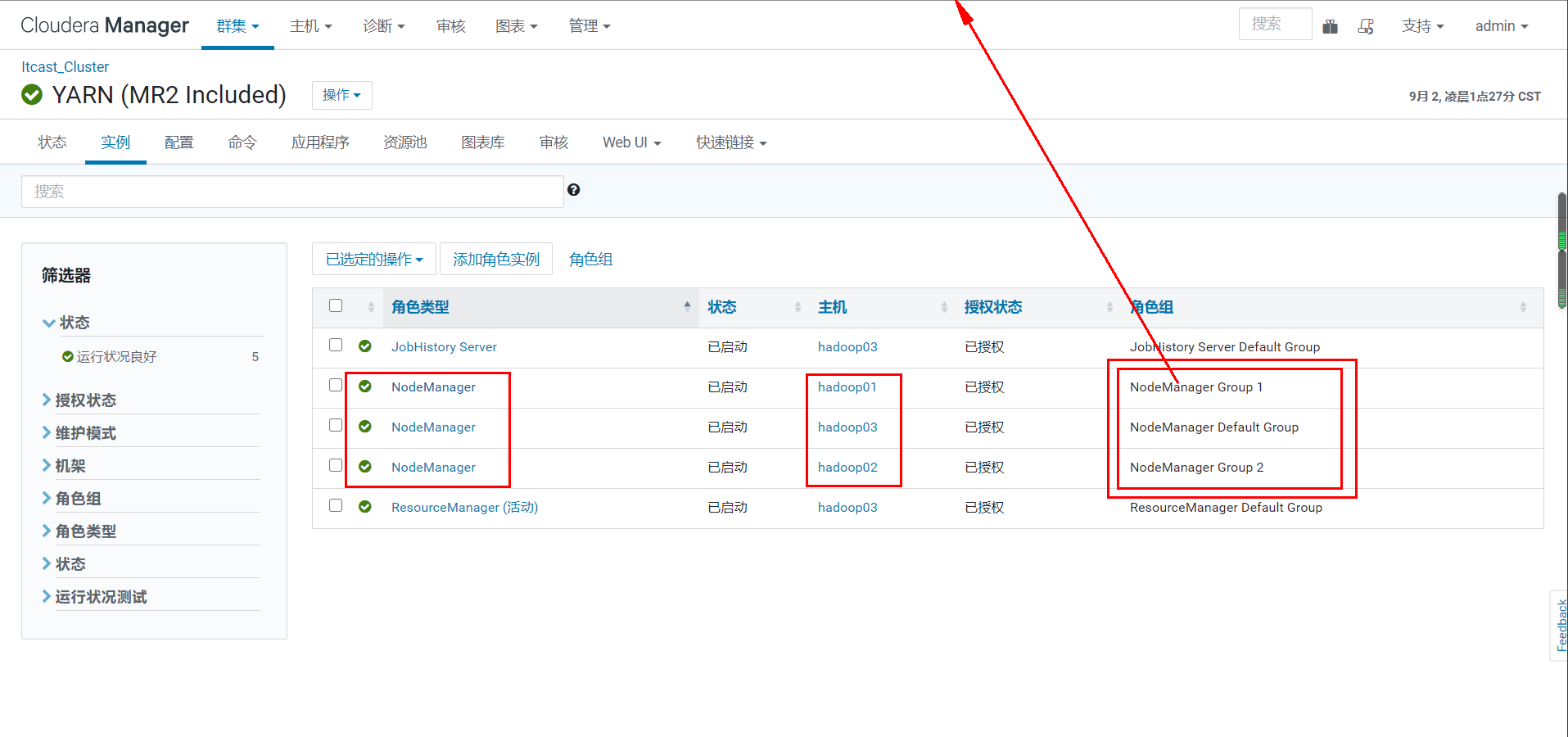


###### 内存配置

配置项：yarn.nodemanager.resource.memory-mb

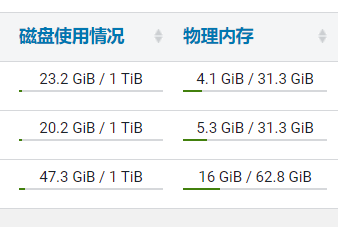
设置该nodemanager节点上可以为容器分配的总内存，默认为8G，如果节点内存资源不足8G，要减少这个值，yarn不会智能的去检测内存资源，一般按照服务器剩余可用内存资源进行配置。生产上根据经验一般要预留15-20%的内存，那么可用内存就是实际**内存\*0.8**，比如实际内存是64G，那么64\*0.8=51.2G，我们设置成50G就可以了（固定经验值）。





通过CM所有主机查看剩余内存：

可以看到第一台剩余内存为31.3-4.1=27.2G。



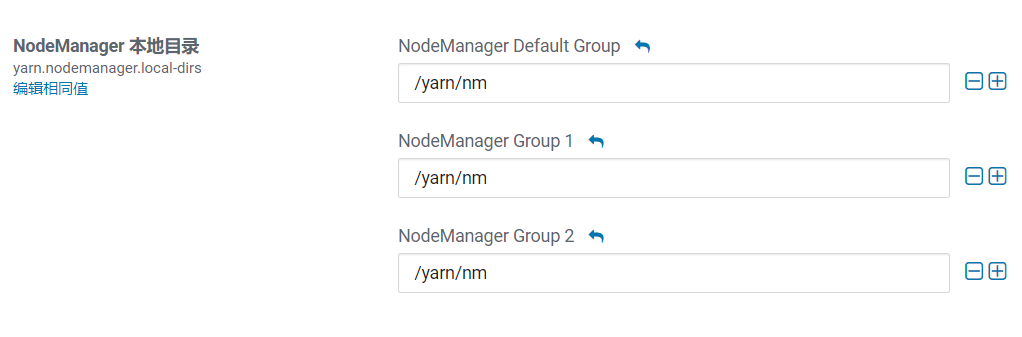
**注意**，要同时设置yarn.scheduler.maximum-allocation-mb为一样的值，yarn.app.mapreduce.am.command-opts（JVM内存）的值要同步修改为略小的值（-Xmx1024m）。

###### 本地目录

**yarn.nodemanager.local-dirs（Yarn）**

NodeManager 存储中间数据文件的本地文件系统中的目录列表。

如果单台服务器上有多个磁盘挂载，则配置的值应当是分布在各个磁盘上目录，这样可以充分利用节点的IO读写能力。



##### MapReduce内存配置

当MR内存溢出时，可以根据服务器配置进行调整。

mapreduce.map.memory.mb

为作业的每个 Map 任务分配的物理内存量(MiB)，默认为0，自动判断大小。

mapreduce.reduce.memory.mb

为作业的每个 Reduce 任务分配的物理内存量(MiB)，默认为0，自动判断大小。

mapreduce.map.java.opts、mapreduce.reduce.java.opts

Map和Reduce的JVM配置选项。

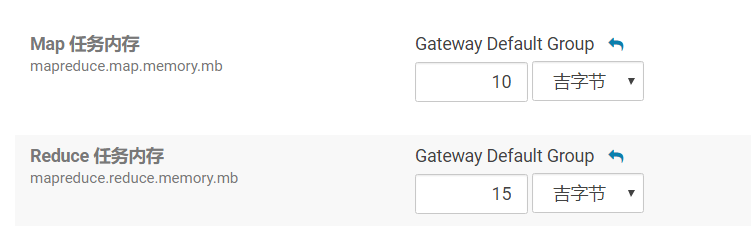
注意：

mapreduce.map.java.opts一定要小于mapreduce.map.memory.mb；

mapreduce.reduce.java.opts一定要小于mapreduce.reduce.memory.mb，格式**-Xmx4096m**。

注意：

此部分所有配置均不能大于Yarn的NodeManager内存配置。





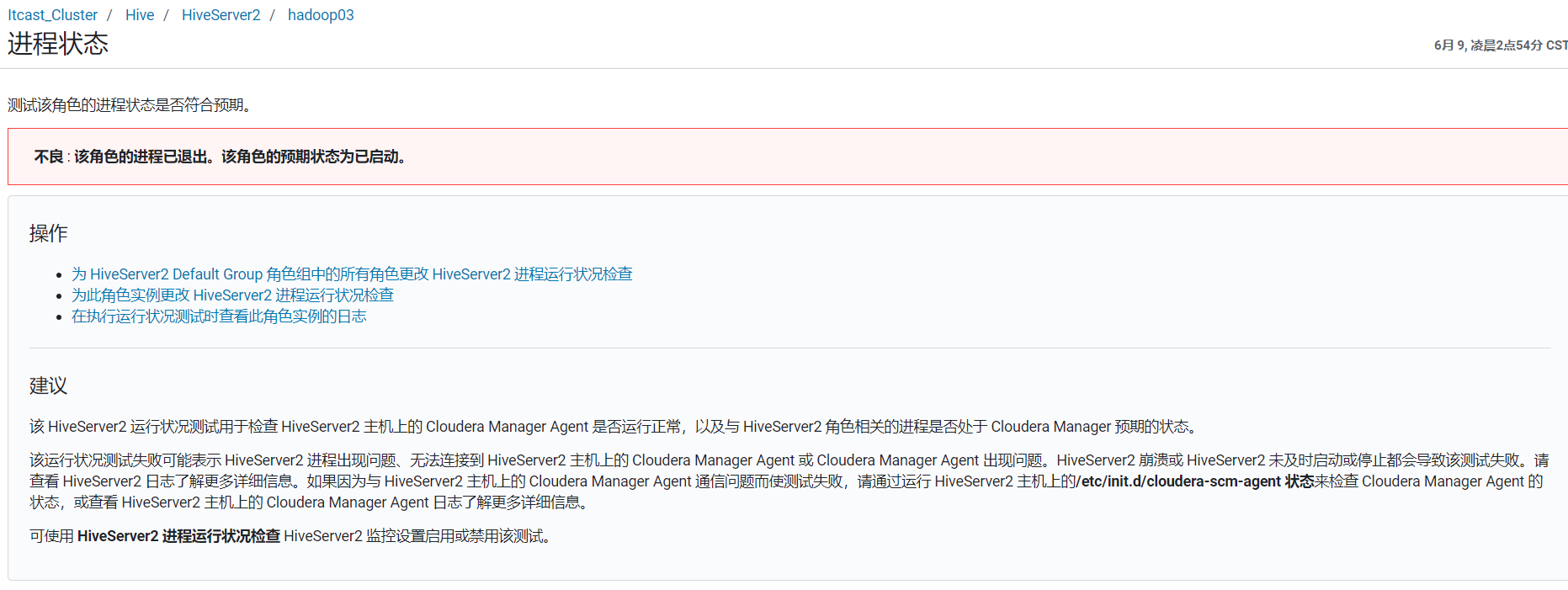


#### Hive基础配置

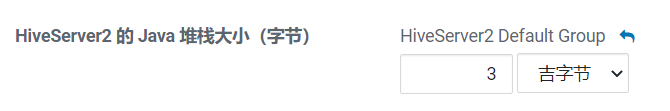
##### HiveServer2 的 Java 堆栈

Hiveserver2异常退出，导致连接失败的问题。





解决方法：修改HiveServer2 的 Java 堆栈大小。

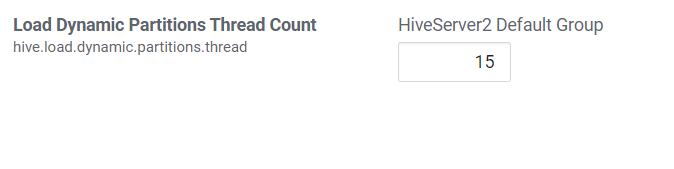


##### 动态生成分区的线程数

hive.load.dynamic.partitions.thread

用于加载动态生成的分区的线程数。加载需要将文件重命名为它的最终位置，并更新关于新分区的一些元数据。默认值为 15 。

当有大量动态生成的分区时，增加这个值可以提高性能。根据服务器配置修改。

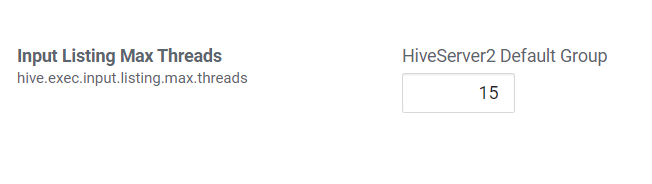


##### 监听输入文件线程数

hive.exec.input.listing.max.threads

Hive用来监听输入文件的最大线程数。默认值：15。

当需要读取大量分区时，增加这个值可以提高性能。根据服务器配置进行调整。



#### 压缩配置

##### Map输出压缩

除了创建表时指定保存数据时压缩，在查询分析过程中，Map的输出也可以进行压缩。由于map任务的输出需要写到磁盘并通过网络传输到reducer节点，所以通过使用LZO、LZ4或者Snappy这样的快速压缩方式，是可以获得性能提升的，因为需要传输的数据减少了。

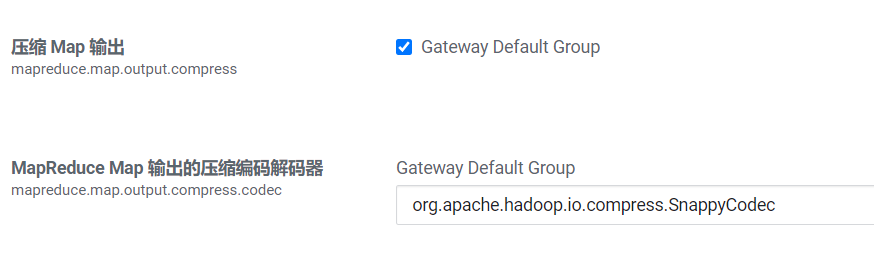
**MapReduce配置项：**

* mapreduce.map.output.compress

设置是否启动map输出压缩，默认为false。在需要减少网络传输的时候，可以设置为true。

* mapreduce.map.output.compress.codec

设置map输出压缩编码解码器，默认为org.apache.hadoop.io.compress.DefaultCodec，推荐使用SnappyCodec：org.apache.hadoop.io.compress.SnappyCodec。



##### Reduce结果压缩

是否对任务输出结果压缩，默认值false。对传输数据进行压缩，既可以减少文件的存储空间，又可以加快数据在网络不同节点之间的传输速度。

配置项：

1. mapreduce.output.fileoutputformat.compress

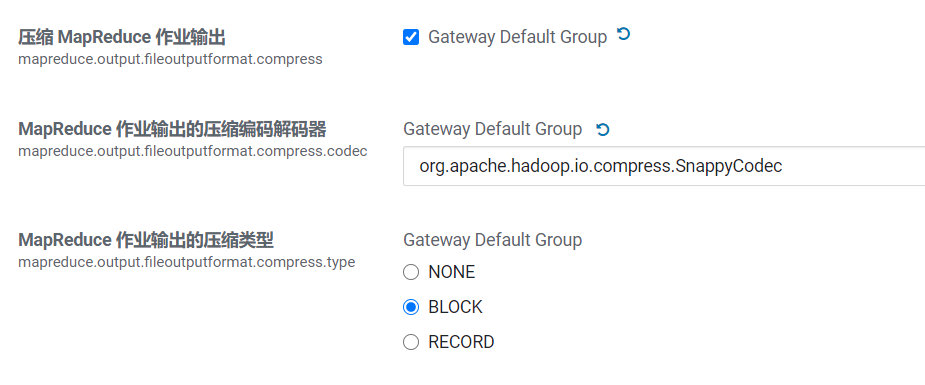
是否启用 MapReduce 作业输出压缩。

1. mapreduce.output.fileoutputformat.compress.codec

指定要使用的压缩编码解码器，推荐SnappyCodec。

1. mapreduce.output.fileoutputformat.compress.type

指定MapReduce作业输出的压缩方式，默认值RECORD，可配置值有：NONE、RECORD、BLOCK。推荐使用BLOCK，即针对一组记录进行批量压缩，压缩效率更高。



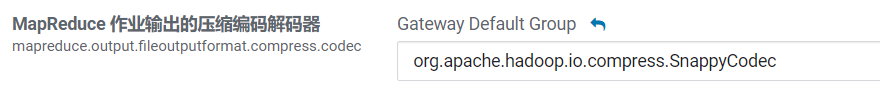
##### Hive执行过程通用压缩设置

主要包括压缩/解码器设置和压缩方式设置：

* mapreduce.output.fileoutputformat.compress.codec（Yarn）
  + map输出所用的压缩编码解码器，默认为org.apache.hadoop.io.compress.DefaultCodec；

推荐使用SnappyCodec：org.apache.hadoop.io.compress.SnappyCodec。

* mapreduce.output.fileoutputformat.compress.type
  + 输出产生任务数据的压缩方式，默认值RECORD，可配置值有：NONE、RECORD、BLOCK。推荐使用BLOCK，即针对一组记录进行批量压缩，压缩效率更高。





##### Hive多个Map-Reduce中间数据压缩

控制 Hive 在多个map-reduce作业之间生成的中间文件是否被压缩。压缩编解码器和其他选项由上面Hive通用压缩mapreduce.output.fileoutputformat.compress.\*确定。

|  |
| --- |
| set hive.exec.compress.intermediate=true; |

##### Hive最终结果压缩

控制是否压缩查询的最终输出(到 local/hdfs 文件或 Hive table)。压缩编解码器和其他选项由 上面Hive通用压缩mapreduce.output.fileoutputformat.compress.\*确定。

|  |
| --- |
| set hive.exec.compress.output=true; |

#### 其他

##### JVM重用（不再支持）

随着Hadoop版本的升级，已自动优化了JVM重用选项，MRv2开始不再支持JVM重用。（旧版本配置项：mapred.job.reuse.jvm.num.tasks、mapreduce.job.jvm.numtasks）

##### Hive执行引擎（了解）

CDH支持的引擎包括MapReduce和Spark两种，可自由选择，Spark不一定比MR快，Hive2.x和Hadoop3.x经过多次优化，Hive-MR引擎的性能已经大幅提升。

配置项：hive.execution.engine



CDH默认不支持Tez引擎：<https://docs.cloudera.com/documentation/enterprise/6/release-notes/topics/rg_cdh_620_unsupported_features.html>



### 全量流程

OLTP原始数据(mysql)——》数据采集(ODS)——》清洗转换(DWD)——》统计分析(DWS)——》导出至OLAP(Mysql)，如图：



#### 数据采集

##### web\_chat\_ems表

###### SQL：

|  |
| --- |
| select id,  create\_date\_time,  session\_id,  sid,  create\_time,  seo\_source,  seo\_keywords,  ip,  area,  country,  province,  city,  origin\_channel,  user as user\_match,  manual\_time,  begin\_time,  end\_time,  last\_customer\_msg\_time\_stamp,  last\_agent\_msg\_time\_stamp,  reply\_msg\_count,  msg\_count,  browser\_name,  os\_info,  "2019-07-01" as starts\_time from web\_chat\_ems\_2019\_07; |

###### Sqoop：

|  |
| --- |
| sqoop import \  --connect jdbc:mysql://192.168.52.150:3306/nev \  --username root \  --password 123456 \  --query 'select id, create\_date\_time, session\_id, sid, create\_time, seo\_source, seo\_keywords, ip, area, country, province, city, origin\_channel, user as user\_match, manual\_time, begin\_time, end\_time, last\_customer\_msg\_time\_stamp, last\_agent\_msg\_time\_stamp, reply\_msg\_count, msg\_count, browser\_name, os\_info, "2019-07-01" as starts\_time from web\_chat\_ems\_2019\_07 where $CONDITIONS' \  --hcatalog-database itcast\_ods \  --hcatalog-table web\_chat\_ems \  -m 100 \  --split-by id |

-m 100，指的是使用100个MapReduce任务并行处理；

而split-by参数，是指以哪个字段为基础进行分割。

##### web\_chat\_text\_ems表

###### SQL

|  |
| --- |
| select id,  referrer,  from\_url,  landing\_page\_url,  url\_title,  platform\_description,  other\_params,  history,  "2019-07-01" as start\_time from web\_chat\_text\_ems\_2019\_07; |

###### Sqoop

|  |
| --- |
| sqoop import \  --connect jdbc:mysql://192.168.52.150:3306/nev \  --username root \  --password 123456 \  --query 'select id,referrer,from\_url,landing\_page\_url,url\_title,platform\_description,other\_params,history, "2019-07-01" as start\_time from web\_chat\_text\_ems\_2019\_07 where $CONDITIONS' \  --hcatalog-database itcast\_ods \  --hcatalog-table web\_chat\_text\_ems \  -m 100 \  --split-by id |

#### 数据清洗转换

##### 时间函数

###### unix\_timestamp()日期转为时间戳

1.unix\_timestamp() 获取当前时间戳

例如：

|  |
| --- |
| select unix\_timestamp() --1565858389 |

2.unix\_timestamp(string timestame) 输入的时间戳格式必须为'yyyy-MM-dd HH:mm:ss',如不符合则返回null

例如：

|  |
| --- |
| select unix\_timestamp('2019-08-15 16:40:00') --1565858400  select unix\_timestamp('2019-08-15') --null |

3.unix\_timestamp(string date,string pattern) 将指定时间字符串格式字符串转化成unix时间戳,如不符合则返回null

例如：

|  |
| --- |
| select unix\_timestamp('2019-08-15','yyyy-MM-dd') --1565798400  select unix\_timestamp('2019-08-15 16:40:00','yyyy-MM-dd HH:mm:ss') --1565858400  select unix\_timestamp('2019-08-15','yyyy-MM-dd HH:mm:ss') --null |

###### from\_unixtime()时间戳转为日期

1. from\_unixtime(bigint unixtime,string format) 将时间戳秒数转化为UTC时间，并用字符串表示，可通过format规定的时间格式，指定输出的时间格式，其中unixtime 是10位的时间戳值，而13位的所谓毫秒的是不可以的。

例如：

|  |
| --- |
| select from\_unixtime(1565858389,'yyyy-MM-dd HH:mm:ss') --2019-08-15 16:39:49  select from\_unixtime(1565858389,'yyyy-MM-dd') --2019-08-15 |

1. 如果unixtime为13位的，需要先转成10位

|  |
| --- |
| select from\_unixtime(***cast(1553184000488/1000 as int)***,'yyyy-MM-dd HH:mm:ss') --2019-03-22 00:00:00  select from\_unixtime(cast(substr(1553184000488,1,10) as int),'yyyy-MM-dd HH:mm:ss') --2019-03-22 00:00:00 |

###### 当前时间

|  |
| --- |
| select from\_unixtime(unix\_timestamp(),'yyyy-MM-dd HH:mm:ss') -- 2019-08-15 17:18:55 |

###### 获取时间所在的季度

|  |
| --- |
| quarter('2015-04-08') --2 |

2015-04-08所在的季度为当年的第二个季度。

##### 字符串截取函数

大多数数据库中都有substr和substring两种字符串截取函数。但与其他的关系型数据库不同，在hive中，substr与substring函数的使用方式是完全一致的，属于同一个函数。

###### 两个参数

语法：substr(string A, int start)，substring(string A, int start)

返回值: string

说明：返回字符串A从start位置到结尾的字符串

栗子：

|  |
| --- |
| select substr('2020-06-06', 6), substring('2020-06-06', 6); --06-06 06-06 |

###### 三个参数

语法: substr(string A, int start, int len)，substring(string A, intstart, int len)

返回值: string

说明：返回字符串A从start位置开始，**长度**为len的字符串。

栗子：

|  |
| --- |
| select substr('2020-06-06', 6,2), substring('2020-06-06', 6,2); --06 06 |

##### 分析

从ODS层到DWD层，数据粒度是一致的，并且要保证数据的质量。主要做两件事：

1. 数据清洗：空数据、不满足业务需求的数据处理

对于访问客户量指标，已知的原始数据是经过咨询业务系统严格清洗过的数据，所以此处可以省略清洗过程。

1. 数据转换：数据格式和数据形式的转换，比如时间类型可以转换为同样的展现形式“yyyy-MM-dd HH:mm:ss”或者时间戳类型，金钱类型的数据可以统一转换为以元为单位或以分为单位的数值。

|  |  |  |
| --- | --- | --- |
| 原始字段 | 清洗转换 | 目标字段 |
| msg\_count（STRING） | 类型由String转为Int；  如果值为空，则转换为0； | msg\_count（INT） |
| create\_time（STRING） | 转为时间戳Int类型 | create\_time（BIGINT） |
| 切割年份 | yearinfo（STRING） |
| 切割月份 | monthinfo（STRING） |
| 切割日 | dayinfo（STRING） |
| 切割小时 | hourinfo（STRING） |
| 获取季度 | quarterinfo（STRING） |

##### 代码

|  |
| --- |
| *--动态分区配置* set hive.exec.dynamic.partition=true; set hive.exec.dynamic.partition.mode=nonstrict; *--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;  insert into table itcast\_dwd.visit\_consult\_dwd partition (yearinfo, monthinfo, dayinfo) select  wce.session\_id,  wce.sid,  *unix\_timestamp*(wce.create\_time, 'yyyy-MM-dd HH:mm:ss**.SSS**') as create\_time,  wce.seo\_source,  wce.ip,  wce.area,  *cast*(***if*(**wce.msg\_count is null, 0, wce.msg\_count**)** as int) as msg\_count,  wcte.referrer,  wcte.from\_url,  wcte.landing\_page\_url,  wcte.url\_title,  wcte.platform\_description,  wcte.other\_params,  wcte.history,  *substr*(wce.create\_time, 12, 2) as hourinfo,  *quarter*(wce.create\_time) as quarterinfo,  *substr*(wce.create\_time, 1, 4) as yearinfo,  *substr*(wce.create\_time, 6, 2) as monthinfo,  *substr*(wce.create\_time, 9, 2) as dayinfo from itcast\_ods.web\_chat\_ems wce inner join itcast\_ods.web\_chat\_text\_ems wcte on wce.id = wcte.id; |

##### 问题

过多的动态分区会导致如下错误：

|  |
| --- |
| Error: java.lang.RuntimeException: org.apache.hadoop.hive.ql. metadata.HiveFatalException: [Error 20004]: Fatal error occurred when node tried to create too many dynamic partitions. The maximum number of dynamic partitions is controlled by hive.exec.max.dynamic.partitions and hive.exec.max.dynamic.partitions.pernode. Maximum was set to: 100 |

解决：

|  |
| --- |
| set hive.exec.max.dynamic.partitions.pernode=10000; set hive.exec.max.dynamic.partitions=100000; |

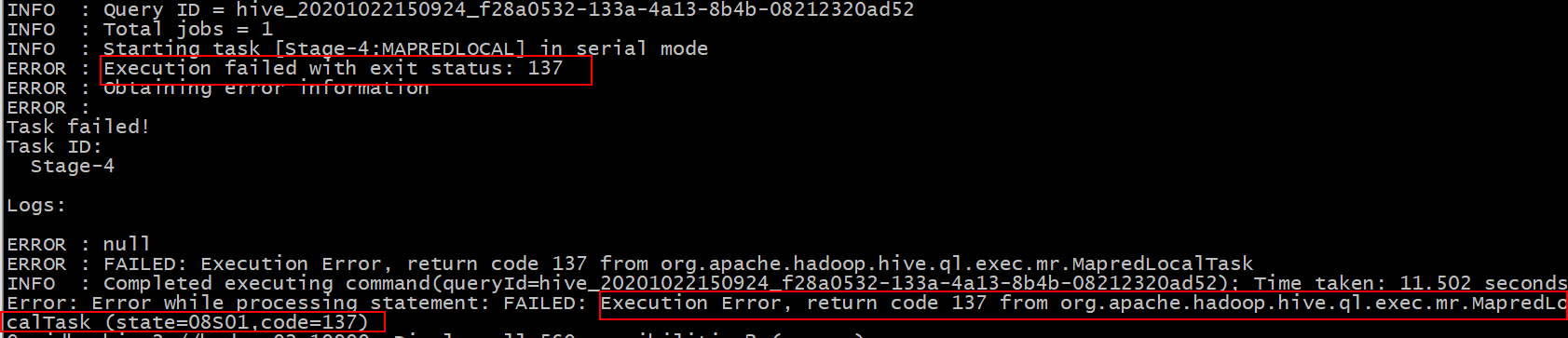
Hive动态分区创建文件数过多错误：

|  |
| --- |
| [Fatal Error] total number of created files now is 100385, which exceeds 100000. Killing the job. |

解决：

|  |
| --- |
| set hive.exec.max.created.files=150000; |

问题2:



Execution failed with exit status: 137

出现原因:

可能是由于服务器内存不足引起的, 在进行mapJoin时候, 内存不足以放下小表中的数据

解决方案: 不让其进行mapjoin优化操作

-- 关闭自动装载(尤其是对于内存比较小的机器)  
set hive.auto.convert.join= false;

#### 统计分析

##### 分析

DWD层之后是DWM中间层和DWS业务层。回顾建模分析阶段，我们已经得到了指标相关的维度：年、季度、月、天、小时、地区、来源渠道、页面。分两大类：

时间维度：年、季度、月、天、小时

业务属性维度：地区、来源渠道、页面、总访问量。

在DWS层按照不同维度使用count+distinct来统计指标，形成宽表。

**空值处理**

事实表中的维度关联键不能存在空值，关联的维度信息必须用代理键（-1）而不是空值表示未知的条件。

##### 代码

我们的维度一共有两大类：时间维度和产品属性维度，在DWS层我们可以产出一个宽表，将所有维度的数据都生成出来，供APP层和OLAP应用来使用。

###### 地区分组

统计地区维度时，需要设置产品属性类型groupType为1（地区），同时将其他产品属性设置为-1（搜索来源、来源渠道、会话来源页面），便于团队理解，减少自己和团队出错率的同时也降低了沟通成本。

在insertsql中，尽量为查询出的字段加上别名，特别是字段多的表，便于识别。

小时维度：

|  |
| --- |
| *--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;  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  area,  '-1' as seo\_source,  '-1' as origin\_channel,  hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time\_str,  '-1' as from\_url,  '1' as grouptype,  '1' as time\_type,  yearinfo, monthinfo, dayinfo from itcast\_dwd.visit\_consult\_dwd group by area, yearinfo, quarterinfo, monthinfo, dayinfo, hourinfo; |

天维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select   *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo,'-',dayinfo) as time\_str,  '-1' as from\_url,  '1' as grouptype,  '2' as time\_type,  yearinfo, monthinfo, dayinfo from itcast\_dwd.visit\_consult\_dwd  group by area, yearinfo, quarterinfo, monthinfo, dayinfo; |

月维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select   *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo) as time\_str,  '-1' as from\_url,  '1' as grouptype,  '3' as time\_type,  yearinfo, monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd  group by area, yearinfo, quarterinfo, monthinfo; |

季度维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select   *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-Q',quarterinfo) as time\_str,  '-1' as from\_url,  '1' as grouptype,  '4' as time\_type,  yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd  group by area, yearinfo, quarterinfo; |

年维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select   *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.session\_id) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  wce.area as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  '-1' as from\_url,  '1' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce group by wce.area,wce.yearinfo; |

###### 搜索来源分组

小时维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select   *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  seo\_source,  '-1' as origin\_channel,  hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time\_str,  '-1' as from\_url,  '2' as grouptype,  '1' as time\_type,  yearinfo, monthinfo, dayinfo from itcast\_dwd.visit\_consult\_dwd  group by seo\_source, yearinfo, quarterinfo, monthinfo, dayinfo, hourinfo; |

天维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo,'-',dayinfo) as time\_str,  '-1' as from\_url,  '2' as grouptype,  '2' as time\_type,  yearinfo, monthinfo, dayinfo from itcast\_dwd.visit\_consult\_dwd group by seo\_source, yearinfo, quarterinfo, monthinfo, dayinfo; |

月维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo) as time\_str,  '-1' as from\_url,  '2' as grouptype,  '3' as time\_type,  yearinfo, monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd group by seo\_source, yearinfo, quarterinfo, monthinfo; |

季度维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-Q',quarterinfo) as time\_str,  '-1' as from\_url,  '2' as grouptype,  '4' as time\_type,  yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd group by seo\_source, yearinfo, quarterinfo; |

年维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.session\_id) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  '-1' as from\_url,  '2' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce group by wce.seo\_source,wce.yearinfo; |

###### 来源渠道分组

小时维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  origin\_channel,  hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time\_str,  '-1' as from\_url,  '3' as grouptype,  '1' as time\_type,  yearinfo, monthinfo, dayinfo from itcast\_dwd.visit\_consult\_dwd group by origin\_channel, yearinfo, quarterinfo, monthinfo, dayinfo, hourinfo; |

天维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo,'-',dayinfo) as time\_str,  '-1' as from\_url,  '3' as grouptype,  '2' as time\_type,  yearinfo, monthinfo, dayinfo from itcast\_dwd.visit\_consult\_dwd group by origin\_channel, yearinfo, quarterinfo, monthinfo, dayinfo; |

月维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo) as time\_str,  '-1' as from\_url,  '3' as grouptype,  '3' as time\_type,  yearinfo, monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd group by origin\_channel, yearinfo, quarterinfo, monthinfo; |

季度维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-Q',quarterinfo) as time\_str,  '-1' as from\_url,  '3' as grouptype,  '4' as time\_type,  yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd group by origin\_channel, yearinfo, quarterinfo; |

年维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.session\_id) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  '-1' as from\_url,  '3' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce group by wce.origin\_channel,wce.yearinfo; |

###### 会话来源页面分组

小时维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select   *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time\_str,  from\_url,  '4' as grouptype,  '1' as time\_type,  yearinfo, monthinfo, dayinfo from itcast\_dwd.visit\_consult\_dwd  group by from\_url, yearinfo, quarterinfo, monthinfo, dayinfo, hourinfo; |

天维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo,'-',dayinfo) as time\_str,  from\_url,  '4' as grouptype,  '2' as time\_type,  yearinfo, monthinfo, dayinfo from itcast\_dwd.visit\_consult\_dwd group by from\_url, yearinfo, quarterinfo, monthinfo, dayinfo; |

月维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo) as time\_str,  from\_url,  '4' as grouptype,  '3' as time\_type,  yearinfo, monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd group by from\_url, yearinfo, quarterinfo, monthinfo; |

季度维度：

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-Q',quarterinfo) as time\_str,  from\_url,  '4' as grouptype,  '4' as time\_type,  yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd group by from\_url, yearinfo, quarterinfo; |

年维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.session\_id) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  from\_url,  '4' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce group by wce.from\_url,wce.yearinfo; |

###### 总访问量

小时（小时段区间的基础数据）

因为小时段数据可以直接sum求和，因此OLAP应用可以在小时数据基础上，进行简单的sum操作以获取到区间小时段数据。

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select   *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time\_str,  '-1' as from\_url,  '5' as grouptype,  '1' as time\_type,  yearinfo, monthinfo, dayinfo from itcast\_dwd.visit\_consult\_dwd  group by yearinfo, quarterinfo, monthinfo, dayinfo, hourinfo; |

天

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo,'-',dayinfo) as time\_str,  '-1' as from\_url,  '5' as grouptype,  '2' as time\_type,  yearinfo, monthinfo, dayinfo from itcast\_dwd.visit\_consult\_dwd group by yearinfo, quarterinfo, monthinfo, dayinfo; |

月

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-',monthinfo) as time\_str,  '-1' as from\_url,  '5' as grouptype,  '3' as time\_type,  yearinfo, monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd group by yearinfo, quarterinfo, monthinfo; |

季度

|  |
| --- |
| insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid) as sid\_total,  *count*(distinct session\_id) as session\_total,  *count*(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat*(yearinfo,'-Q',quarterinfo) as time\_str,  '-1' as from\_url,  '5' as grouptype,  '4' as time\_type,  yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd group by yearinfo, quarterinfo; |

年

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.session\_id) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  '-1' as from\_url,  '5' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce group by wce.yearinfo; |

#### 导出数据

##### 创建mysql表

|  |
| --- |
| create database scrm\_bi default character set utf8mb4 collate utf8mb4\_general\_ci; |

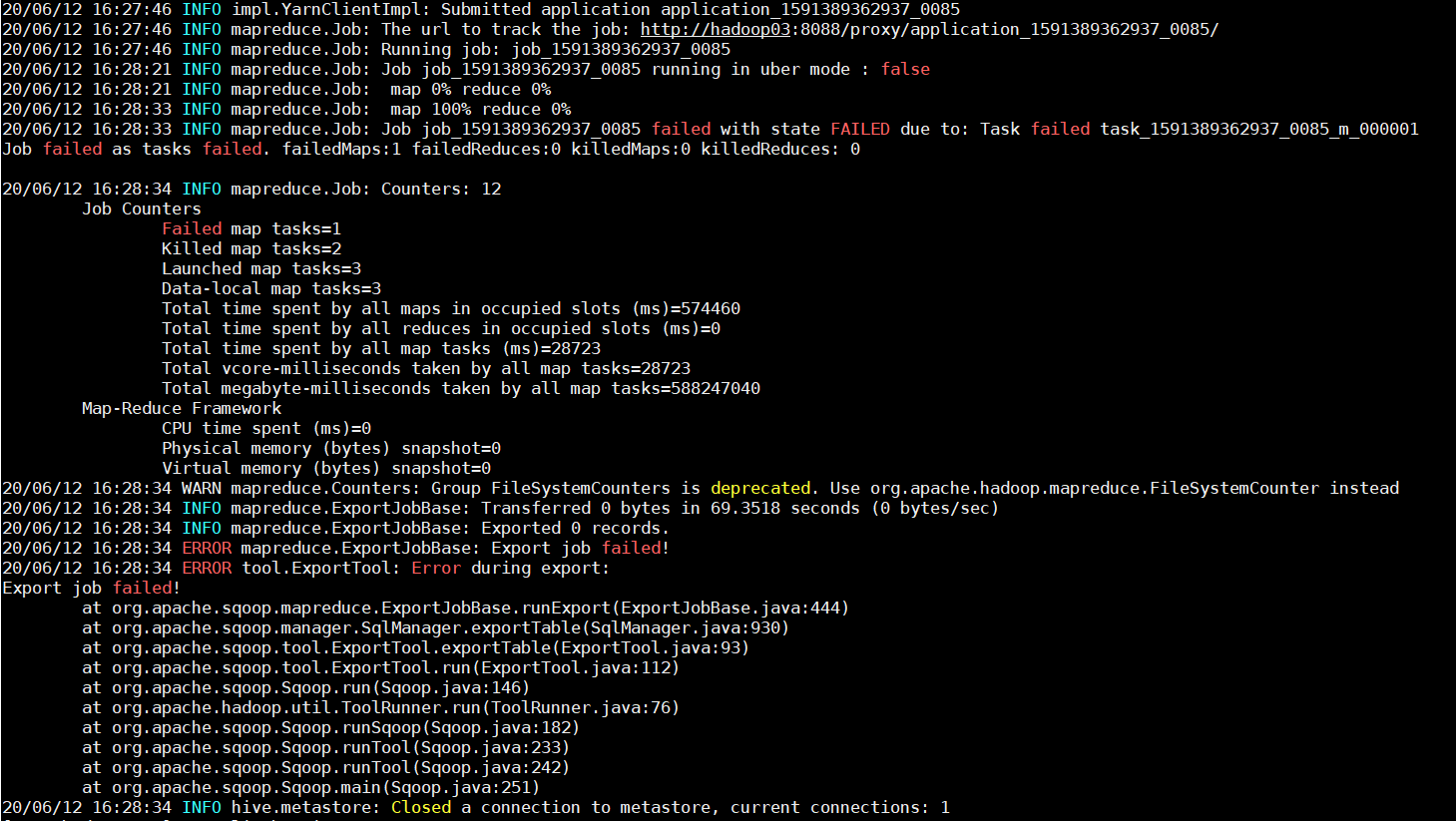
|  |
| --- |
| CREATE TABLE `itcast\_visit` (  sid\_total int(11) COMMENT '根据sid去重求count',  sessionid\_total int(11) COMMENT '根据sessionid去重求count',  ip\_total int(11) COMMENT '根据IP去重求count',  area varchar(32) COMMENT '区域信息',  seo\_source varchar(32) COMMENT '搜索来源',  origin\_channel varchar(32) COMMENT '来源渠道',  hourinfo varchar(32) COMMENT '小时信息',  quarterinfo varchar(32) COMMENT '季度',  time\_str varchar(32) COMMENT '时间明细',  from\_url varchar(32) comment '会话来源页面',  groupType varchar(32) COMMENT '产品属性类型：1.地区；2.搜索来源；3.来源渠道；4.会话来源页面；5.总访问量',  time\_type varchar(32) COMMENT '时间聚合类型：1、按小时聚合；2、按天聚合；3、按月聚合；4、按季度聚合；5、按年聚合；',  yearinfo varchar(32) COMMENT '年信息',  monthinfo varchar(32) COMMENT '月信息',  dayinfo varchar(32) COMMENT '日信息' ); |

##### 执行sqoop导出脚本

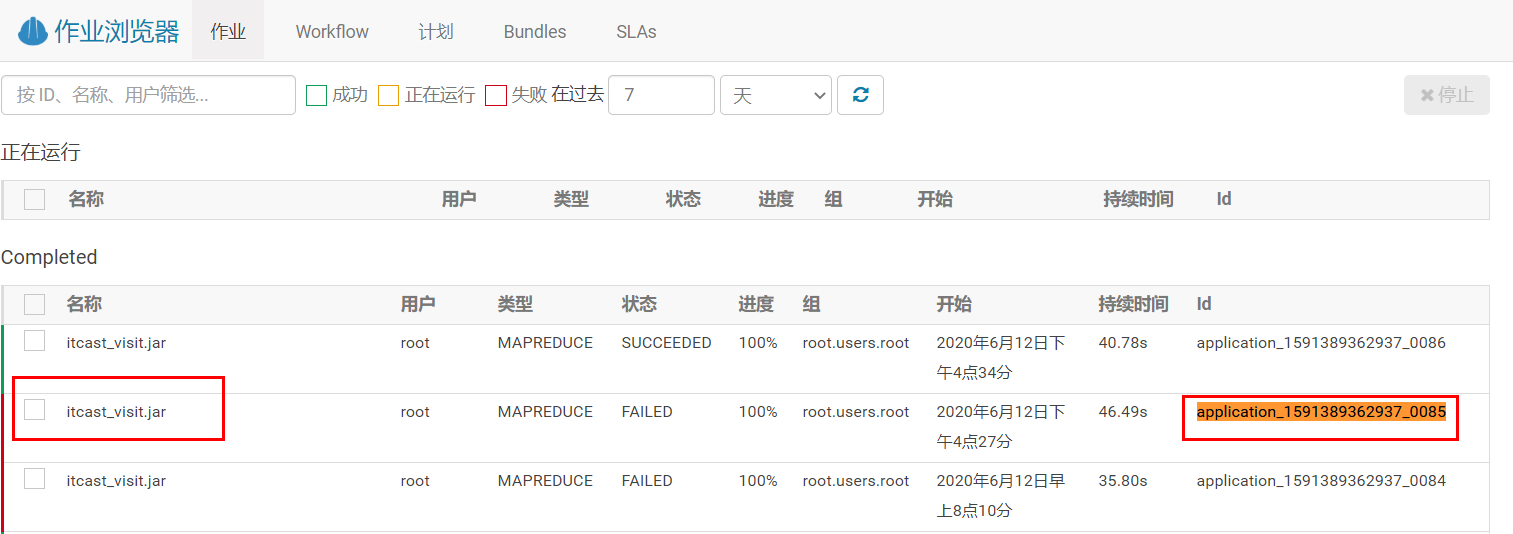
|  |
| --- |
| sqoop **export** \ --connect "jdbc:mysql://192.168.52.150:3306/scrm\_bi?useUnicode=true&characterEncoding=utf-8" \ --username root \ --password '123456' \ --table itcast\_visit \ --hcatalog-database itcast\_dws \ --hcatalog-table visit\_dws \ -m 100 |

##### MR错误日志

执行错误：

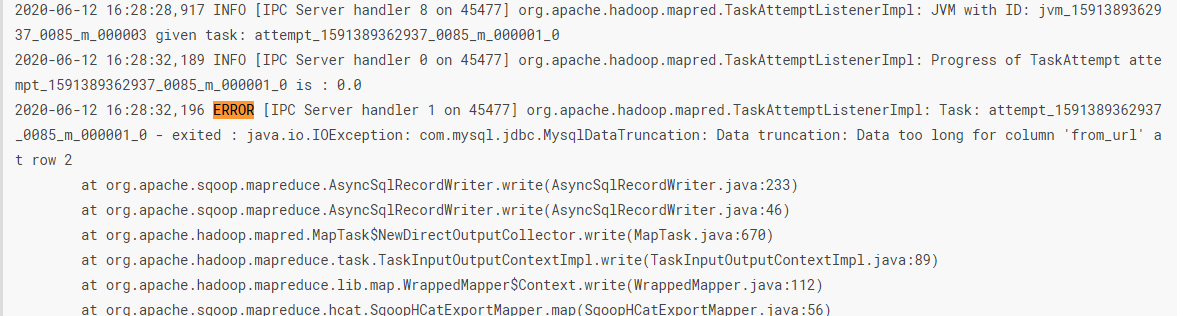


在hue作业中找到application\_1591389362937\_0085：



查看具体错误信息：





原因是from\_url字段长度不够，修改后再次执行：

|  |
| --- |
| drop table itcast\_visit;  CREATE TABLE `itcast\_visit` (  sid\_total int(11) COMMENT '根据sid去重求count',  sessionid\_total int(11) COMMENT '根据sessionid去重求count',  ip\_total int(11) COMMENT '根据IP去重求count',  area varchar(32) COMMENT '区域信息',  seo\_source varchar(32) COMMENT '搜索来源',  origin\_channel varchar(32) COMMENT '来源渠道',  hourinfo varchar(32) COMMENT '小时信息',  quarterinfo varchar(32) COMMENT '季度',  time\_str varchar(32) COMMENT '时间明细',  from\_url varchar(2083) comment '会话来源页面',  groupType varchar(32) COMMENT '产品属性类型：1.地区；2.搜索来源；3.来源渠道；4.会话来源页面',  time\_type varchar(32) COMMENT '时间聚合类型：1、按小时聚合；2、按天聚合；3、按月聚合；4、按季度聚合；5、按年聚合；',  yearinfo varchar(32) COMMENT '年信息',  monthinfo varchar(32) COMMENT '月信息',  dayinfo varchar(32) COMMENT '日信息' )ENGINE=InnoDB AUTO\_INCREMENT=22 DEFAULT CHARSET=utf8 COLLATE=utf8\_bin; |

再次执行sqoop脚本，执行成功。

### 增量流程

#### 数据采集

增量更新和全量更新区别是，增量更新采用的是T+1模式，分析的数据只有一天的量。

1. Sql中需要增加where条件，只查询昨天一天的数据，而不是所有表数据。
2. Sqoop指定分区：增量更新每次只更新一天的数据，也可以使用静态分区方式导入。通过指定hive-partition-key和hive-partition-value两个参数实现。hive-partition-key用来指定分区字段名，hive-partition-value用来指定分区的值，也就是昨天的日期（今天0点采集的数据属于昨天）。

##### 造数据

**在mysql中执行：**

|  |
| --- |
| *# 备份表* create table web\_chat\_ems\_2019\_07\_back select *\** from web\_chat\_ems\_2019\_07;  *# 将2019-07-01变更为2019-07-25，模拟新数据* update web\_chat\_ems\_2019\_07 set create\_time = *concat*('2019-07-25 ', *substr*(create\_time, 12)) where create\_time between '2019-07-01 00:00:00' and '2019-07-02 00:00:00'; |

##### SQL脚本

###### web\_chat\_ems表

|  |
| --- |
| select id,  create\_date\_time,  session\_id,  sid,  create\_time,  seo\_source,  seo\_keywords,  ip,  area,  country,  province,  city,  origin\_channel,  user as user\_match,  manual\_time,  begin\_time,  end\_time,  last\_customer\_msg\_time\_stamp,  last\_agent\_msg\_time\_stamp,  reply\_msg\_count,  msg\_count,  browser\_name,  os\_info,  "2019-07-25" as starts\_time from web\_chat\_ems\_2019\_07 where create\_time between "2019-07-25 00:00:00" and "2019-07-25 23:59:59" and $CONDITIONS; |

###### web\_chat\_text\_ems表

副表没有创建日期字段，通过关联主表，使用主表的日期字段来进行判断。

|  |
| --- |
| select id,  referrer,  from\_url,  landing\_page\_url,  url\_title,  platform\_description,  other\_params,  history,  "2019-07-25" as start\_time from web\_chat\_text\_ems\_2019\_07 wcte,  (select id as wce\_id, create\_time  from web\_chat\_ems\_2019\_07  where create\_time between "2019-07-25 00:00:00" and "2019-07-25 23:59:59") wce where wcte.id = wce.wce\_id  and $CONDITIONS; |

##### sqoop脚本

###### web\_chat\_ems表

|  |
| --- |
| sqoop import \  --connect jdbc:mysql://192.168.52.150:3306/nev \  --username root \  --password 123456 \  --query 'select id,  create\_date\_time,  session\_id,  sid,  create\_time,  seo\_source,  seo\_keywords,  ip,  area,  country,  province,  city,  origin\_channel,  user as user\_match,  manual\_time,  begin\_time,  end\_time,  last\_customer\_msg\_time\_stamp,  last\_agent\_msg\_time\_stamp,  reply\_msg\_count,  msg\_count,  browser\_name,  os\_info,  "2019-07-25" as starts\_time  from web\_chat\_ems\_2019\_07  where create\_time between "2019-07-25 00:00:00" and "2019-07-25 23:59:59" and $CONDITIONS' \  --hcatalog-database itcast\_ods \  --hcatalog-table web\_chat\_ems \  --hive-partition-key starts\_time \  --hive-partition-value 2019-07-25 \  -m 100 \  --split-by id |

###### web\_chat\_text\_ems表

|  |
| --- |
| sqoop import \  --connect jdbc:mysql://192.168.52.150:3306/nev \  --username root \  --password 123456 \  --query '  select id,  referrer,  from\_url,  landing\_page\_url,  url\_title,  platform\_description,  other\_params,  history,  "2019-07-25" as start\_time  from web\_chat\_text\_ems\_2019\_07 wcte,  (select id as wce\_id, create\_time  from web\_chat\_ems\_2019\_07  where create\_time between "2019-07-25 00:00:00" and "2019-07-25 23:59:59") wce  where wcte.id = wce.wce\_id  and $CONDITIONS' \  --hcatalog-database itcast\_ods \  --hcatalog-table web\_chat\_text\_ems \  --**hive-partition-key** start\_time \  --**hive-partition-value** 2019-07-25 \  -m 100 \  --split-by id |

##### shell脚本

###### date命令

获取今天的日期

|  |
| --- |
| date  date +%Y%m%d |

**指定日期获取内容 -d或--date=**

|  |
| --- |
| **#获取指定日期的年月日格式输出**  date -d "2014-11-12" +%Y%m%d  **#获取指定日期的星期（周几）格式输出**  date --date="2014-11-23" +%w |

**日期加减 -d或--date==**

|  |
| --- |
| **#获取上周日期**（day,month,year,hour）  date **-d** "-1 week" +%Y%m%d  **#获取昨天日期**  date **-d** '-1 day' "+%Y-%m-%d"  date **--date=**"-24 hour" +%Y%m%d |

日期格式化规则：

%a 当前域的星期缩写 (Sun..Sat)

%A 当前域的星期全写 (Sunday..Saturday)

%b 当前域的月份缩写(Jan..Dec)

%B 当前域的月份全称 (January..December)

%d 两位的天 (01..31)

%D 短时间格式 (mm/dd/yy)

%e 短格式天 ( 1..31)

%F 文件时间格式 same as %Y-%m-%d

%h same as %b

%H 24小时制的小时 (00..23)

%I 12小时制的小时 (01..12)

%j 一年中的第几天 (001..366)

%k 短格式24小时制的小时 ( 0..23)

%m 双位月份 (01..12)

%M 双位分钟 (00..59)

%r 12小时制的时间表示（时:分:秒,双位） time, 12-hour (hh:mm:ss [AP]M)

%R 24小时制的时间表示 （时:分,双位）time, 24-hour (hh:mm)

%s 自基础时间 1970-01-01 00:00:00 到当前时刻的秒数(a GNU extension)

%T 24小时制时间表示(hh:mm:ss)

%u 数字表示的星期（从星期一开始 1-7）

%x 本地日期格式 (mm/dd/yy)

%X 本地时间格式 (%H:%M:%S)

%y 两位的年(00..99)

%Y 年 (1970…)

###### ${} 与 $()、`` 与 (())、$(())

变量替换

在 bash shell 中，**${}**是用来作变量替换的。

一般情况下，$var与${var}是没有区别的，但是用${ }会比较精确的界定变量名称的范围。

|  |
| --- |
| A=Linux  echo $AB #表示变量AB  echo ${A}B #表示变量A后连接着B |

命令替换

在 bash shell 中，**$( )与` `**（反引号）都是用来作命令替换的。

命令替换与变量替换差不多，区别是一个为执行变量，一个为执行命令，先完成引号里的命令行，然后将其结果替换出来，再重组成新的命令行。

|  |
| --- |
| echo $(date "+%Y-%m-%d") #2020-07-01  echo today is `date "+%Y-%m-%d"` #today is 2020-07-01 |

数学运算

(())

双小括号命令是用来执行数学表达式的，可以在其中进行各种逻辑运算、数学运算，也支持更多的运算符（如++、--等）。

|  |
| --- |
| echo $(((5 \* 2))) #10 |

在(( )) 中的变量名称，可于其前面加 $ 符号来执行替换，也可以不用。

|  |
| --- |
| i=5  echo $(((i=$i\*2))) #10  echo $(((i=i\*2))) #20 |

$(())

$((( )))的缩写。

|  |
| --- |
| echo $(((i\*2))) #40  echo $((i\*2)) #40 |

###### 串行与并行

shell脚本默认是按顺序串行执行的，使用&可以将一个命令放在后台运行，从而使shell脚本能够继续往后执行：

|  |
| --- |
| sleep 5 **&**  echo "done" |

上面的脚本执行后会立即打印出"done"，sleep命令被扔给后台执行，不会阻塞脚本执行。

如果想要在进入下个循环前，必须等待上个后台命令执行完毕，可以使用wait命令：

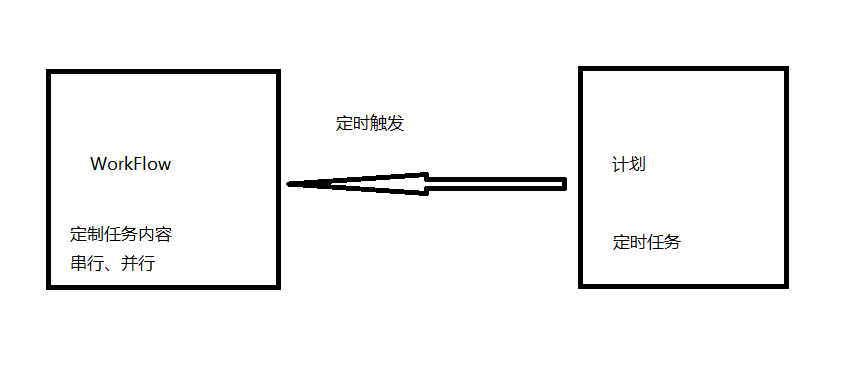
|  |
| --- |
| sleep 5 &  **wait**  echo "done" |

这样，需要等待5s后才能在屏幕上看到"done"。

###### 增量采集Shell

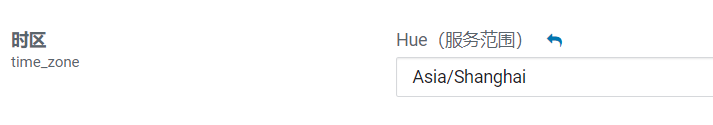
|  |
| --- |
| #! /bin/bash  #SQOOP\_HOME=/opt/cloudera/parcels/CDH-6.2.1-1.cdh6.2.1.p0.1425774/bin/sqoop  SQOOP\_HOME=/usr/bin/sqoop  if [[ $1 == "" ]];then  TD\_DATE=`date -d '1 days ago' "+%Y-%m-%d"`  else  TD\_DATE=$1  fi  V\_YEAR=$(date --date="${TD\_DATE}" +%Y)  V\_MONTH=$(date --date="${TD\_DATE}" +%m)  V\_TABLE\_web\_chat\_ems="web\_chat\_ems\_${V\_YEAR}\_${V\_MONTH}"  V\_TABLE\_web\_chat\_text\_ems="web\_chat\_text\_ems\_${V\_YEAR}\_${V\_MONTH}"  ${SQOOP\_HOME} import \  --connect jdbc:mysql://192.168.52.150:3306/nev \  --username root \  --password 123456 \  --query "select id,  create\_date\_time,  session\_id,  sid,  create\_time,  seo\_source,  seo\_keywords,  ip,  area,  country,  province,  city,  origin\_channel,  user as user\_match,  manual\_time,  begin\_time,  end\_time,  last\_customer\_msg\_time\_stamp,  last\_agent\_msg\_time\_stamp,  reply\_msg\_count,  msg\_count,  browser\_name,  os\_info,  '${TD\_DATE}' as starts\_time  from ${V\_TABLE\_web\_chat\_ems}  where create\_time between '${TD\_DATE} 00:00:00' and '${TD\_DATE} 23:59:59' and \$CONDITIONS" \  --hcatalog-database itcast\_ods \  --hcatalog-table web\_chat\_ems \  --hive-partition-key starts\_time \  --hive-partition-value ${TD\_DATE} \  -m 100 \  --split-by id  #如果想要确保等待上个后台命令执行完毕再执行后续任务，可以使用wait命令：  wait  ${SQOOP\_HOME} import \  --connect jdbc:mysql://192.168.52.150:3306/nev \  --username root \  --password 123456 \  --query "  select id,  referrer,  from\_url,  landing\_page\_url,  url\_title,  platform\_description,  other\_params,  history,  '${TD\_DATE}' as start\_time  from ${V\_TABLE\_web\_chat\_text\_ems} wcte,  (select id as wce\_id, create\_time  from ${V\_TABLE\_web\_chat\_ems}  where create\_time between '${TD\_DATE} 00:00:00' and '${TD\_DATE} 23:59:59') wce  where wcte.id = wce.wce\_id  and \$CONDITIONS" \  --hcatalog-database itcast\_ods \  --hcatalog-table web\_chat\_text\_ems \  --hive-partition-key start\_time \  --hive-partition-value ${TD\_DATE} \  -m 100 \  --split-by id |

##### Oozie调度



###### 时区设置

1. ClouderaManager中，hue 配置修改time\_zone的值为：Asia/Shanghai



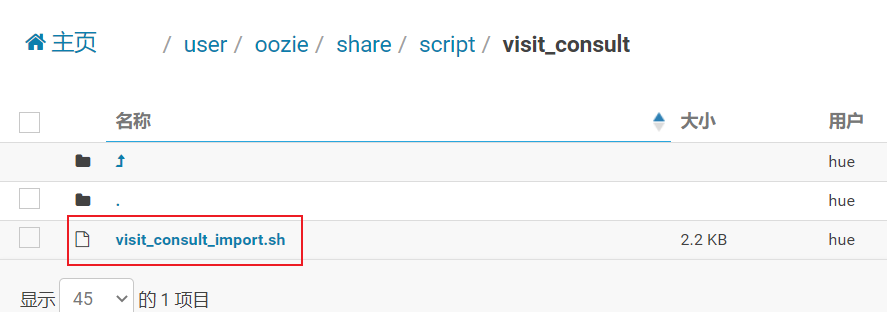
1. ClouderaManager中，oozie配置修改oozie-site.xml，添加配置项：

|  |
| --- |
| name: oozie.processing.timezone  Value: GMT+0800  Description: oozie时区设置为东八区区时 |

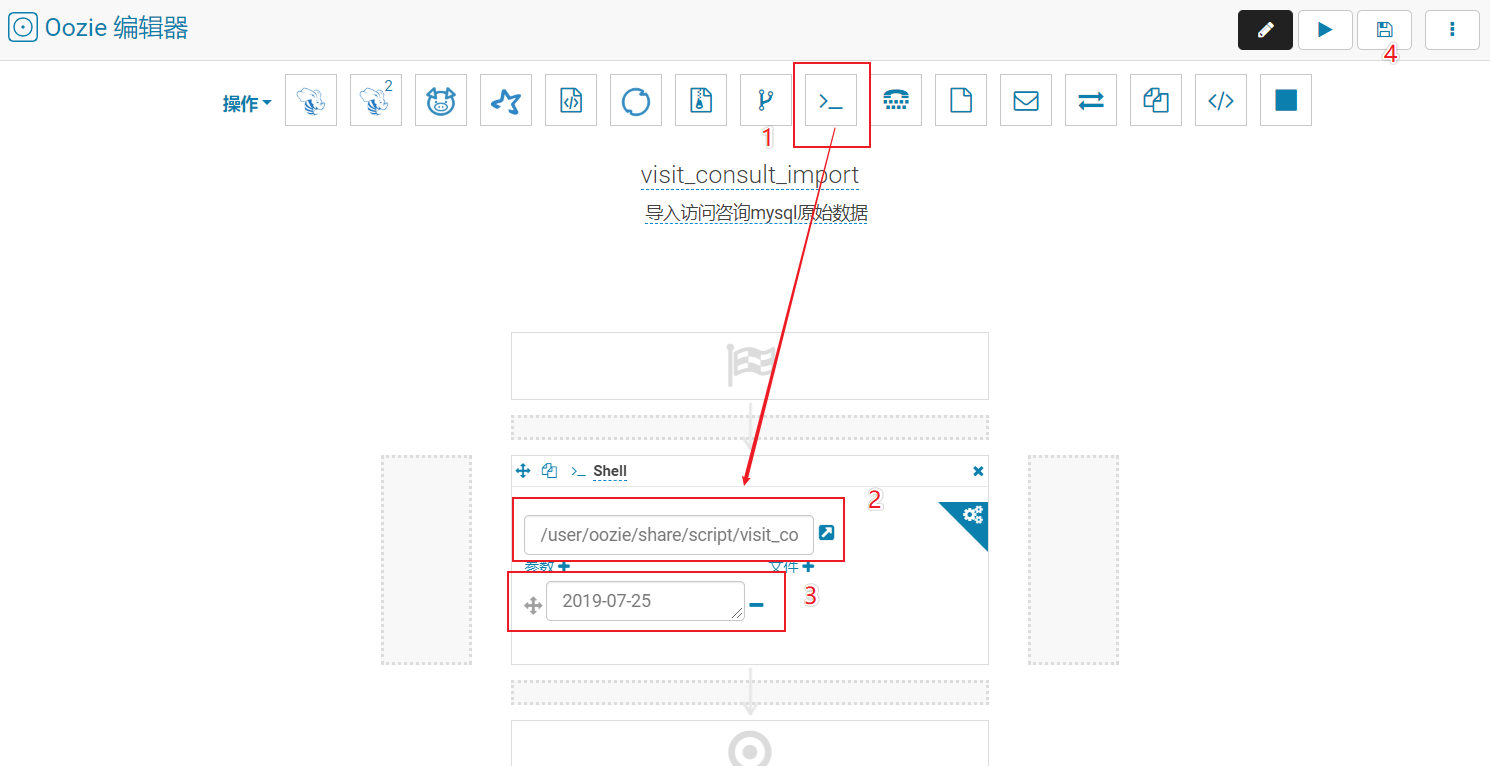


###### 创建workflow

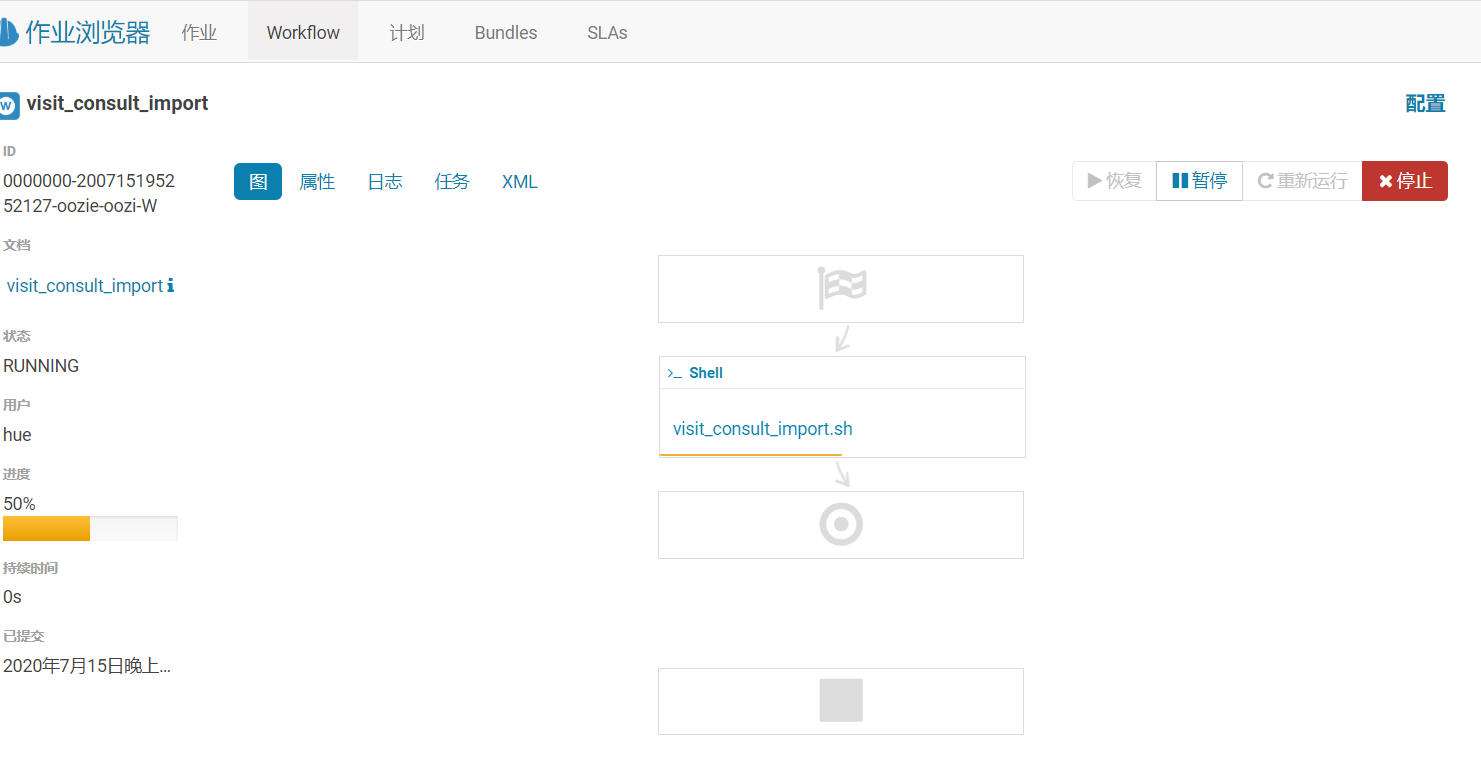
上传shell脚本到HDFS：



设置workflow



测试workflow

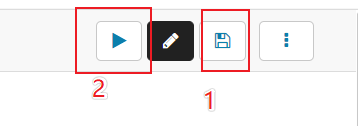


###### 创建调度计划

创建计划，注意时区要设置正确。



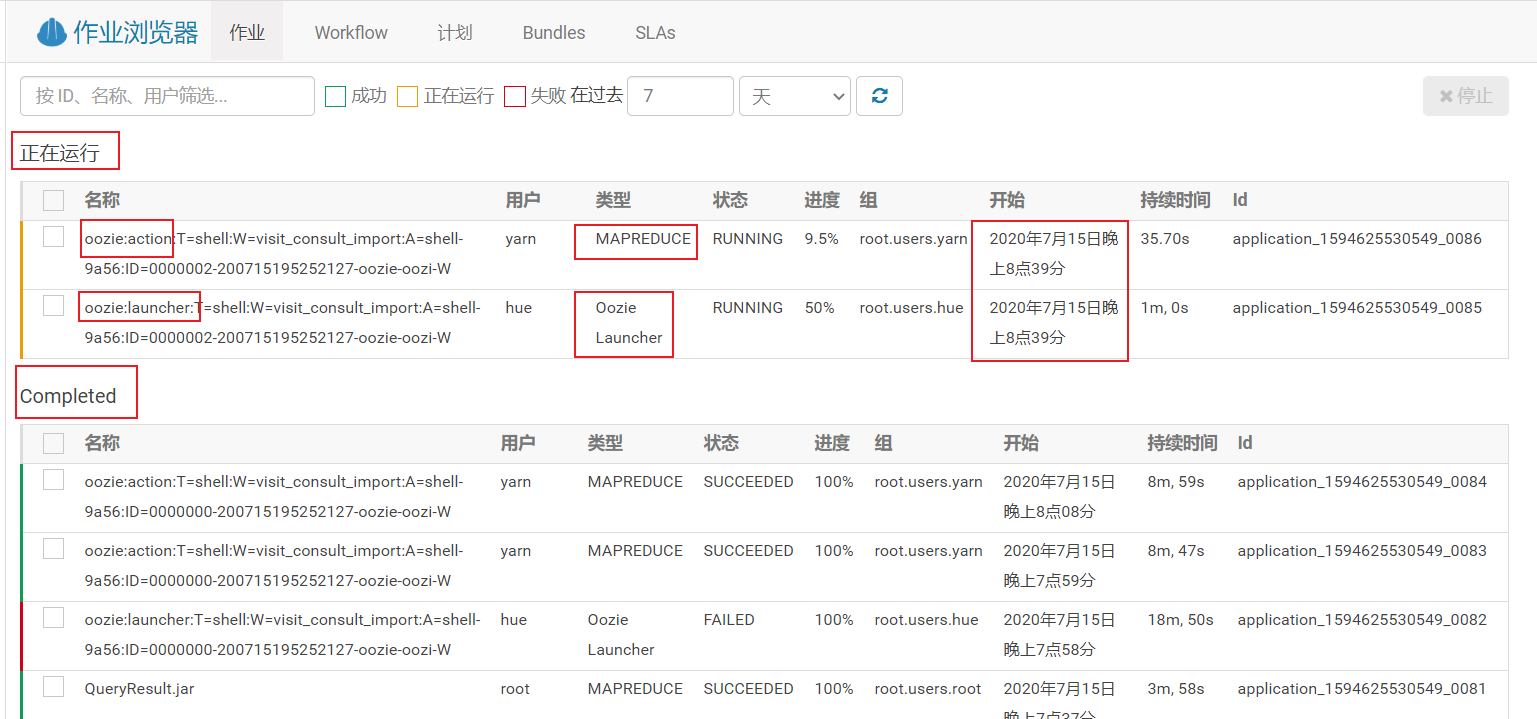
保存，并提交

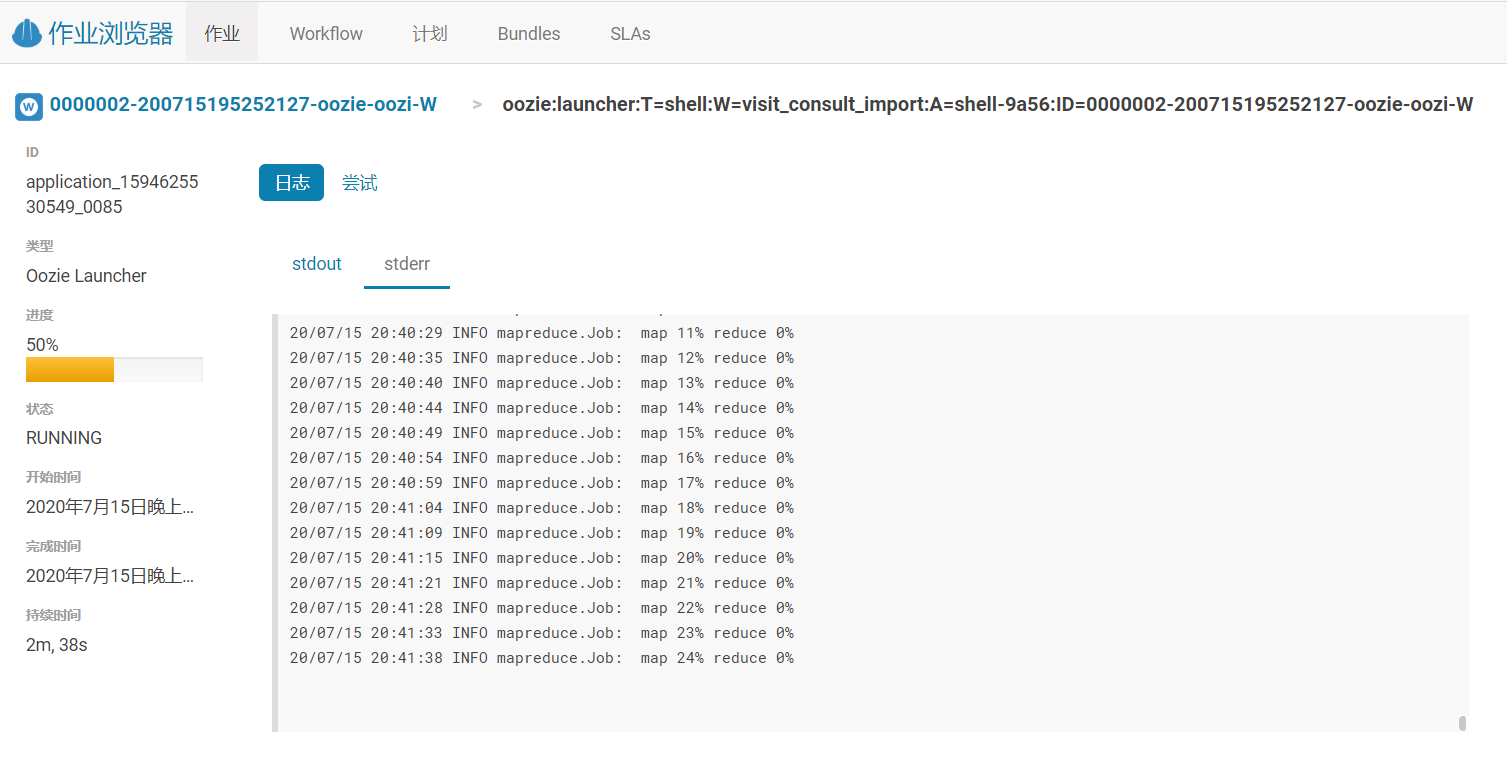


查看提交列表

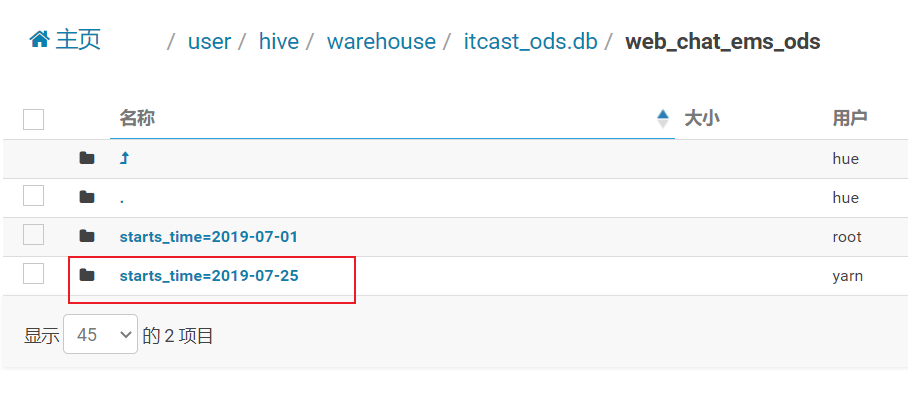


查看触发记录





执行成功



#### 数据清洗转换

增量清洗转换时，如果同一天的分区已有旧数据，需要覆盖掉，则可以使用**Insert overwrite**覆盖写入，或者通过sql语句删除。否则会出现数据重复，导致后续计算出错。（注意内部表不能直接删除HDFS分区）

需要增加where条件，指向ODS层的采集日期start\_time字段，只清洗转换昨天的数据，旧数据已经计算过，不需要重复计算。

##### SQL

|  |
| --- |
| *--动态分区配置* set hive.exec.dynamic.partition=true; set hive.exec.max.dynamic.partitions=2000; 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;  insert into table itcast\_dwd.visit\_consult\_dwd partition (yearinfo, monthinfo, dayinfo) select   wce.session\_id,  wce.sid,  *unix\_timestamp*(wce.create\_time, 'yyyy-MM-dd HH:mm:ss.SSS') as create\_time,  wce.seo\_source,  wce.ip,  wce.area,  *cast*(*if*(wce.msg\_count is null, 0, wce.msg\_count) as int) as msg\_count,  wce.origin\_channel,  wcte.referrer,  wcte.from\_url,  wcte.landing\_page\_url,  wcte.url\_title,  wcte.platform\_description,  wcte.other\_params,  wcte.history,  *substr*(wce.create\_time, 12, 2) as hourinfo,  *quarter*(wce.create\_time) as quarterinfo,  *substr*(wce.create\_time, 1, 4) as yearinfo,  *substr*(wce.create\_time, 6, 2) as monthinfo,  *substr*(wce.create\_time, 9, 2) as dayinfo from itcast\_ods.web\_chat\_ems wce inner join itcast\_ods.web\_chat\_text\_ems wcte on wce.id = wcte.id where wce.starts\_time='2019-07-25'; |

##### Shell脚本

|  |
| --- |
| #! /bin/bash  HIVE\_HOME=/usr/bin/hive  if [[ $1 == "" ]];then  TD\_DATE=`date -d '1 days ago' "+%Y-%m-%d"`  else  TD\_DATE=$1  fi  ${HIVE\_HOME} -S -e "  --动态分区配置  set hive.exec.dynamic.partition=true;  set hive.exec.max.dynamic.partitions=2000;  set hive.exec.dynamic.partition.mode=nonstrict;  set hive.exec.max.dynamic.partitions.pernode=10000;  set hive.exec.max.dynamic.partitions=100000;  set hive.exec.max.created.files=150000;  --hive压缩  set hive.exec.compress.intermediate=true;  set hive.exec.compress.output=true;  --写入时压缩生效  set hive.exec.orc.compression.strategy=COMPRESSION;  insert into table itcast\_dwd.visit\_consult\_dwd partition (yearinfo, monthinfo, dayinfo)  select  wce.session\_id,  wce.sid,  unix\_timestamp(wce.create\_time, 'yyyy-MM-dd HH:mm:ss.SSS') as create\_time,  wce.seo\_source,  wce.ip,  wce.area,  cast(if(wce.msg\_count is null, 0, wce.msg\_count) as int) as msg\_count,  wce.origin\_channel,  wcte.referrer,  wcte.from\_url,  wcte.landing\_page\_url,  wcte.url\_title,  wcte.platform\_description,  wcte.other\_params,  wcte.history,  substr(wce.create\_time, 12, 2) as hourinfo,  quarter(wce.create\_time) as quarterinfo,  substr(wce.create\_time, 1, 4) as yearinfo,  substr(wce.create\_time, 6, 2) as monthinfo,  substr(wce.create\_time, 9, 2) as dayinfo  from itcast\_ods.web\_chat\_ems wce inner join itcast\_ods.web\_chat\_text\_ems wcte  on wce.id = wcte.id  where wce.starts\_time='${TD\_DATE}';  " |

运行后报错：Failing Oozie Launcher, output.properties data exceeds its limit [2048]

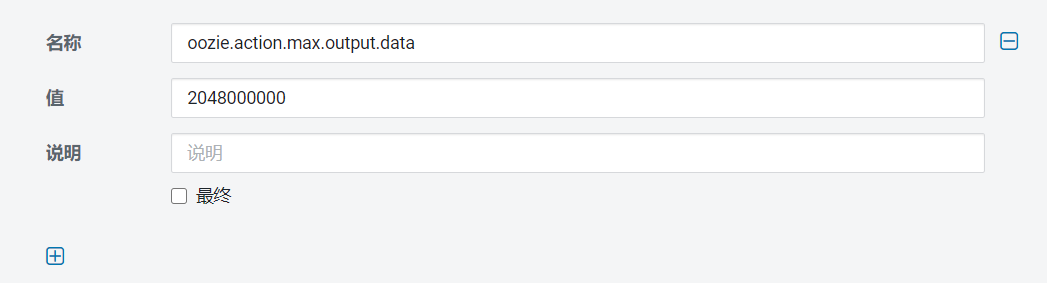
此错误是oozie输出内容的大小受限，并不影响hive执行结果的正确性。



解决：

修改oozie配置oozie-site.xml，添加子配置项：

|  |
| --- |
| Name：oozie.action.max.output.data  Value：2048000000 |



重新运行oozie workflow。

#### 统计分析

增量统计时，同时增加where条件，只统计昨天所在的区间数据（年、月、日、时）。

注意DWS层维度较多， 小时数据和天数据(2019-07-01/2019-07-01 10:00:00)，季度数据和年数据(2019-Q3/2019)，不同维度存在分区目录相同的情况。如果使用的是Insert overwrite覆盖写入，天和小时数据存在于同一个分区中，会出现新数据互相覆盖的问题。

使用Insert Into比较安全。

分区内有旧数据的话，则要先通过sql删除旧数据：

|  |
| --- |
| *-- 精确删除数据 --删除20190901的05点数据* delete from itcast\_dws.visit\_dws where yearinfo='2019' and monthinfo='07' and dayinfo='01' and hourinfo='05' and time\_type=1; *--删除20190901天数据* delete from itcast\_dws.visit\_dws where yearinfo='2019' and monthinfo='09' and dayinfo='01' and time\_type=2; *--删除201909月数据* delete from itcast\_dws.visit\_dws where yearinfo='2019' and monthinfo='09' and dayinfo='-1' and time\_type=3; *--删除2019年3季度数据* delete from itcast\_dws.visit\_dws where yearinfo='2019' and quarterinfo='3' and time\_type=4; *--删除2019年数据* delete from itcast\_dws.visit\_dws where yearinfo='2019' and time\_type=4;  *--删除分区下所有数据 --删除20190901天和小时数据（会把日期里面的小时数据一起删除掉，月和年尽量使用delete语句）* alter table itcast\_dws.visit\_dws drop partition (yearinfo='2019', monthinfo='09', dayinfo='01'); |

##### 地区分组

小时维度：

|  |
| --- |
| *--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;  INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  wce.area as area,  '-1' as seo\_source,  '-1' as origin\_channel,  wce.hourinfo as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo,'-',wce.dayinfo,' ',wce.hourinfo) as time\_str,  '-1' as from\_url,  '1' as groupType,  '1' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  wce.dayinfo as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where ***concat\_ws*('-',wce.yearinfo,wce.monthinfo,wce.dayinfo)='${TD\_DATE}'** group by wce.area,wce.yearinfo,wce.quarterinfo,wce.monthinfo,wce.dayinfo,wce.hourinfo; |

天维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select   *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  wce.area as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo,'-',wce.dayinfo) as time\_str,  '-1' as from\_url,  '1' as groupType,  '2' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  wce.dayinfo as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.monthinfo,wce.dayinfo)='${TD\_DATE}' group by wce.area,wce.yearinfo,wce.quarterinfo,wce.monthinfo,wce.dayinfo; |

月维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  wce.area as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo) as time\_str,  '-1' as from\_url,  '1' as groupType,  '3' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where ***concat\_ws*(' - ', wce.yearinfo, wce.monthinfo) = ' ${V\_Month}'** group by wce.area,wce.yearinfo,wce.quarterinfo,wce.monthinfo; |

季度维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  wce.area as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-Q',wce.quarterinfo) as time\_str,  '-1' as from\_url,  '1' as groupType,  '4' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where ***concat\_ws*('-',wce.yearinfo,wce.quarterinfo)='${V\_QUARTER}'** group by wce.area,wce.yearinfo,wce.quarterinfo; |

年维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  wce.area as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  '-1' as from\_url,  '1' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where **wce.yearinfo='${V\_Year}'** group by wce.area,wce.yearinfo; |

##### 搜索来源分组

小时维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo, monthinfo, dayinfo) select *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '' - 1'' as area,  wce.seo\_source as seo\_source,  ''-1'' as origin\_channel,  wce.hourinfo as hourinfo,  wce.quarterinfo as quarterinfo,  CONCAT(wce.yearinfo,''-'',wce.monthinfo,''-'',wce.dayinfo,'' '',wce.hourinfo) as time\_str,  ''-1'' as from\_url,  ''2'' as groupType,  ''1'' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  wce.dayinfo as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.monthinfo,wce.dayinfo)='${TD\_DATE}' group by wce.seo\_source, wce.yearinfo, wce.quarterinfo, wce.monthinfo, wce.dayinfo, wce.hourinfo; |

天维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  wce.seo\_source as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo,'-',wce.dayinfo) as time\_str,  '-1' as from\_url,  '2' as groupType,  '2' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  wce.dayinfo as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.monthinfo,wce.dayinfo)='${TD\_DATE}' group by wce.seo\_source,wce.yearinfo,wce.quarterinfo,wce.monthinfo,wce.dayinfo; |

月维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  wce.seo\_source as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo) as time\_str,  '-1' as from\_url,  '2' as groupType,  '3' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.monthinfo)='${V\_Month}' group by wce.seo\_source,wce.yearinfo,wce.quarterinfo,wce.monthinfo; |

季度维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  wce.seo\_source as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-Q',wce.quarterinfo) as time\_str,  '-1' as from\_url,  '2' as groupType,  '4' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.quarterinfo)='${V\_QUARTER}' group by wce.seo\_source,wce.yearinfo,wce.quarterinfo; |

年维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  wce.seo\_source as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  '-1' as from\_url,  '2' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where wce.yearinfo='${V\_Year}' group by wce.seo\_source,wce.yearinfo; |

##### 来源渠道分组

小时维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.`visit\_dws` PARTITION (yearinfo,monthinfo,dayinfo)  select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  wce.origin\_channel as origin\_channel,  wce.hourinfo as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo,'-',wce.dayinfo,' ',wce.hourinfo) as time\_str,  '-1' as from\_url,  '3' as groupType,  '1' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  wce.dayinfo as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.monthinfo,wce.dayinfo)='${TD\_DATE}' group by wce.origin\_channel,wce.yearinfo,wce.quarterinfo,wce.monthinfo,wce.dayinfo,wce.hourinfo; |

天维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.`visit\_dws` PARTITION (yearinfo,monthinfo,dayinfo)  select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  wce.origin\_channel as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo,'-',wce.dayinfo) as time\_str,  '-1' as from\_url,  '3' as groupType,  '2' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  wce.dayinfo as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.monthinfo,wce.dayinfo)='${TD\_DATE}' group by wce.origin\_channel,wce.yearinfo,wce.quarterinfo,wce.monthinfo,wce.dayinfo; |

月维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.`visit\_dws` PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  wce.origin\_channel as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo) as time\_str,  '-1' as from\_url,  '3' as groupType,  '3' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.monthinfo)='${V\_Month}' group by wce.origin\_channel,wce.yearinfo,wce.quarterinfo,wce.monthinfo; |

季度维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.`visit\_dws` PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  wce.origin\_channel as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-Q',wce.quarterinfo) as time\_str,  '-1' as from\_url,  '3' as groupType,  '4' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.quarterinfo)='${V\_QUARTER}' group by wce.origin\_channel,wce.yearinfo,wce.quarterinfo; |

年维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.`visit\_dws` PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  wce.origin\_channel as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  '-1' as from\_url,  '3' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd wce where wce.yearinfo='${V\_Year}' group by wce.origin\_channel,wce.yearinfo; |

##### 会话来源页面分组

小时维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.`visit\_dws` PARTITION (yearinfo,monthinfo,dayinfo)  select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  wce.hourinfo as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo,'-',wce.dayinfo,' ',wce.hourinfo) as time\_str,  wce.from\_url as from\_url,  '4' as groupType,  '1' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  wce.dayinfo as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.monthinfo,wce.dayinfo)='${TD\_DATE}' group by wce.from\_url,wce.yearinfo,wce.quarterinfo,wce.monthinfo,wce.dayinfo,wce.hourinfo; |

天维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.`visit\_dws` PARTITION (yearinfo,monthinfo,dayinfo)  select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo,'-',wce.dayinfo) as time\_str,  wce.from\_url as from\_url,  '4' as groupType,  '2' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  wce.dayinfo as dayinfo from itcast\_dwd.visit\_consult\_dwd tablesample(bucket 1 out of 10 on sid) wce where *concat\_ws*('-',wce.yearinfo,wce.monthinfo,wce.dayinfo)='${TD\_DATE}' group by wce.from\_url,wce.yearinfo,wce.quarterinfo,wce.monthinfo,wce.dayinfo; |

月维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.`visit\_dws` PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo) as time\_str,  wce.from\_url as from\_url,  '4' as groupType,  '3' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd tablesample(bucket 1 out of 10 on sid) wce where *concat\_ws*('-',wce.yearinfo,wce.monthinfo)='${V\_Month}' group by wce.from\_url,wce.yearinfo,wce.quarterinfo,wce.monthinfo; |

季度维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.`visit\_dws` PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-Q',wce.quarterinfo) as time\_str,  wce.from\_url as from\_url,  '4' as groupType,  '4' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd tablesample(bucket 1 out of 10 on sid) wce where *concat\_ws*('-',wce.yearinfo,wce.quarterinfo)='${V\_QUARTER}' group by wce.from\_url,wce.yearinfo,wce.quarterinfo; |

年维度：

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.`visit\_dws` PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  wce.from\_url as from\_url,  '4' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd tablesample(bucket 1 out of 10 on sid) wce where wce.yearinfo='${V\_Year}' group by wce.from\_url,wce.yearinfo; |

##### 总访问量

小时（小时段基础数据）

因为小时段数据可以直接sum求和，因此OLAP应用可以在小时数据基础上，进行简单的sum操作以获取到区间小时段数据。

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  wce.area as area,  '-1' as seo\_source,  '-1' as origin\_channel,  wce.hourinfo as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo,'-',wce.dayinfo,'-',wce.hourinfo) as time\_str,  '-1' as from\_url,  '5' as groupType,  '-1' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  wce.dayinfo as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.monthinfo,wce.dayinfo)='${TD\_DATE}' group by wce.yearinfo,wce.quarterinfo,wce.monthinfo,wce.dayinfo,wce.hourinfo; |

天

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo,'-',wce.dayinfo) as time\_str,  '-1' as from\_url,  '5' as groupType,  '-1' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  wce.dayinfo as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.monthinfo,wce.dayinfo)='${TD\_DATE}' group by wce.yearinfo,wce.quarterinfo,wce.monthinfo,wce.dayinfo; |

月

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo) as time\_str,  '-1' as from\_url,  '5' as groupType,  '-1' as time\_type,  wce.yearinfo as yearinfo,  wce.monthinfo as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.monthinfo)='${V\_Month}' group by wce.yearinfo,wce.quarterinfo,wce.monthinfo; |

季度

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  wce.quarterinfo as quarterinfo,  *CONCAT*(wce.yearinfo,'-Q',wce.quarterinfo) as time\_str,  '-1' as from\_url,  '5' as groupType,  '-1' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where *concat\_ws*('-',wce.yearinfo,wce.quarterinfo)='${V\_QUARTER}' group by wce.yearinfo,wce.quarterinfo; |

年

|  |
| --- |
| INSERT INTO TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo) select  *COUNT*(DISTINCT wce.sid) as sid\_total,  *COUNT*(DISTINCT wce.sessionid) as sessionid\_total,  *COUNT*(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  '-1' as from\_url,  '5' as groupType,  '-1' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd wce where wce.yearinfo='${V\_Year}' group by wce.yearinfo; |

##### OOzie Shell

季度值可以通过公式获取，QUARTER=$((**($MONTH-1)/3+1**))

date +%-m，和+%m的区别是，-m会将前缀的0去掉，比如7，而不是07。结果能够作为int类型参与运算。

|  |
| --- |
| #! /bin/bash  HIVE\_HOME=/usr/bin/hive  if [[ $1 == "" ]];then  TD\_DATE=`date -d '1 days ago' "+%Y-%m-%d"`  else  TD\_DATE=$1  fi  TD\_YEAR=$(date -d "$TD\_DATE" +%Y)  TD\_MONTH=$(date -d "$TD\_DATE" +%m)  TD\_DAY=$(date -d "$TD\_DATE" +%d)  month\_for\_quarter=`date --date="$TD\_DATE" +%**-m**`  TD\_QUARTER=$((**($**month\_for\_quarter**-1)/3+1**))  ${HIVE\_HOME} -S -e "  --动态分区配置  set hive.exec.dynamic.partition=true;  set hive.exec.max.dynamic.partitions=2000;  set hive.exec.dynamic.partition.mode=nonstrict;  set hive.exec.max.dynamic.partitions.pernode=10000;  set hive.exec.max.dynamic.partitions=100000;  set hive.exec.max.created.files=150000;  --hive压缩  set hive.exec.compress.intermediate=true;  set hive.exec.compress.output=true;  --写入时压缩生效  set hive.exec.orc.compression.strategy=COMPRESSION;  ---------地区分组  --小时  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  area,  '-1' as seo\_source,  '-1' as origin\_channel,  hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time\_str,  '-1' as from\_url,  '1' as grouptype,  '1' as time\_type,  yearinfo, monthinfo, dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH' and dayinfo='$TD\_DAY'  group by area, yearinfo, quarterinfo, monthinfo, dayinfo, hourinfo;  --天  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo,'-',dayinfo) as time\_str,  '-1' as from\_url,  '1' as grouptype,  '2' as time\_type,  yearinfo, monthinfo, dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH' and dayinfo='$TD\_DAY'  group by area, yearinfo, quarterinfo, monthinfo, dayinfo;  --月  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo) as time\_str,  '-1' as from\_url,  '1' as grouptype,  '3' as time\_type,  yearinfo, monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH'  group by area, yearinfo, quarterinfo, monthinfo;  --季度  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-Q',quarterinfo) as time\_str,  '-1' as from\_url,  '1' as grouptype,  '4' as time\_type,  yearinfo,  '-1' as monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and quarterinfo='$TD\_QUARTER'  group by area, yearinfo, quarterinfo;  --年  INSERT overwrite TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo)  select  COUNT(DISTINCT wce.sid) as sid\_total,  COUNT(DISTINCT wce.session\_id) as sessionid\_total,  COUNT(DISTINCT wce.ip) as ip\_total,  wce.area as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  '-1' as from\_url,  '1' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd wce  where yearinfo='$TD\_YEAR'  group by wce.area,wce.yearinfo;    ---------搜索来源分组  --小时  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  seo\_source,  '-1' as origin\_channel,  hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time\_str,  '-1' as from\_url,  '2' as grouptype,  '1' as time\_type,  yearinfo, monthinfo, dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH' and dayinfo='$TD\_DAY'  group by seo\_source, yearinfo, quarterinfo, monthinfo, dayinfo, hourinfo;  --天  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo,'-',dayinfo) as time\_str,  '-1' as from\_url,  '2' as grouptype,  '2' as time\_type,  yearinfo, monthinfo, dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH' and dayinfo='$TD\_DAY'  group by seo\_source, yearinfo, quarterinfo, monthinfo, dayinfo;  --月  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo) as time\_str,  '-1' as from\_url,  '2' as grouptype,  '3' as time\_type,  yearinfo, monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH'  group by seo\_source, yearinfo, quarterinfo, monthinfo;  --季度  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-Q',quarterinfo) as time\_str,  '-1' as from\_url,  '2' as grouptype,  '4' as time\_type,  yearinfo,  '-1' as monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and quarterinfo='$TD\_QUARTER'  group by seo\_source, yearinfo, quarterinfo;  --年  INSERT overwrite TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo)  select  COUNT(DISTINCT wce.sid) as sid\_total,  COUNT(DISTINCT wce.session\_id) as sessionid\_total,  COUNT(DISTINCT wce.ip) as ip\_total,  '-1' as area,  seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  '-1' as from\_url,  '2' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd wce  where yearinfo='$TD\_YEAR'  group by wce.seo\_source,wce.yearinfo;  ---------来源渠道分组  --小时  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  origin\_channel,  hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time\_str,  '-1' as from\_url,  '3' as grouptype,  '1' as time\_type,  yearinfo, monthinfo, dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH' and dayinfo='$TD\_DAY'  group by origin\_channel, yearinfo, quarterinfo, monthinfo, dayinfo, hourinfo;  --天  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo,'-',dayinfo) as time\_str,  '-1' as from\_url,  '3' as grouptype,  '2' as time\_type,  yearinfo, monthinfo, dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH' and dayinfo='$TD\_DAY'  group by origin\_channel, yearinfo, quarterinfo, monthinfo, dayinfo;  --月  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo) as time\_str,  '-1' as from\_url,  '3' as grouptype,  '3' as time\_type,  yearinfo, monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH'  group by origin\_channel, yearinfo, quarterinfo, monthinfo;  --季度  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-Q',quarterinfo) as time\_str,  '-1' as from\_url,  '3' as grouptype,  '4' as time\_type,  yearinfo,  '-1' as monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and quarterinfo='$TD\_QUARTER'  group by origin\_channel, yearinfo, quarterinfo;  --年  INSERT overwrite TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo)  select  COUNT(DISTINCT wce.sid) as sid\_total,  COUNT(DISTINCT wce.session\_id) as sessionid\_total,  COUNT(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  '-1' as from\_url,  '3' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd wce  where yearinfo='$TD\_YEAR'  group by wce.origin\_channel,wce.yearinfo;    ---------会话来源页面分组  --小时  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time\_str,  from\_url,  '4' as grouptype,  '1' as time\_type,  yearinfo, monthinfo, dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH' and dayinfo='$TD\_DAY'  group by from\_url, yearinfo, quarterinfo, monthinfo, dayinfo, hourinfo;  --天  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo,'-',dayinfo) as time\_str,  from\_url,  '4' as grouptype,  '2' as time\_type,  yearinfo, monthinfo, dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH' and dayinfo='$TD\_DAY'  group by from\_url, yearinfo, quarterinfo, monthinfo, dayinfo;  --月  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo) as time\_str,  from\_url,  '4' as grouptype,  '3' as time\_type,  yearinfo, monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH'  group by from\_url, yearinfo, quarterinfo, monthinfo;  --季度  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-Q',quarterinfo) as time\_str,  from\_url,  '4' as grouptype,  '4' as time\_type,  yearinfo,  '-1' as monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and quarterinfo='$TD\_QUARTER'  group by from\_url, yearinfo, quarterinfo;  --年  INSERT overwrite TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo)  select  COUNT(DISTINCT wce.sid) as sid\_total,  COUNT(DISTINCT wce.session\_id) as sessionid\_total,  COUNT(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  from\_url,  '4' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd wce  where yearinfo='$TD\_YEAR'  group by wce.from\_url,wce.yearinfo;  ---------总访问量分组  --小时  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time\_str,  '-1' as from\_url,  '5' as grouptype,  '1' as time\_type,  yearinfo, monthinfo, dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH' and dayinfo='$TD\_DAY'  group by yearinfo, quarterinfo, monthinfo, dayinfo, hourinfo;  --天  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo,'-',dayinfo) as time\_str,  '-1' as from\_url,  '5' as grouptype,  '2' as time\_type,  yearinfo, monthinfo, dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH' and dayinfo='$TD\_DAY'  group by yearinfo, quarterinfo, monthinfo, dayinfo;  --月  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-',monthinfo) as time\_str,  '-1' as from\_url,  '5' as grouptype,  '3' as time\_type,  yearinfo, monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH'  group by yearinfo, quarterinfo, monthinfo;  --季度  insert into itcast\_dws.visit\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid) as sid\_total,  count(distinct session\_id) as session\_total,  count(distinct ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat(yearinfo,'-Q',quarterinfo) as time\_str,  '-1' as from\_url,  '5' as grouptype,  '4' as time\_type,  yearinfo,  '-1' as monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where yearinfo='$TD\_YEAR' and quarterinfo='$TD\_QUARTER'  group by yearinfo, quarterinfo;  --年  INSERT overwrite TABLE itcast\_dws.visit\_dws PARTITION (yearinfo,monthinfo,dayinfo)  select  COUNT(DISTINCT wce.sid) as sid\_total,  COUNT(DISTINCT wce.session\_id) as sessionid\_total,  COUNT(DISTINCT wce.ip) as ip\_total,  '-1' as area,  '-1' as seo\_source,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  wce.yearinfo as time\_str,  '-1' as from\_url,  '5' as groupType,  '5' as time\_type,  wce.yearinfo as yearinfo,  '-1' as monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd wce  where yearinfo='$TD\_YEAR'  group by wce.yearinfo;  " |

#### 导出数据

增量数据导出mysql，需要做成sh脚本，用oozie定时触发，分析流程完毕后执行导出。

因为每次统计最大的时间范围是年，所以每次导出都需要覆盖导出一年的数据。此时需要先将mysql中的旧数据删除，然后再将新数据导入。

|  |
| --- |
| #! /bin/bash  SQOOP\_HOME=/usr/bin/sqoop  if [[ $1 == "" ]];then  TD\_DATE=`date -d '1 days ago' "+%Y-%m-%d"`  else  TD\_DATE=$1  fi  TD\_YEAR=$(date -d "$TD\_DATE" +%Y)  mysql -uroot -p123456 -h192.168.52.150 -P3306 -e "**delete from** scrm\_bi.itcast\_visit where yearinfo='$TD\_YEAR';"  wait  $SQOOP\_HOME export \  --connect "jdbc:mysql://192.168.52.150:3306/scrm\_bi?useUnicode=true&characterEncoding=utf-8" \  --username root \  --password '123456' \  --table itcast\_visit \  --hcatalog-database itcast\_dws \  --hcatalog-table visit\_dws \  --hcatalog-partition-keys yearinfo \  --hcatalog-partition-values $TD\_YEAR \  -m 100 |

## 咨询客户量实现

### 建模

#### 指标和维度

指标：咨询客户量是单位时间内有效咨询客服的去重后客户数量，以天为单位显示咨询客户。客户与网咨有说一句话的称为有效咨询。

维度：

* 时间维度：年、季度、月、天
* 业务属性维度：地区、来源渠道

#### 原始数据结构

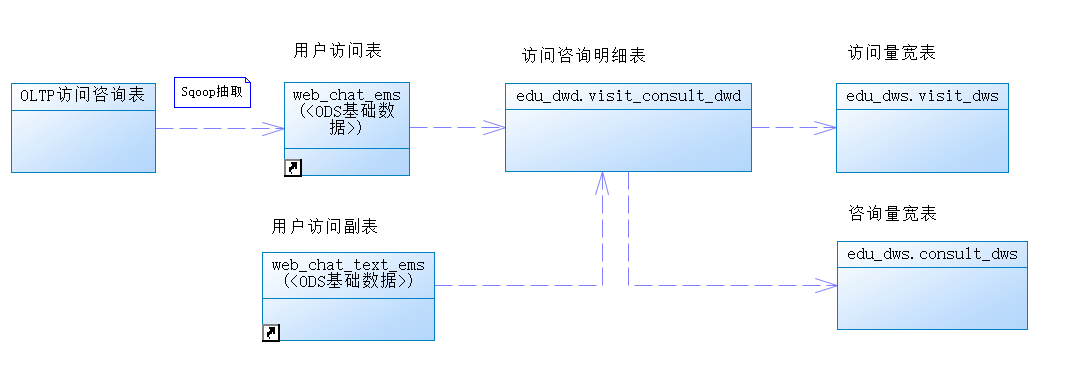
咨询客户量的数据来源和访问客户量一致，都是咨询系统的访问会话信息月表web\_chat\_ems，表名的格式为web\_chat\_ems\_年\_月，年份为4位数字，月份为二位数字，如果为单数时，前面会用0来补全，比如web\_chat\_ems\_2019\_07。

#### 事实表和维度表

事实表就是我们的客户访问表，而维度数据都包含在事实表中，没有需要额外关联的维度表。

#### 分层

ODS层是原始数据，一般不允许修改，所以使用外部表保证数据的安全性，避免误删除；DW和APP层是统计数据，为了使覆盖插入等操作更方便，满足业务需求的同时，提高开发和测试效率，推荐使用内部表。



##### ODS层

由于咨询客户量的原始数据和访问客户量一致，所以ODS层可以直接复用itcast\_ods.web\_chat\_ems表，内容是咨询系统OLTP数据库的web\_chat\_ems\_20XX\_XX等月表中抽取的原始数据；

##### DWD层

可以复用访问客户量指标的DWD层：itcast\_dwd.visit\_consult\_dwd。

##### DWS层

客户访问量指标中，DWS层我们增加了两个标识字段：时间和业务属性。在咨询客户量指标中，我们采用同样的方式来进行统计，最后在APP层或OLAP应用中再做进一步的分组取值。

写入时压缩生效

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

|  |
| --- |
| CREATE TABLE IF NOT EXISTS itcast\_dws.consult\_dws (  sid\_total INT COMMENT '根据sid去重求count',  sessionid\_total INT COMMENT '根据sessionid去重求count',  ip\_total INT COMMENT '根据IP去重求count',  area STRING COMMENT '区域信息',  origin\_channel STRING COMMENT '来源渠道',  hourinfo STRING COMMENT '创建时间，统计至小时',  quarterinfo STRING COMMENT '季度',  time\_str STRING COMMENT '时间明细',  groupType STRING COMMENT '产品属性类型：1.地区；2.来源渠道',  time\_type STRING COMMENT '时间聚合类型：1、按小时聚合；2、按天聚合；3、按月聚合；4、按季度聚合；5、按年聚合；' ) COMMENT '咨询量DWS宽表' PARTITIONED BY (yearinfo string, monthinfo STRING, dayinfo string) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS ORC LOCATION '/user/hive/warehouse/itcast\_dws.db/consult\_dws' TBLPROPERTIES ('orc.compress'='SNAPPY'); |

##### APP层

如果客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

同访问客户量指标。

#### 数据清洗转换

同访问客户量指标。

#### 统计分析

##### 地区分组

天维度：

|  |
| --- |
| *--动态分区配置* set hive.exec.dynamic.partition=true; set hive.exec.max.dynamic.partitions=2000; 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;  *-------地区分组 --天* insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid),  *count*(distinct session\_id),  *count*(distinct ip),  area,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat\_ws*('-',yearinfo,monthinfo,dayinfo),  '1',  '2',  yearinfo,monthinfo,dayinfo from itcast\_dwd.visit\_consult\_dwd **where msg\_count >= 1** group by area, yearinfo, quarterinfo, monthinfo, dayinfo; |

月维度：

|  |
| --- |
| insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid),  *count*(distinct session\_id),  *count*(distinct ip),  area,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat\_ws*('-',yearinfo,monthinfo),  '1',  '3',  yearinfo,monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd where msg\_count >= 1 group by area, yearinfo, quarterinfo, monthinfo; |

季度维度：

|  |
| --- |
| insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid),  *count*(distinct session\_id),  *count*(distinct ip),  area,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat\_ws*('-Q',yearinfo,quarterinfo),  '1',  '4',  yearinfo,'-1' as monthinfo,'-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd where msg\_count >= 1 group by area, yearinfo, quarterinfo; |

年维度：

|  |
| --- |
| insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid),  *count*(distinct session\_id),  *count*(distinct ip),  area,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  yearinfo,  '1',  '5',  yearinfo,'-1' as monthinfo,'-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd where msg\_count >= 1 group by area, yearinfo; |

##### 来源渠道分组

天维度：

|  |
| --- |
| insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid),  *count*(distinct session\_id),  *count*(distinct ip),  '-1' as area,  origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat\_ws*('-',yearinfo,monthinfo,dayinfo),  '2',  '2',  yearinfo,monthinfo,dayinfo from itcast\_dwd.visit\_consult\_dwd where msg\_count >= 1 group by origin\_channel, yearinfo, quarterinfo, monthinfo, dayinfo; |

月维度：

|  |
| --- |
| insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid),  *count*(distinct session\_id),  *count*(distinct ip),  '-1' as area,  origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat\_ws*('-',yearinfo,monthinfo),  '2',  '3',  yearinfo,monthinfo,  '-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd where msg\_count >= 1 group by origin\_channel, yearinfo, quarterinfo, monthinfo; |

季度维度：

|  |
| --- |
| insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid),  *count*(distinct session\_id),  *count*(distinct ip),  '-1' as area,  origin\_channel,  '-1' as hourinfo,  quarterinfo,  *concat\_ws*('-Q',yearinfo,quarterinfo),  '2',  '4',  yearinfo,'-1' as monthinfo,'-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd where msg\_count >= 1 group by origin\_channel, yearinfo, quarterinfo; |

年维度：

|  |
| --- |
| insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo) select  *count*(distinct sid),  *count*(distinct session\_id),  *count*(distinct ip),  '-1' as area,  origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  yearinfo,  '2',  '5',  yearinfo,'-1' as monthinfo,'-1' as dayinfo from itcast\_dwd.visit\_consult\_dwd where msg\_count >= 1 group by origin\_channel, yearinfo; |

#### 导出数据

##### 创建mysql表

|  |
| --- |
| CREATE TABLE `itcast\_consult` (  sid\_total int(11) COMMENT '去重并聚合sid',  sessionid\_total int(11) COMMENT '去重并聚合sessionid',  ip\_total int(11) COMMENT '去重并聚合ip',  area varchar(32) COMMENT '区域信息',  origin\_channel varchar(32) COMMENT '来源渠道',  hourinfo varchar(5) COMMENT '创建时间，统计至小时',  quarterinfo varchar(5) COMMENT '季度',  time\_str varchar(32) COMMENT '时间明细',  groupType varchar(5) COMMENT '产品属性类型：1.地区；2.来源渠道',  time\_type varchar(5) COMMENT '时间聚合类型：1、按小时聚合；2、按天聚合；3、按月聚合；4、按季度聚合；5、按年聚合；',  yearinfo varchar(5) COMMENT '创建时间，统计至年',  monthinfo varchar(5) COMMENT '创建时间，统计至月',  dayinfo varchar(5) COMMENT '创建时间，统计至天' ) COMMENT='客户咨询统计数据'; |

##### 执行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\_consult \  --hcatalog-database itcast\_dws \  --hcatalog-table consult\_dws \  -m 100 |

### 增量流程

#### 数据采集

同访问客户量指标。

#### 数据清洗转换

同访问客户量指标。

#### 统计分析

##### SQL

###### 地区分组

|  |
| --- |
| *--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;  *-- 地区分组 -- 天维度：* INSERT INTO TABLE itcast\_dws.consult\_dws PARTITION (yearinfo,monthinfo,dayinfo) SELECT  *count*(distinct wce.sid),  '-1',  '-1',  wce.area,  wce.quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo,'-',wce.dayinfo),  '2',  '1',  wce.yearinfo,  wce.monthinfo,  wce.dayinfo FROM itcast\_dwd.visit\_consult\_dwd wce where wce.msg\_count >= 1  **and *concat\_ws*('-',wce.yearinfo,wce.monthinfo,wce.dayinfo)='${TD\_DATE}'** group by wce.yearinfo,wce.quarterinfo,wce.monthinfo,wce.dayinfo,wce.area; *-- 月维度：* INSERT INTO TABLE itcast\_dws.consult\_dws PARTITION (yearinfo,monthinfo,dayinfo) SELECT  *count*(distinct wce.sid),  '-1',  '-1',  wce.area,  wce.quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo),  '3',  '1',  wce.yearinfo,  wce.monthinfo,  '-1' FROM itcast\_dwd.visit\_consult\_dwd wce where wce.msg\_count >= 1  and *concat\_ws*(' - ', wce.yearinfo, wce.monthinfo) = ' ${V\_Month}' group by wce.yearinfo,wce.quarterinfo,wce.monthinfo,wce.area; *-- 季度维度：* INSERT INTO TABLE itcast\_dws.consult\_dws PARTITION (yearinfo,monthinfo,dayinfo) SELECT  *count*(distinct wce.sid),  '-1',  '-1',  wce.area,  wce.quarterinfo,  *CONCAT*(wce.yearinfo,'-Q',wce.quarterinfo),  '4',  '1',  wce.yearinfo,  '-1',  '-1' FROM itcast\_dwd.visit\_consult\_dwd wce where wce.msg\_count >= 1  and *concat\_ws*('-',wce.yearinfo,wce.quarterinfo)='${V\_QUARTER}' group by wce.yearinfo,wce.quarterinfo,wce.area; *-- 年维度：* INSERT INTO TABLE itcast\_dws.consult\_dws PARTITION (yearinfo,monthinfo,dayinfo) SELECT  *count*(distinct wce.sid),  '-1',  '-1',  wce.area,  '-1',  wce.yearinfo,  '5',  '1',  wce.yearinfo,  '-1',  '-1' FROM itcast\_dwd.visit\_consult\_dwd wce where wce.msg\_count >= 1  and wce.yearinfo='${V\_Year}' group by wce.yearinfo,wce.area; |

###### 来源渠道分组

|  |
| --- |
| *--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;  *--来源渠道分组 --天维度：* INSERT INTO TABLE itcast\_dws.consult\_dws PARTITION (yearinfo,monthinfo,dayinfo) SELECT  *count*(distinct wce.sid),  '-1',  wce.origin\_channel,  '-1',  wce.quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo,'-',wce.dayinfo),  '2',  '2',  wce.yearinfo,  wce.monthinfo,  wce.dayinfo FROM itcast\_dwd.visit\_consult\_dwd wce where wce.msg\_count >= 1  and *concat\_ws*('-',wce.yearinfo,wce.monthinfo,wce.dayinfo)='${TD\_DATE}' group by wce.yearinfo,wce.quarterinfo,wce.monthinfo,wce.dayinfo,wce.origin\_channel; *--月维度：* INSERT INTO TABLE itcast\_dws.consult\_dws PARTITION (yearinfo,monthinfo,dayinfo) SELECT  *count*(distinct wce.sid),  '-1',  wce.origin\_channel,  '-1',  wce.quarterinfo,  *CONCAT*(wce.yearinfo,'-',wce.monthinfo),  '3',  '2',  wce.yearinfo,  wce.monthinfo,  '-1' FROM itcast\_dwd.visit\_consult\_dwd wce where wce.msg\_count >= 1  and *concat\_ws*(' - ', wce.yearinfo, wce.monthinfo) = ' ${V\_Month}' group by wce.yearinfo,wce.quarterinfo,wce.monthinfo,wce.origin\_channel; *--季度维度：* INSERT INTO TABLE itcast\_dws.consult\_dws PARTITION (yearinfo,monthinfo,dayinfo) SELECT  *count*(distinct wce.sid),  '-1',  wce.origin\_channel,  '-1',  wce.quarterinfo,  *CONCAT*(wce.yearinfo,'-Q',wce.quarterinfo),  '4',  '2',  wce.yearinfo,  '-1',  '-1' FROM itcast\_dwd.visit\_consult\_dwd wce where wce.msg\_count >= 1  and *concat\_ws*('-',wce.yearinfo,wce.quarterinfo)='${V\_QUARTER}' group by wce.yearinfo,wce.quarterinfo,wce.origin\_channel; *--年维度：* INSERT INTO TABLE itcast\_dws.consult\_dws PARTITION (yearinfo,monthinfo,dayinfo) SELECT  *count*(distinct wce.sid),  '-1',  wce.origin\_channel,  '-1',  '-1',  wce.yearinfo,  '5',  '2',  wce.yearinfo,  '-1',  '-1' FROM itcast\_dwd.visit\_consult\_dwd wce where wce.msg\_count >= 1  and wce.yearinfo='${V\_Year}' group by wce.yearinfo,wce.origin\_channel; |

##### OOzie Shell脚本示例

|  |
| --- |
| #! /bin/bash  HIVE\_HOME=/usr/bin/hive  if [[ $1 == "" ]];then  TD\_DATE=`date -d '1 days ago' "+%Y-%m-%d"`  else  TD\_DATE=$1  fi  TD\_YEAR=$(date -d "$TD\_DATE" +%Y)  TD\_MONTH=$(date -d "$TD\_DATE" +%m)  TD\_DAY=$(date -d "$TD\_DATE" +%d)  TD\_QUARTER=$((($TD\_MONTH-1)/3+1))  $HIVE\_HOME -S -e "  --动态分区配置  set hive.exec.dynamic.partition=true;  set hive.exec.max.dynamic.partitions=2000;  set hive.exec.dynamic.partition.mode=nonstrict;  set hive.exec.max.dynamic.partitions.pernode=10000;  set hive.exec.max.dynamic.partitions=100000;  set hive.exec.max.created.files=150000;  --hive压缩  set hive.exec.compress.intermediate=true;  set hive.exec.compress.output=true;  --写入时压缩生效  set hive.exec.orc.compression.strategy=COMPRESSION;  -------地区分组  --天  insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid),  count(distinct session\_id),  count(distinct ip),  area,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat\_ws('-',yearinfo,monthinfo,dayinfo),  '1',  '2',  yearinfo,monthinfo,dayinfo  from itcast\_dwd.visit\_consult\_dwd  where msg\_count >= 1 and yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH' and dayinfo = '$TD\_DAY'  group by area, yearinfo, quarterinfo, monthinfo, dayinfo;  --月  insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid),  count(distinct session\_id),  count(distinct ip),  area,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat\_ws('-',yearinfo,monthinfo),  '1',  '3',  yearinfo,monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where msg\_count >= 1 and yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH'  group by area, yearinfo, quarterinfo, monthinfo;  --季度  insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid),  count(distinct session\_id),  count(distinct ip),  area,  '-1' as origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat\_ws('-Q',yearinfo,quarterinfo),  '1',  '4',  yearinfo,'-1' as monthinfo,'-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where msg\_count >= 1 and yearinfo='$TD\_YEAR' and quarterinfo='$TD\_QUARTER'  group by area, yearinfo, quarterinfo;  --年  insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid),  count(distinct session\_id),  count(distinct ip),  area,  '-1' as origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  yearinfo,  '1',  '5',  yearinfo,'-1' as monthinfo,'-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where msg\_count >= 1 and yearinfo='$TD\_YEAR'  group by area, yearinfo;    ------来源渠道分组  --天  insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid),  count(distinct session\_id),  count(distinct ip),  '-1' as area,  origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat\_ws('-',yearinfo,monthinfo,dayinfo),  '2',  '2',  yearinfo,monthinfo,dayinfo  from itcast\_dwd.visit\_consult\_dwd  where msg\_count >= 1 and yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH' and dayinfo = '$TD\_DAY'  group by origin\_channel, yearinfo, quarterinfo, monthinfo, dayinfo;  --月  insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid),  count(distinct session\_id),  count(distinct ip),  '-1' as area,  origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat\_ws('-',yearinfo,monthinfo),  '2',  '3',  yearinfo,monthinfo,  '-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where msg\_count >= 1 and yearinfo='$TD\_YEAR' and monthinfo='$TD\_MONTH'  group by origin\_channel, yearinfo, quarterinfo, monthinfo;  --季度  insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid),  count(distinct session\_id),  count(distinct ip),  '-1' as area,  origin\_channel,  '-1' as hourinfo,  quarterinfo,  concat\_ws('-Q',yearinfo,quarterinfo),  '2',  '4',  yearinfo,'-1' as monthinfo,'-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where msg\_count >= 1 and yearinfo='$TD\_YEAR' and quarterinfo='$TD\_QUARTER'  group by origin\_channel, yearinfo, quarterinfo;  --年  insert into itcast\_dws.consult\_dws partition (yearinfo, monthinfo, dayinfo)  select  count(distinct sid),  count(distinct session\_id),  count(distinct ip),  '-1' as area,  origin\_channel,  '-1' as hourinfo,  '-1' as quarterinfo,  yearinfo,  '2',  '5',  yearinfo,'-1' as monthinfo,'-1' as dayinfo  from itcast\_dwd.visit\_consult\_dwd  where msg\_count >= 1 and yearinfo='$TD\_YEAR'  group by origin\_channel, yearinfo;  " |

#### 导出数据

##### 导出到mysql

增量数据导出mysql，需要做成sh脚本，用oozie定时触发，在整个流程分析完毕后执行。

因为每次统计最大的时间范围是年，所以每次导出都需要覆盖导出一年的数据。此时需要先将mysql中的旧数据删除，然后再将新数据导入。

|  |
| --- |
| #! /bin/bash  SQOOP\_HOME=/usr/bin/sqoop  if [[ $1 == "" ]];then  TD\_DATE=`date -d '1 days ago' "+%Y-%m-%d"`  else  TD\_DATE=$1  fi  TD\_YEAR=$(date -d "$TD\_DATE" +%Y)  mysql -uroot -p123456 -h192.168.52.150 -P3306 -e "  **delete from** scrm\_bi.itcast\_consult where yearinfo='$TD\_YEAR';  "  sqoop export \  --connect "jdbc:mysql://192.168.52.150:3306/scrm\_bi?useUnicode=true&characterEncoding=utf-8" \  --username root \  --password 123456 \  --table itcast\_consult \  --hcatalog-database itcast\_dws \  --hcatalog-table consult\_dws \  --hcatalog-partition-keys yearinfo \  --hcatalog-partition-values $TD\_YEAR \  -m 100 |

##### oozie调度

略。