# 大数据项目“用户画像”

# 一、项目学习目标：

* 理解网站用户画像分析的业务背景
* 理解网站用户画像分析中常用分析指标的业务含义
* 掌握网站用户画像分析系统的技术架构
* 掌握网站用户画像分析系统中各环节的技术实现
* 能独立设计完成一个初步的网站用户画像分析系统

|  |
| --- |
| 网站用户画像分析，业务知识，推荐书籍：  《网站分析实战——如何以数据驱动决策，提升网站价值》王彦平，吴盛锋编著 |

# 二、项目业务背景

## 2.1 什么是用户画像

### 2.1.1 WEB访问日志

**即指用户访问网站时的所有访问、浏览、点击行为数据。**

**比如点击了哪一个链接，在哪个网页停留时间最多，采用了哪个搜索项、总体浏览时间等。而所有这些信息都可被保存在网站日志中。通过分析这些数据，可以获知许多对网站运营至关重要的信息。采集的数据越全面，分析就能越精准。**

### 2.1.2 日志的生成渠道：

**1）是网站的web服务器所记录的web访问日志；**

**2）是通过在页面嵌入自定义的js代码来获取用户的所有访问行为（比如鼠标悬停的位置，点击的页面组件等），然后通过ajax请求到后台记录日志；这种方式所能采集的信息最全面；**

**3）通过在页面上埋点1像素的图片，将相关页面访问信息请求到后台记录日志；**

### 2.1.3 日志数据内容详述：

**在实际操作中，有以下几个方面的数据可以被采集：**

1. **访客的系统属性特征。比如所采用的操作系统、浏览器、域名和访问速度等。**
2. **访问特征。包括停留时间、点击的URL等。**
3. **来源特征。包括网络内容信息类型、内容分类和来访URL等。**
4. **产品特征。包括所访问的产品编号、产品类别、产品颜色、产品价格、产品利润、产品数量和特价等级等。**

**以电商某东为例，其点击日志格式如下：**

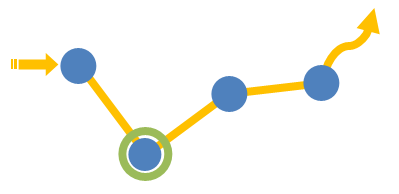
|  |
| --- |
| ***GET /log.gif?t=item.010001&m=UA-J2011-1&pin=-&uid=1679790178&sid=1679790178|12&v=je=1$sc=24-bit$sr=1600x900$ul=zh-cn$cs=GBK$dt=【云南白药套装】云南白药 牙膏 180g×3 （留兰香型）【行情 报价 价格 评测】-京东$hn=item.jd.com$fl=16.0 r0$os=win$br=chrome$bv=39.0.2171.95$wb=1437269412$xb=1449548587$yb=1456186252$zb=12$cb=4$usc=direct$ucp=-$umd=none$uct=-$ct=1456186505411$lt=0$tad=-$sku=1326523$cid1=1316$cid2=1384$cid3=1405$brand=20583$pinid=-&ref=&rm=1456186505411 HTTP/1.1*** |

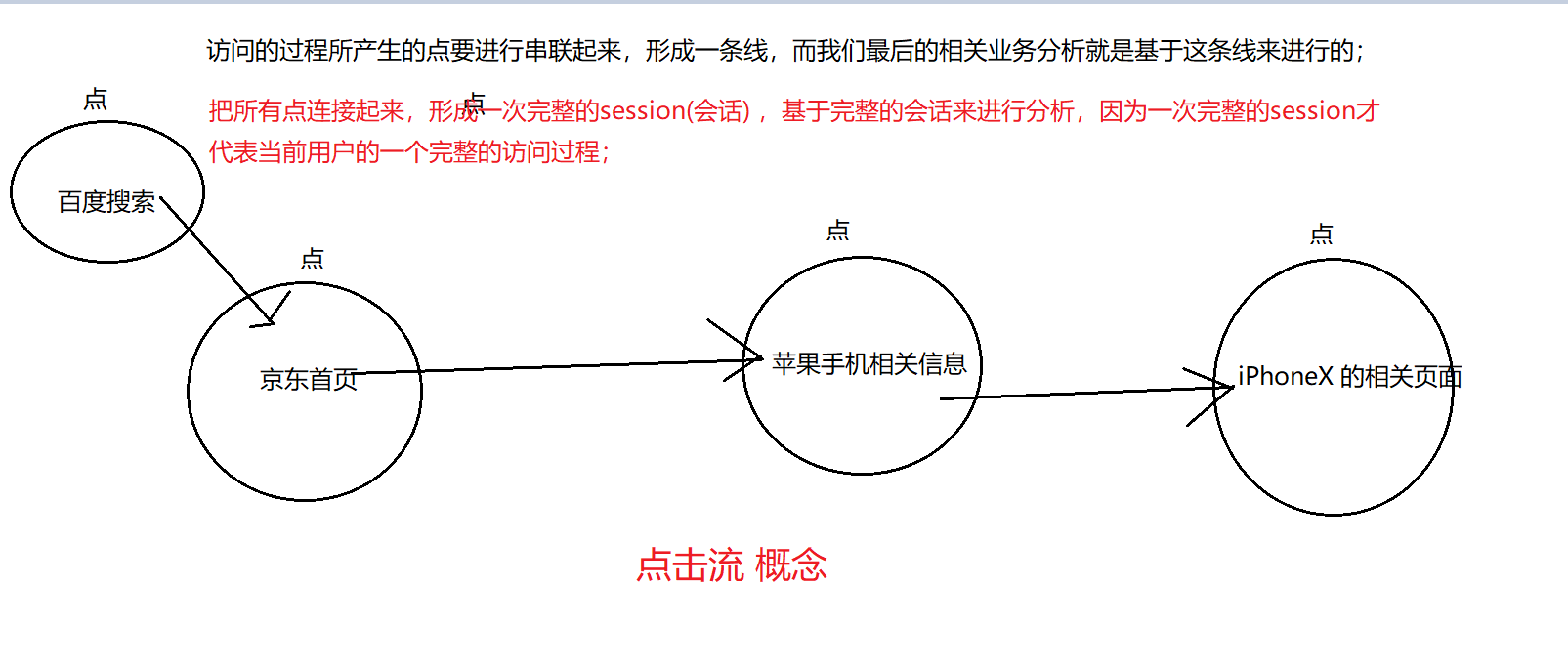
## 2.2 用户画像模型

### 2.2.1“用户画像”概念

**用户画像这个概念更注重用户浏览网站的整个流程，网站日志中记录的用户点击就像是图上的“点”，而用户画像更像是将这些“点”串起来形成的“线”。也可以把“点”认为是网站的Page，而“线”则是访问网站的Session。所以用户画像是由网站日志中整理得到的，它可以比网站日志包含更多的信息，从而使基于用户画像统计得到的结果更加丰富和高效。**

**将各个散乱的点，梳理成各个相同session的数据，这样就可以根据具体的业务来统计分析；**

****



### 2.2.2“用户画像”模型生成

**用户画像在具体操作上是由散点状的点击日志数据梳理所得，从而，点击数据在数据建模时应该存在两张模型表（Pageviews和visits）：**

1. **用于生成用户画像的访问日志表（散乱的一些信息，然后基于该表来梳理出各个业务模型）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *时间戳* | *IP地址* | *Cookie* | *Session* | *请求URL* | *Referal* |
| *2012-01-01 12:31:12* | *101.0.0.1* | *User01* | *S001* | */a/...* | *somesite.com* |
| *2012-01-01 12:31:16* | *201.0.0.2* | *User02* | *S002* | */a/...* | *-* |
| *2012-01-01 12:33:06* | *101.0.0.2* | *User03* | *S002* | */b/...* | *baidu.com* |
| *2012-01-01 15:16:39* | *234.0.0.3* | *User01* | *S003* | */c/...* | *google.com* |
| *2012-01-01 15:17:11* | *101.0.0.1* | *User01* | *S004* | */d/...* | */c/...* |
| *2012-01-01 15:19:23* | *101.0.0.1* | *User01* | *S004* | */e/...* | */d/....* |

1. **页面用户画像模型Pageviews表(按session聚集（同一session）的访问页面信息)(每个session中的每个url也即是访问页面,的记录信息, 相差半个小时了就认为是下一个session了)**

**表示一个页面的信息：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Session* | *userid* | *时间* | *访问页面URL* | *停留时长* | *第几步* |
| *S001* | *User01* | *2012-01-01 12:31:12* | */a/....* | *30* | *1* |
| *S002* | *User02* | *2012-01-01 12:31:16* | */a/....* | *10* | *1* |
| *S002* | *User02* | *2012-01-01 12:33:06* | */b/....* | *110* | *2* |
| *S002* | *User02* | *2012-01-01 12:35:06* | */e/....* | *30* | *3* |

根据业务分析，设计出Pageviews模型：PageViewsBean

|  |
| --- |
| *private String session; //会话id*  *private String remote\_addr; //记录客户端的ip地址*  *private String timestr; //时间*  *private String request; //记录请求的url、http协议*  *private int step; //访问页面数*  *private String staylong; //停留*  *private String referal; //来源url*  *private String useragent; //浏览器信息*  *private String bytes\_send; /发送内容的大小*  *private String status; /状态码* |

1. **用户画像模型Visits表（以相同session为Key,发送给reduce,reduce会根据序号来判断哪个是第一个页面。。。）**

**表示一个session下的所有信息；**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Session* | *起始时间* | *结束时间* | *进入页面* | *离开页面* | *访问页面数* | *IP* | *cookie* | *referal* |
| *S001* | *2012-01-01 12:31:12* | *2012-01-01 12:31:12* | */a/...* | */a/...* | *1* | *101.0.0.1* | *User01* | *somesite.com* |
| *S002* | *2012-01-01 12:31:16* | *2012-01-01 12:35:06* | */a/...* | */e/...* | *3* | *201.0.0.2* | *User02* | *-* |
| *S003* | *2012-01-01 12:35:42* | *2012-01-01 12:35:42* | */c/...* | */c/...* | *1* | *234.0.0.3* | *User03* | *baidu.com* |
| *S004* | *2012-01-01 15:16:39* | *2012-01-01 15:19:23* | */c/...* | */e/...* | *3* | *101.0.0.1* | *User01* | *google.com* |
| *……* | *……* | *……* | *……* | *……* | *……* | *……* | *……* | *……* |

Visits模型：VisitBean

|  |
| --- |
| *private String session; //用户session*  *private String remote\_addr; //userIp*  *private String inTime; //起始时间*  *private String outTime; //结束时间*  *private String inPage; //起始页面*  *private String outPage; //结束页面*  *private String referal; //来访的url*  *private int pageVisits; //访问的总页数* |

这就是用户画像模型，当WEB日志转化成用户画像的时候，很多网站分析度量的计算变得简单了，这就是用户画像的“魔力”所在。基于用户画像我们可以统计出许多常见的网站分析度量。

## 2.3 网站流量数据分析的意义

**网站流量统计分析，可以帮助网站管理员、运营人员、推广人员等实时获取网站流量信息，并从流量来源、网站内容、网站访客特性等多方面提供网站分析的数据依据。从而帮助提高网站流量，提升网站用户体验，让访客更多的沉淀下来变成会员或客户，通过更少的投入获取最大化的收入。**

**如下表：**

|  |  |  |
| --- | --- | --- |
| *网站的眼睛* | *网站的神经* | *网站的大脑* |
| *访问者来自哪里？*  *访问者在寻找什么？*  *哪些页面最受欢迎？*  *访问者从哪里进入？* | *网页布局合理吗？*  *网站导航清晰吗？*  *哪些功能存在问题*  *网站内容有效吗*  *转化路径靠谱吗？* | *如何分解目标？*  *如何分配广告预算？*  *如何衡量产品表现？*  *哪些产品需要优化？*  *哪些指标需要关注？* |

### 2.3.1 用户画像分析的意义可分为两大方面：

**1、技术上：**

**可以合理修改网站结构及适度分配资源，构建后台服务器群组：**

**1）辅助改进网络的拓扑设计，提高性能；**

**2）在有高度相关性的节点之间安排快速有效的访问路径；**

**3）帮助企业更好地设计网站主页和安排网页内容；**

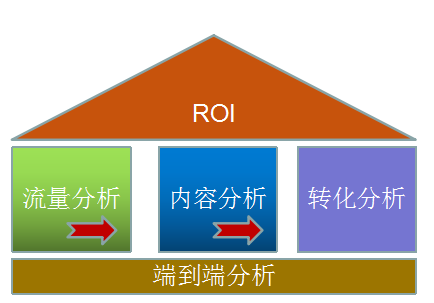
**2、业务上：**

1. **帮助企业改善市场营销决策，如把广告放在适当的Web页面上。**
2. **优化页面及业务流程设计，提高流量转化率。**
3. **帮助企业更好地根据客户的兴趣来安排内容。**
4. **帮助企业对客户群进行细分，针对不同客户制定个性化的促销策略等。**

终极目标是：改善网站(电商、社交、电影、小说)的运营，获取更高投资回报率（ROI）

## 2.4 如何进行网站流量分析

**流量分析整体来说是一个内涵非常丰富的体系，其整体过程是一个金字塔结构：**

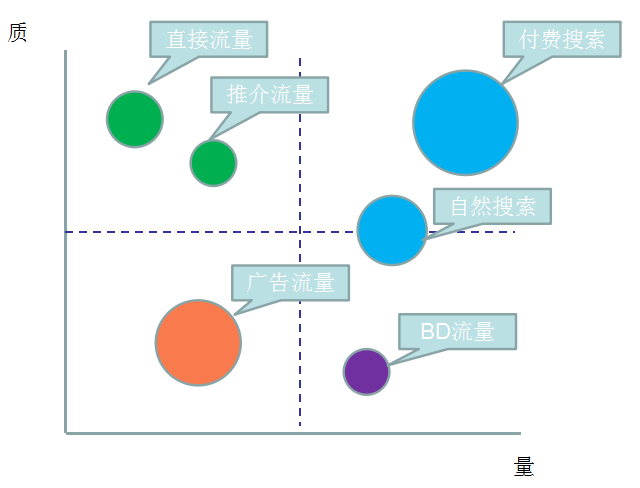
****

### 2.4.1 流量分析模型举例

**通常有以下几大类的分析需求：**

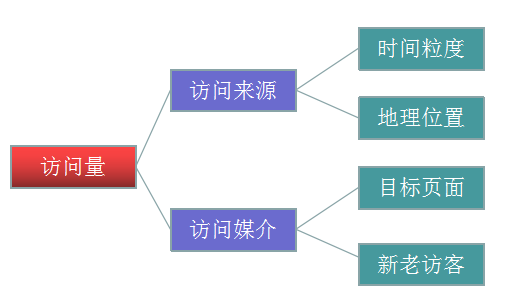
#### 2.4.1.1 网站流量质量分析

**流量对于每个网站来说都是很重要，但流量并不是越多越好，应该更加看重流量的质量，换句话来说就是流量可以为我们带来多少收入。**

****

#### 2.4.1.2 网站流量多维度细分

**细分是指通过不同维度对指标进行分割，查看同一个指标在不同维度下的表现，进而找出有问题的那部分指标，对这部分指标进行优化。**

****

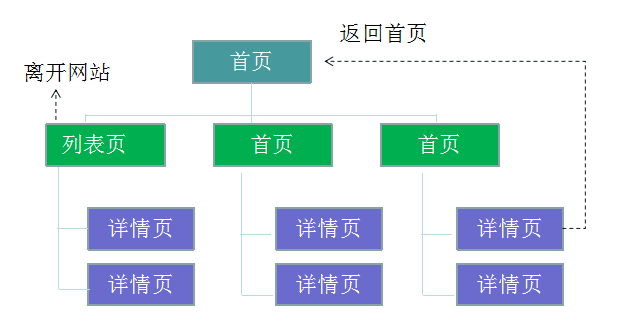
#### 2.4.1.3 网站内容及导航分析

**对于所有网站来说，页面都可以被划分为三个类别：**

* **导航页**
* **功能页**
* **内容页**

**首页和列表页都是典型的导航页，站内搜索页面、注册表单页面和购物车页面都是典型的功能页，而产品详情页、新闻和文章页都是典型的内容页。**

**比如从内容导航分析中，以下两类行为就是网站运营者不希望看到的行为：**

****

**第一个问题：访问者从导航页进入，在还没有看到内容页面之前就从导航页离开网站，需要分析导航页造成访问者中途离开的原因。**

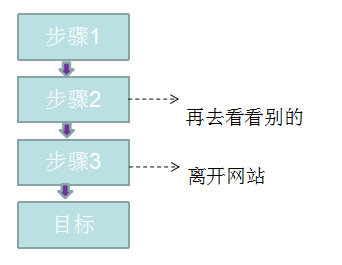
**第二个问题：访问者从导航页进入内容页后，又返回到导航页，说明需要分析内容页的最初设计，并考虑中内容页提供交叉的信息推荐；**

#### 2.4.1.4 网站转化及漏斗分析

**所谓转化，即网站业务流程中的一个封闭渠道，引导用户按照流程最终实现业务目标（比如商品成交）；而漏斗模型则是指进入渠道的用户在各环节递进过程中逐渐流失的形象描述；**

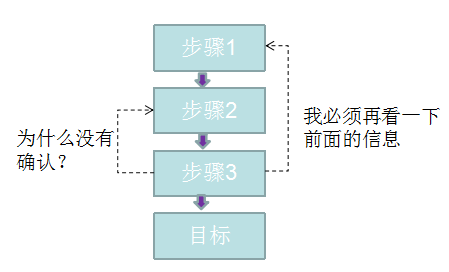
**对于转化渠道，主要进行两部分的分析：访问者的流失和迷失**

1. **阻力和流失**

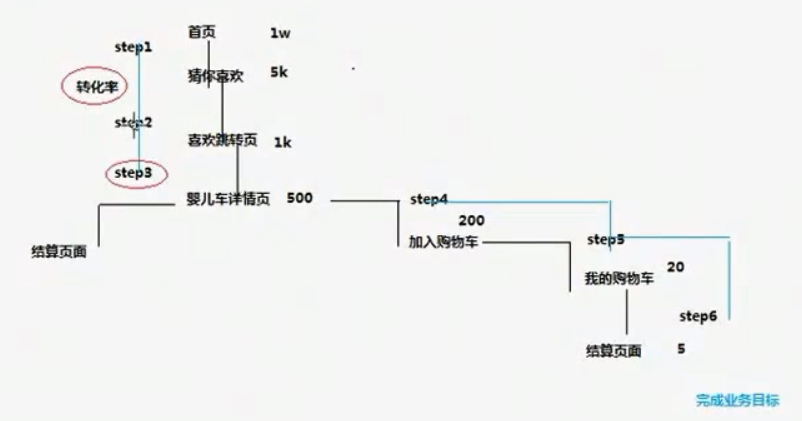
****

**造成流失的原因很多，如：不恰当的商品或活动推荐、对支付环节中专业名词的解释、帮助信息等内容不当；**

1. **迷失**

****

**造成迷失的主要原因是转化流量设计不合理，访问者在特定阶段得不到需要的信息，并且不能根据现有的信息作出决策；**

****

**一个用户从进入网站到最后的转化路径图，每个环节都可以求相应的转化率**

**总之，网站流量分析是一门内容非常丰富的学科，本课程中主要关注网站分析过程中的技术运用，更多关于网站流量分析的业务知识可学习推荐资料。**

## 2.5 流量分析常见指标：

**课程中涉及的分析指标主要位于以下几大方面：**

### **2.5.1 基础分析（PV,IP,UV）**

#### 趋势分析：

**根据选定的时段，提供网站流量数据，通过流量趋势变化形态，为您分析网站访客的访问规律、网站发展状况提供参考。**

#### 对比分析：

**根据选定的两个对比时段，提供网站流量在时间上的纵向对比报表，帮您发现网站发展状况、发展规律、流量变化率等。**

#### 当前在线：

**提供当前时刻站点上的访客量，以及最近15分钟流量、来源、受访、访客变化情况等，方便用户及时了解当前网站流量状况。**

#### 访问明细：

**提供最近7日的访客访问记录，可按每个PV或每次访问行为（访客的每次会话）显示，并可按照来源、搜索词等条件进行筛选。 通过访问明细，用户可以详细了解网站流量的累计过程，从而为用户快速找出流量变动原因提供最原始、最准确的依据。**

**IP 和UV类似，UV更准确，因为当在一个公司内，对外访问时，显示的是同一个公网ip（路由器Ip），只有当日志中没有用户标识时，才使用ip来作为指标.**

### 2.5.2 来源分析

#### 来源分类：

**提供不同来源形式（直接输入、搜索引擎、其他外部链接、站内来源）、不同来源项引入流量的比例情况。通过精确的量化数据，帮助用户分析什么类型的来路产生的流量多、效果好，进而合理优化推广方案。**

#### 搜索引擎：

**提供各搜索引擎以及搜索引擎子产品引入流量的比例情况。从搜索引擎引入流量的的角度，帮助用户了解网站的SEO、SEM效果，从而为制定下一步SEO、SEM计划提供依据。**

#### 搜索词：

**提供访客通过搜索引擎进入网站所使用的搜索词，以及各搜索词引入流量的特征和分布。帮助用户了解各搜索词引入流量的质量，进而了解访客的兴趣关注点、网站与访客兴趣点的匹配度，为优化SEO 方案及SEM提词方案提供详细依据。**

#### 最近7日的访客搜索记录：

**可按每个PV或每次访问行为（访客的每次会话）显示，并可按照访客类型、地区等条件进行筛选。为您搜索引擎优化提供最详细的原始数据。**

#### 来路域名：

**提供具体来路域名引入流量的分布情况，并可按“社会化媒体”、“搜索引擎”、“邮箱”等网站类型对来源域名进行分类。 帮助用户了解哪类推广渠道产生的流量多、效果好，进而合理优化网站推广方案。**

#### 来路页面：

**提供具体来路页面引入流量的分布情况。尤其对于通过流量置换、包广告位等方式从其他网站引入流量的用户，该功能可以方便、清晰地展现广告引入的流量及效果，为优化推广方案提供依据。**

#### 来源升降榜：

**提供开通统计后任意两日的TOP10000搜索词、来路域名引入流量的对比情况，并按照变化的剧烈程度提供排行榜。 用户可通过此功能快速找到哪些来路对网站流量的影响比较大，从而及时排查相应来路问题。**

### 2.5.3 受访分析

#### 受访域名：

**提供访客对网站中各个域名的访问情况。 一般情况下，网站不同域名提供的产品、内容各有差异，通过此功能用户可以了解不同内容的受欢迎程度以及网站运营成效。**

#### 受访页面：

**提供访客对网站中各个页面的访问情况。 站内入口页面为访客进入网站时浏览的第一个页面，如果入口页面的跳出率较高则需要关注并优化；站内出口页面为访客访问网站的最后一个页面，对于离开率较高的页面需要关注并优化。**

#### 受访升降榜：

**提供开通统计后任意两日的TOP10000受访页面的浏览情况对比，并按照变化的剧烈程度提供排行榜。 可通过此功能验证经过改版的页面是否有流量提升或哪些页面有巨大流量波动，从而及时排查相应问题。**

#### 热点图：

**记录访客在页面上的鼠标点击行为，通过颜色区分不同区域的点击热度；支持将一组页面设置为"关 注范围"，并可按来路细分点击热度。 通过访客在页面上的点击量统计，可以了解页面设计是否合理、广告位的安排能否获取更多佣金等。**

#### 用户视点：

**提供受访页面对页面上链接的其他站内页面的输出流量，并通过输出流量的高低绘制热度图，与热点图不同的是，所有记录都是实际打开了下一页面产生了浏览次数（PV）的数据，而不仅仅是拥有鼠标点击行为。**

#### 访问轨迹：

**提供观察焦点页面的上下游页面，了解访客从哪些途径进入页面，又流向了哪里。 通过上游页面列表比较出不同流量引入渠道的效果；通过下游页面列表了解用户的浏览习惯，哪些页面元素、内容更吸引访客点击。**

### 2.5.4 访客分析

#### 地区运营商：

**提供各地区访客、各网络运营商访客的访问情况分布。 地方网站、下载站等与地域性、网络链路等结合较为紧密的网站，可以参考此功能数据，合理优化推广运营方案。**

#### 终端详情：

**提供网站访客所使用的浏览终端的配置情况。参考此数据进行网页设计、开发，可更好地提高网站兼容性，以达到良好的用户交互体验。**

#### 新老访客：

**当日访客中，历史上第一次访问该网站的访客记为当日新访客；历史上已经访问过该网站的访客记为老访客。新访客与老访客进入网站的途径和浏览行为往往存在差异。该功能可以辅助分析不同访客的行为习惯，针对不同访客优化网站，例如为制作新手导航提供数据支持等。**

#### 忠诚度：

**从访客一天内回访网站的次数（日访问频度）与访客上次访问网站的时间两个角度，分析访客对网站的访问粘性、忠诚度、吸引程度。 由于提升网站内容的更新频率、增强用户体验与用户价值可以有更高的忠诚度，因此该功能在网站内容更新及用户体验方面提供了重要参考。**

#### 活跃度：

**从访客单次访问浏览网站的时间与网页数两个角度，分析访客在网站上的活跃程度。 由于提升网站内容的质量与数量可以获得更高的活跃度，因此该功能是网站内容分析的关键指标之一。**

### 2.5.5 转化路径分析

#### 2.5.5.1 转化定义

**·访客在您的网站完成了某项您期望的活动，记为一次转化，如注册或下载。**

#### 2.5.5.2 目标示例：

**·获得用户目标：在线注册、创建账号等。**

**·咨询目标：咨询、留言、电话等。**

**·互动目标：视频播放、加入购物车、分享等。**

**·收入目标：在线订单、付款等。**

#### 2.5.5.3 转化数据的应用：

·在报告的自定义指标中勾选转化指标，实时掌握网站的推广及运营情况。

·结合“全部来源”、“转化路径”、“页面上下游”等报告分析访问漏斗，提高转化率。

·对“转化目标”设置价值，预估转化收益，衡量ROI。

#### 2.5.5.4 路径分析：

根据设置的特定路线，监测某一流程的完成转化情况，算出每步的转换率和流失率数据，如注册流程，购买流程等。

#### 2.5.5.5 转化类型：

****

页面

****

**事件**

# 三、整体技术流程及架构

## 3.1 数据处理流程：

**该项目是一个纯粹的数据分析项目，其整体流程基本上就是依据数据的处理流程进行，依此有以下几个大的步骤：**

### 3.1.1 数据采集

**首先，通过页面嵌入JS代码的方式获取用户访问行为，并发送到web服务的后台记录日志。然后，将各服务器上生成的用户画像日志通过实时或批量的方式汇聚到HDFS文件系统中。**

**当然，一个综合分析系统，数据源可能不仅包含用户画像，还有数据库中的业务数据（如用户信息、商品信息、订单信息等）及对分析有益的外部数据。**

### 3.1.2 数据预处理

**通过mapreduce程序对采集到的用户画像进行预处理，比如清洗，格式整理，滤除脏数据等。**

### 3.1.3 数据入库

**将预处理之后的数据导入到HIVE仓库中相应的库和表中；**

### 3.1.4 数据分析

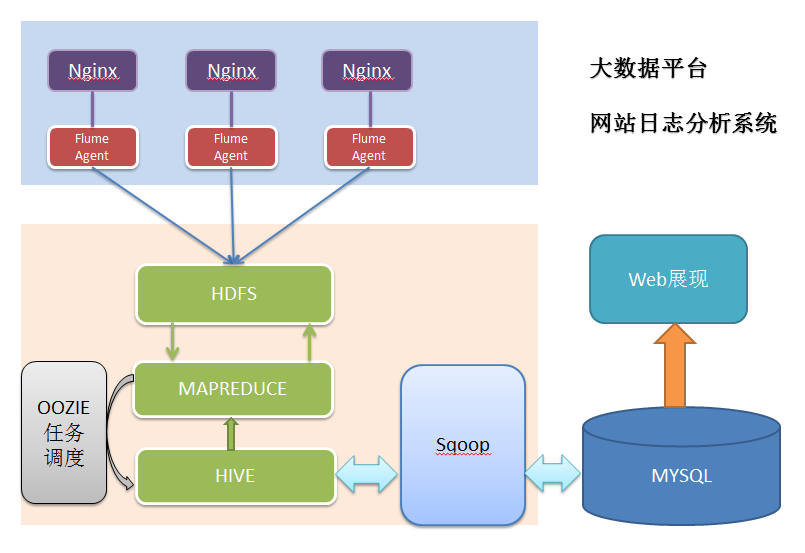
**项目的核心内容，即根据需求开发ETL分析语句，得出各种统计结果；**

### 3.1.5 数据展现

**将分析所得数据进行可视化；**

## 3.2 项目架构：

**由于本项目是一个纯粹数据分析项目，其整体结构亦跟分析流程匹配，并没有特别复杂的结构，如下图：**

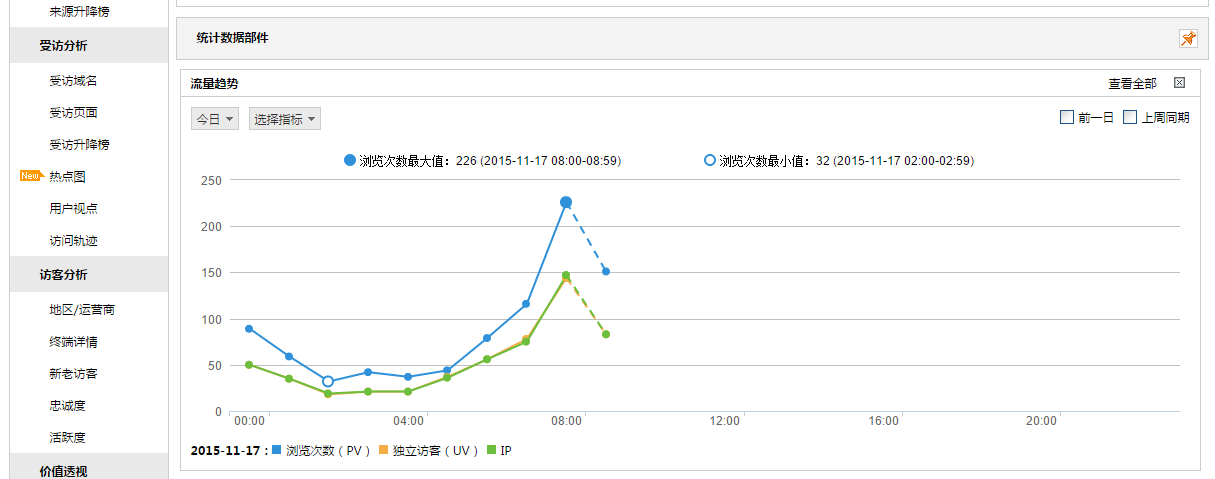
****

**其中，需要强调的是：**

系统的数据分析不是一次性的，而是按照一定的时间频率反复计算，因而整个处理链条中的各个环节需要按照一定的先后依赖关系紧密衔接，即涉及到大量任务单元的管理调度，所以，项目中需要添加一个任务调度模块；

## 3.3 数据展现

**数据展现的目的是将分析所得的数据进行可视化，以便运营决策人员能更方便地获取数据，更快更简单地理解数据：**

****

# 四、模块开发一：数据采集

## 4.1 需求：

**数据采集的需求广义上来说分为两大部分：**

1）是在页面采集用户的访问行为，具体开发工作：

开发页面埋点js，采集用户访问行为，后台接受页面js请求记录日志。此部分工作也可以归属为“数据源”，其开发工作通常由web开发团队负责。

2）是从web服务器上汇聚日志到HDFS，是数据分析系统的数据采集，此部分工作由数据分析平台建设团队负责，具体的技术实现有很多方式：

* **Shell脚本**

**优点：轻量级，开发简单**

**缺点：对日志采集过程中的容错处理不便控制**

* **Java采集程序**

**优点：可对采集过程实现精细控制**

**缺点：开发工作量大**

* **Flume日志采集框架**

**成熟的开源日志采集系统，且本身就是hadoop生态体系中的一员，与hadoop体系中的各种框架组件具有天生的亲和力，可扩展性强；**

## 4.2 技术选型：

**在用户画像日志分析这种场景中，对数据采集部分的可靠性、容错能力要求通常不会非常严苛，因此使用通用的flume日志采集框架完全可以满足需求。**

**本项目即使用flume来实现日志采集。**

## 4.3 Flume日志采集系统搭建：

### 4.3.1 数据源信息：

**本项目分析的数据用nginx服务器所生成的流量日志，存放在各台nginx服务器上.**

****

### 4.3.2 数据内容样例：

**数据的具体内容在采集阶段其实不用太关心。**

|  |
| --- |
| ***58.215.204.118 - - [18/Sep/2013:06:51:35 +0000] "GET /wp-includes/js/jquery/jquery.js?ver=1.10.2 HTTP/1.1" 304 0 "http://blog.fens.me/nodejs-socketio-chat/" "Mozilla/5.0 (Windows NT 5.1; rv:23.0) Gecko/20100101 Firefox/23.0"*** |

**字段解析：**

|  |
| --- |
| ***1、访客ip地址： 58.215.204.118***  ***2、访客用户信息： - -***  ***3、请求时间： [18/Sep/2013:06:51:35 +0000]***  ***4、请求方式： GET***  ***5、请求的url： /wp-includes/js/jquery/jquery.js?ver=1.10.2***  ***6、请求所用协议： HTTP/1.1***  ***7、响应码： 304***  ***8、返回的数据流量： 0***  ***9、访客的来源url： <http://blog.fens.me/nodejs-socketio-chat/>***  ***10、访客所用浏览器： Mozilla/5.0 (Windows NT 5.1; rv:23.0) Gecko/20100101 Firefox/23.0*** |

### 4.3.3 日志文件生成规律：

**基本规律为：当前正在写的文件为access\_log；文件体积达到256M（自定义，一般情况一个文件约为1G），或时间间隔达到60分钟，即滚动重命名切换成历史日志文件；**

**形如： access\_log.2015-11-10-13-00.log**

**当然，每个公司的web服务器日志策略不同，可在web程序的log4j.properties中定义，如下：**

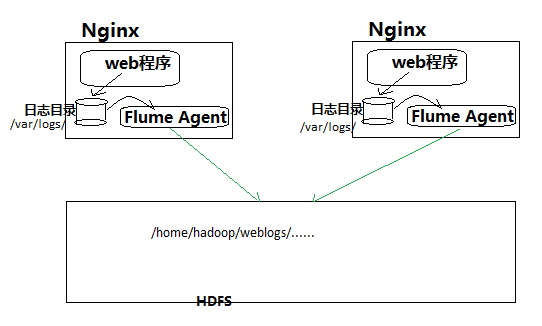
|  |
| --- |
| *log4j.appender.logDailyFile = org.apache.log4j.DailyRollingFileAppender*  *log4j.appender.logDailyFile.layout = org.apache.log4j.PatternLayout*  *log4j.appender.logDailyFile.layout.ConversionPattern = [%-5p][%-22d{yyyy/MM/dd HH:mm:ssS}][%l]%n%m%n*  *log4j.appender.logDailyFile.Threshold = DEBUG*  *log4j.appender.logDailyFile.ImmediateFlush = TRUE*  *log4j.appender.logDailyFile.Append = TRUE*  *log4j.appender.logDailyFile.File = /var/logs/access\_log*  *log4j.appender.logDailyFile.DatePattern = '.'yyyy-MM-dd-HH-mm'.log'*  *log4j.appender.logDailyFile.Encoding = UTF-8* |

### 4.3.4 Flume采集实现：

**Flume采集系统的搭建相对简单：**

1. **在个web服务器上部署agent节点，修改配置文件；**
2. **启动agent节点，将采集到的数据汇聚到指定的HDFS目录中；**

**如下图：**

****

* **版本选择：apache-flume-1.6.0**
* **采集规则设计：**

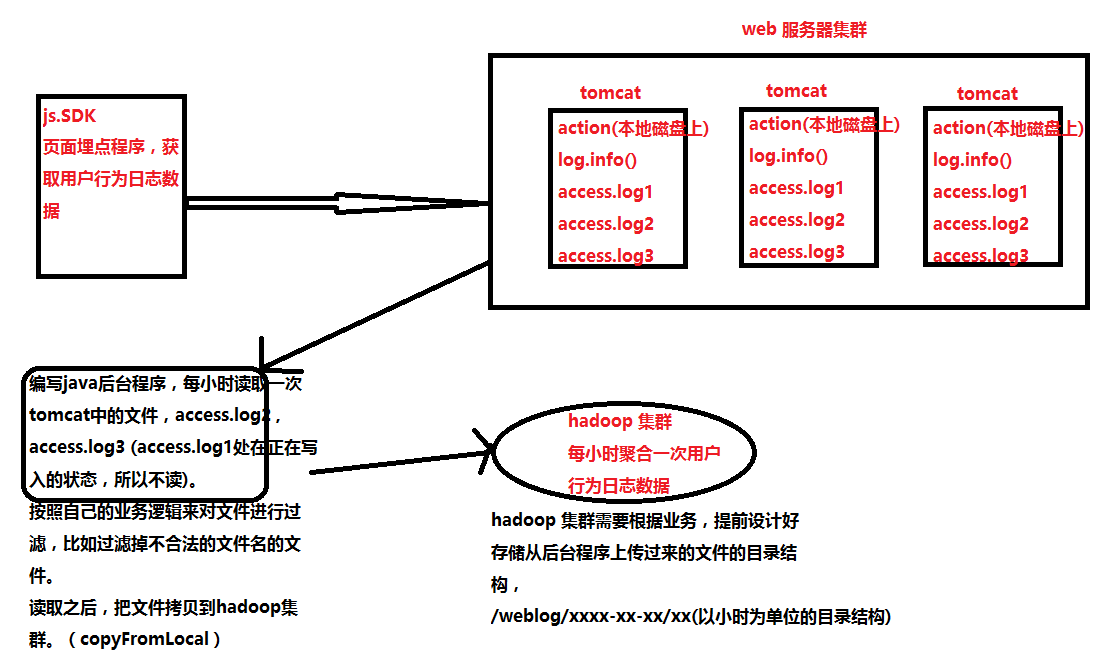
1. **采集源：nginx服务器日志目录**
2. **存放地：/flume/webClick/%y-%m-%d/ （hdfs目录）**

### 4.3.5 脚本：dataCollect.conf

|  |
| --- |
| *a1.sources = r1*  *a1.sinks = k1*  *a1.channels = c1*  *a1.sources.r1.type = exec*  *a1.sources.r1.command = tail -F /root/uek/project/webClick/logs/access.log*  *a1.sinks.k1.type = hdfs*  *#指定hdfs目录格式：年月日 （小时：/%y-%m-%d/%H/ 日小时：/%y-%m-%d/%d-%H/）*  *a1.sinks.k1.hdfs.path = /project/webClick/logs/ %y-%m-%d*  *#生成文件前缀*  *a1.sinks.k1.hdfs.filePrefix = webClick-*    *### 在hdfs上生成文件策略：三种策略，满足一个就会执行 ###*  *#以下策略：每隔60s或者文件大小超过10M的时候产生新文件*  *# hdfs有多少条消息时新建文件，0不基于消息个数*  *a1.sinks.k1.hdfs.rollCount=0*  *# hdfs创建多长时间新建文件，0不基于时间*  *a1.sinks.k1.hdfs.rollInterval=60*  *# hdfs多大时新建文件，0不基于文件大小 单位：byte*  *a1.sinks.k1.hdfs.rollSize=1024000000*    *# 当目前被打开的临时文件在该参数指定的时间（秒）内，没有任何数据写入，则将该临时文件关闭并重命名成目标文件*  *a1.sinks.k1.hdfs.idleTimeout=6*  *a1.sinks.k1.hdfs.fileType=DataStream*  *a1.sinks.k1.hdfs.useLocalTimeStamp=true*    *### 在hdfs上生成目录策略 每五分钟生成一个新目录: ###*  *# 是否启用时间上的”舍弃”，这里的”舍弃”，类似于”四舍五入”，如果启用，则会影响除了%t的其他所有时间表达式*  *a1.sinks.k1.hdfs.round=true*  *# 时间上进行“舍弃”的值；*  *a1.sinks.k1.hdfs.roundValue=24*  *# 时间上进行”舍弃”的单位，包含：second,minute,hour*  *a1.sinks.k1.hdfs.roundUnit=hour*  *a1.channels.c1.type = memory*  *a1.channels.c1.capacity = 10000*  *a1.channels.c1.transactionCapacity = 5000*  *a1.sources.r1.channels = c1*  *a1.sinks.k1.channel = c1* |

### 4.3.6 其他两种数据采集的方式：

#### ★案例：开发shell 采集脚本



##### 需求说明：

用户画像日志每天都10T，在业务应用服务器上，需要准实时上传至数据仓库（Hadoop HDFS）上；

##### 需求分析：

一般上传文件都是在凌晨24点操作，由于很多种类的业务数据都要在晚上进行传输，为了减轻服务器的压力，避开高峰期。

如果需要伪实时的上传，则采用定时上传的方式；

##### 技术分析：

HDFS SHELL: hadoop fs –put xxxx.tar /data

可以使用 Java Api满足上传一个文件，不能满足定时、周期性传入。

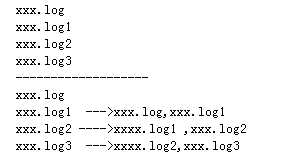
定时调度器：

|  |
| --- |
| *Linux crontab*  *crontab -e*  *\*/5 \* \* \* \* $home/bin/command.sh //五分钟执行一次*  *系统会自动执行脚本，每5分钟一次，执行时判断文件是否符合上传规则，符合则上传* |

##### 实现流程：

日志产生程序：

日志产生程序将日志生成后，产生一个一个的文件，使用滚动模式创建文件名。



日志生成的逻辑由业务系统决定，比如在log4j配置文件中配置生成规则，如：当xxxx.log 等于10G时，滚动生成新日志

|  |
| --- |
| *log4j.logger.msg=info,msg*  *log4j.appender.msg=cn.maoxiangyi.MyRollingFileAppender*  *log4j.appender.msg.layout=org.apache.log4j.PatternLayout*  *log4j.appender.msg.layout.ConversionPattern=%m%n*  *log4j.appender.msg.datePattern='.'yyyy-MM-dd*  *log4j.appender.msg.Threshold=info*  *log4j.appender.msg.append=true*  *log4j.appender.msg.encoding=UTF-8*  *log4j.appender.msg.MaxBackupIndex=100*  *log4j.appender.msg.MaxFileSize=10GB*  *log4j.appender.msg.File=/home/hadoop/logs/log/access.log* |

##### 细节：

如果日志文件后缀是1\2\3等数字，该文件满足需求可以上传的话。把该文件移动到准备上传的工作区间。

工作区间有文件之后，可以使用hadoop put命令将文件上传。

##### 阶段问题：

待上传文件的工作区间的文件，在上传完成之后，是否需要删除掉。

##### 伪代码：

|  |
| --- |
| *使用ls命令读取指定路径下的所有文件信息，*  *ls | while read line*  *//判断line这个文件名称是否符合规则*  *if line=access.log.\* (*  *将文件移动到待上传的工作区间*  *)*  *//批量上传工作区间的文件*  *hadoop fs –put xxx* |

脚本写完之后，配置linux定时任务，每5分钟运行一次。

##### 代码实现：

定时采集用户行为日志 shell脚本：uploadFile2Hdfs.sh

|  |
| --- |
| *#!/bin/bash*  *#编写脚本第一步：配置环境变量：*  *export JAVA\_HOME=/usr/java/default*  *export JRE\_HOME=${JAVA\_HOME}/jre*  *export CLASSPATH=.:${JAVA\_HOME}/lib:${JRE\_HOME}/lib*  *export PATH=${JAVA\_HOME}/bin:$PATH*  *#set hadoop env*  *export HADOOP\_HOME=/home/bigData/hadoop/hadoop-2.7.5*  *export PATH=${HADOOP\_HOME}/bin:${HADOOP\_HOME}/sbin:$PATH*  *#测试：打印环境变量信息是否正常*  *echo "envs: hadoop\_home: $HADOOP\_HOME"*  *#1、flume日志文件存放的目录*  *log\_src\_dir=/root/workerSpace/hadoop/log/*  *#2、待上传文件存放的目录（把文件存放在此目录）*  *log\_toupload\_dir=/root/workerSpace/hadoop/tmp/*  *#提前创建，日志文件上传到hdfs的根路径 ，可以修改成小时 ,在上面添加date变量来获取到小时*  *hdfs\_root\_dir=/testData/20180205/*  *#读取日志文件的目录，判断是否有需要上传的文件*  *echo "log\_src\_dir:"$log\_src\_dir*  *#1、循环一行行读取，然后把读取到的文件存储到变量fileName中；*  *ls $log\_src\_dir | while read fileName*  *# 2、判断 true 就上传到hdfs, false就不上传；*  *do*  *if [[ "$fileName" == access.log.\* ]]; then*  *# if [ "access.log" = "$fileName" ];then*  *#date : 当前年月日时分秒*  *date=`date +%Y\_%m\_%d\_%H\_%M\_%S`*  *#打印信息，检查上面的变量是否正常：*  *echo "moving $log\_src\_dir$fileName to $log\_toupload\_dir"xxxxx\_click\_log\_$fileName"$date"*    *#3、将文件移动到待上传目录并重命名:*  *mv $log\_src\_dir$fileName $log\_toupload\_dir"xxxxx\_click\_log\_$fileName"$date"*  *#4、将待上传的文件path写入一个列表文件willDoing，每一个willDoing文件就是待上传的文件；*  *echo $log\_toupload\_dir"xxxxx\_click\_log\_$fileName"$date >> $log\_toupload\_dir"willDoing."$date*  *fi*  *done*  *#5、循环log\_toupload\_dir目录，找到一个个的willDoing文件*  *ls $log\_toupload\_dir | grep will |grep -v "\_COPY\_" | grep -v "\_DONE\_" | while read line*  *do*  *#打印信息，查看一下每一个willDoing文件的文件名：*  *echo "toupload is in file:"$line*    *#6、将待上传的一个个willDoing改名为willDoing\_COPY\_*  *mv $log\_toupload\_dir$line $log\_toupload\_dir$line"\_COPY\_"*    *#7、查看列表文件willDoing\_COPY\_的内容（一个一个的待上传文件名） ,此处的line 就是列表中的一个待上传文件的path*  *cat $log\_toupload\_dir$line"\_COPY\_" |while read line*    *do*  *#打印信息，检查信息是否正确：*  *echo "puting...$line to hdfs path.....$hdfs\_root\_dir"*    *#上传至hdfs:*  *hadoop fs -put $line $hdfs\_root\_dir*  *done*  *#当上传后，就把上传过的文件名改为\_DONE\_；*  *mv $log\_toupload\_dir$line"\_COPY\_" $log\_toupload\_dir$line"\_DONE\_"*  *done* |

##### 效果展示及操作步骤:

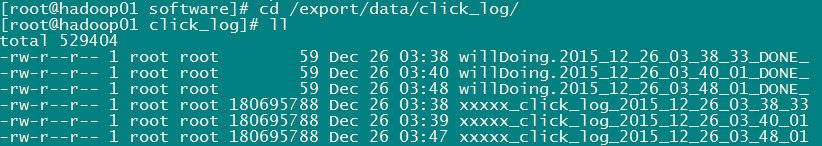
1、日志收集文件收集数据，并将数据保存起来，效果如下：



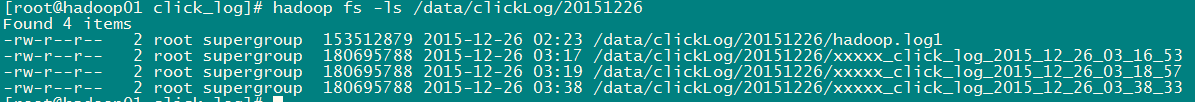
2、上传程序通过crontab定时调度



3、程序运行时产生的临时文件



4、Hadoo hdfs上的效果



#### ★案例：开发JAVA采集程序

##### 需求:

从外部购买数据，数据提供方会实时将数据推送到6台FTP服务器上，我方部署6台接口采集机来对接采集数据，并上传到HDFS中;

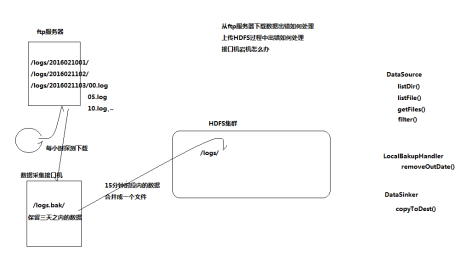
提供商在FTP上生成数据的规则是以小时为单位建立文件夹(2016-03-11-10)，每分钟生成一个文件（00.dat,01.data,02.dat,........）

提供方不提供数据备份，推送到FTP服务器的数据如果丢失，不再重新提供，且FTP服务器磁盘空间有限，最多存储最近10小时内的数据;

由于每一个文件比较小，只有150M左右，因此，我方在上传到HDFS过程中，需要将15分钟时段的数据合并成一个文件上传到HDFS;

为了区分数据丢失的责任，我方在下载数据时最好进行校验

##### 设计分析



# 五、模块开发二：数据预处理

## 5.1目的：

过滤“不合规”数据、格式转换和规整、根据后续的统计需求，过滤分离出各种不同主题(不同栏目path)的基础数据；

## 5.2 预处理代码实现：

### 代码1：贴源bean

|  |
| --- |
| */\*\**  *\* <p>Title: WebLogBean</p>*  *\* <p>Description:*  *\* \* 术语：贴源模型：（松耦合设计）*  *\* 对接外部数据的层（ODS），表结构定义需要和外部数据源保持一致；*  *\* </p>*  *\* @author 优逸客大数据开发团队*  *\* @date 2018年10月1日*  *\*/*  *public class WebLogBean implements Writable{*    *private boolean valid = true; // 判断数据是否合法*    *private String remote\_addr; // 记录客户端的ip地址*  *private String remote\_user; // 记录客户端用户名称,忽略属性"-"*  *private String time\_local; // 记录访问时间与时区*  *private String request; // 记录请求的url*  *private String status; // 记录请求状态；成功是200*  *private String body\_bytes\_sent; // 记录发送给客户端文件主体内容大小*  *private String http\_referer; // 用来记录从那个页面链接访问过来的*  *private String http\_user\_agent; // 记录客户浏览器的相关信息*  *public void set(boolean valid,String remote\_addr, String remote\_user, String time\_local, String request, String status, String body\_bytes\_sent, String http\_referer, String http\_user\_agent) {*  *this.valid = valid;*  *this.remote\_addr = remote\_addr;*  *this.remote\_user = remote\_user;*  *this.time\_local = time\_local;*  *this.request = request;*  *this.status = status;*  *this.body\_bytes\_sent = body\_bytes\_sent;*  *this.http\_referer = http\_referer;*  *this.http\_user\_agent = http\_user\_agent;*  *}*    *@Override*  *public String toString() {*  *StringBuilder sb = new StringBuilder();*  *sb.append(this.valid);*  *sb.append("\001").append(this.getRemote\_addr());*  *sb.append("\001").append(this.getRemote\_user());*  *sb.append("\001").append(this.getTime\_local());*  *sb.append("\001").append(this.getRequest());*  *sb.append("\001").append(this.getStatus());*  *sb.append("\001").append(this.getBody\_bytes\_sent());*  *sb.append("\001").append(this.getHttp\_referer());*  *sb.append("\001").append(this.getHttp\_user\_agent());*  *return sb.toString();*  *}*    *@Override*  *public void readFields(DataInput in) throws IOException {*  *this.valid = in.readBoolean();*  *this.remote\_addr = in.readUTF();*  *this.remote\_user = in.readUTF();*  *this.time\_local = in.readUTF();*  *this.request = in.readUTF();*  *this.status = in.readUTF();*  *this.body\_bytes\_sent = in.readUTF();*  *this.http\_referer = in.readUTF();*  *this.http\_user\_agent = in.readUTF();*  *}*  *@Override*  *public void write(DataOutput out) throws IOException {*  *out.writeBoolean(this.valid);*  *out.writeUTF(null==remote\_addr?"":remote\_addr);*  *out.writeUTF(null==remote\_user?"":remote\_user);*  *out.writeUTF(null==time\_local?"":time\_local);*  *out.writeUTF(null==request?"":request);*  *out.writeUTF(null==status?"":status);*  *out.writeUTF(null==body\_bytes\_sent?"":body\_bytes\_sent);*  *out.writeUTF(null==http\_referer?"":http\_referer);*  *out.writeUTF(null==http\_user\_agent?"":http\_user\_agent);*  *}*  *public boolean isValid() {*  *return valid;*  *}*  *public void setValid(boolean valid) {*  *this.valid = valid;*  *}*  *public String getRemote\_addr() {*  *return remote\_addr;*  *}*  *public void setRemote\_addr(String remote\_addr) {*  *this.remote\_addr = remote\_addr;*  *}*  *public String getRemote\_user() {*  *return remote\_user;*  *}*  *public void setRemote\_user(String remote\_user) {*  *this.remote\_user = remote\_user;*  *}*  *public String getTime\_local() {*  *return time\_local;*  *}*  *public void setTime\_local(String time\_local) {*  *this.time\_local = time\_local;*  *}*  *public String getRequest() {*  *return request;*  *}*  *public void setRequest(String request) {*  *this.request = request;*  *}*  *public String getStatus() {*  *return status;*  *}*  *public void setStatus(String status) {*  *this.status = status;*  *}*  *public String getBody\_bytes\_sent() {*  *return body\_bytes\_sent;*  *}*  *public void setBody\_bytes\_sent(String body\_bytes\_sent) {*  *this.body\_bytes\_sent = body\_bytes\_sent;*  *}*  *public String getHttp\_referer() {*  *return http\_referer;*  *}*  *public void setHttp\_referer(String http\_referer) {*  *this.http\_referer = http\_referer;*  *}*  *public String getHttp\_user\_agent() {*  *return http\_user\_agent;*  *}*  *public void setHttp\_user\_agent(String http\_user\_agent) {*  *this.http\_user\_agent = http\_user\_agent;*  *}*  *}* |

### 代码2：日志解析类

|  |
| --- |
| */\*\**  *\* <p>Title: WebLogParser</p>*  *\* <p>Description:*  *\* \* 对日志进行预处理：1）处理原始日志，过滤出真实PV；（因为静态资源的请求不算,js请求不算）*  *\* 2）转换时间格式；*  *\* 3）对缺失字段进行填充默认值；*  *\* 4）对记录标记valid和invalid；*  *\* </p>*  *\* @author 优逸客大数据开发团队*  *\* @date 2018年10月1日*  *\*/*  *public class WebLogParser {*    *//外部数据源时间格式*  *public static SimpleDateFormat df1 = new SimpleDateFormat("dd/MMM/yyyy:HH:mm:ss", Locale.US);*  *//转化后的时间格式*  *public static SimpleDateFormat df2 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.US);*    */\*\**  *\* 通过对输入的每一行数据进行校验，然后把相应维度抽取出来，返回一个bean*  *\*/*  *public static WebLogBean parser (String line){*    *WebLogBean webLogBean = new WebLogBean();*    *String[] arr = line.split(" ");*    *//如果字段有11以上，则认为是合法数据；*  *if(arr.length > 11){*  *webLogBean.setRemote\_addr(arr[0]);*  *webLogBean.setRemote\_user(arr[1]);*    */\*\**  *\* 时间格式的转换： [18/Sep/2013:06:49:18 +0000]*  *\*/*  *String time\_local = formatDate(arr[3].substring(1)); //去掉中括号*  *// 如果时间字段有缺省，就赋默认值；*  *if(null == time\_local){*  *time\_local="-invalid\_time-";*  *}*  *webLogBean.setTime\_local(time\_local);*    *webLogBean.setRequest(arr[6]);*  *webLogBean.setStatus(arr[8]);*  *webLogBean.setBody\_bytes\_sent(arr[9]);*  *webLogBean.setHttp\_referer(arr[10]);*    */\*\**  *\* 因为浏览器信息不一致，所以从12个字段开始都是浏览器信息，*  *\* 如果useragent元素较多，拼接useragent:*  *\* 60.208.6.156 - - [18/Sep/2013:06:49:48 +0000] "GET /wp-content/uploads/2013/07/rcassandra.png HTTP/1.0" 200 185524 "http://cos.name/category/software/packages/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/29.0.1547.66 Safari/537.36"*  *\*/*  *if(arr.length > 12){*  *StringBuilder sb = new StringBuilder();*  *for(int i=11;i<arr.length;i++){*  *sb.append(arr[i]);*  *}*  *webLogBean.setHttp\_user\_agent(sb.toString());*  *}else{*  *//如果浏览器信息只有一个字段：*  *webLogBean.setHttp\_user\_agent(arr[11]);*  *}*    */\*\**  *\* 状态码大于400，http错误*  *\*/*  *if(Integer.parseInt(webLogBean.getStatus()) >= 400){*  *webLogBean.setValid(false);*  *}*    *//访问时间错误：*  *if("-invalid\_time-".equals(webLogBean.getTime\_local())){*  *webLogBean.setValid(false);*  *}*    *}else{*  */\*\**  *\* 如果字段小于11 缺省，就不合法*  *\*/*  *webLogBean.setValid(false);*  *}*    *return webLogBean;*  *}*      *public static void filtStaticResource(WebLogBean bean , Set<String> pages){*  *if(!pages.contains(bean.getRequest())){*  *bean.setValid(false);*  *}*  *}*    *public static String formatDate(String time\_local){*  *try{*  *return df2.format(df1.parse(time\_local));*  *}catch (Exception e) {*  *return null;*  *}*  *}*  *}* |

### 代码3：数据预处理

|  |
| --- |
| */\*\**  *\* <p>Title: WeblogPreProcess</p>*  *\* <p>Description: ETL(对用户行为日志原始数据，按照业务需求进行预处理)</p>*  *\* @author 优逸客大数据开发团队*  *\* @date 2018年10月1日*  *\*/*  *public class WeblogPreProcess extends Configured implements Tool{*    *private static class MapTask extends Mapper<LongWritable, Text, Text, NullWritable>{*    *//数据字典表：用来存储网站有效url请求：*  *Set<String> pages = new HashSet<String>();*    *Text dataStr = new Text();*  *NullWritable nulls = NullWritable.get();*    */\*\**  *\* 根据业务分析，统计出哪些url请求是需要的，其他的就不接收请求；*  *\* 只接收以下url请求：这些才是真实的PV；*  *\**  *\* 在mr程序运行时，首先一次性加载到一个set中，后续的每一天url与之匹配；*  *\*/*  *@Override*  *protected void setup(Context context) throws IOException, InterruptedException {*  *pages.add("/about");*  *pages.add("/black-ip-list/");*  *pages.add("/cassandra-clustor/");*  *pages.add("/finance-rhive-repurchase/");*  *pages.add("/hadoop-family-roadmap/");*  *pages.add("/hadoop-hive-intro/");*  *pages.add("/hadoop-zookeeper-intro/");*  *pages.add("/hadoop-mahout-roadmap/");*  *}*    *@Override*  *protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {*    *String line = value.toString();*  *//第一步：对输入的每一行数据进行校验：*  *WebLogBean webLogBean = WebLogParser.parser(line);*  *//第二步：过滤掉：js/图片/css等静态资源，只获取有效url:*  *WebLogParser.filtStaticResource(webLogBean, pages);*  *//第三步： 如果数据校验不合法，就直接返回，不输出*  *if(!webLogBean.isValid()){*  *return;*  *}*    *dataStr.set(webLogBean.toString());*    *context.write(dataStr , nulls);*  *}*  *}*      *@Override*  *public int run(String[] arg0) throws Exception {*    *Configuration conf = getConf();*    *Job job = Job.getInstance(conf , "ETL");*  *job.setJarByClass(WeblogPreProcess.class);*  *job.setMapperClass(MapTask.class);*  *job.setOutputKeyClass(Text.class);*  *job.setOutputValueClass(NullWritable.class);*  *FileInputFormat.setInputPaths(job, new Path(arg0[0]));*  *FileOutputFormat.setOutputPath(job, new Path(arg0[1]));*    *//不需要reduce，必须指定，否则默认会有reduce;*  *job.setNumReduceTasks(0);*    *return job.waitForCompletion(true) ? 0 : 1;*  *}*      *public static void main(String[] args) throws Exception {*    *//本地测试：*  *//args = new String[] {"testData/access.log" , "testData/preLog"};*    *//集群测试：*  *//args = new String[] {"/project/webClick/logs/18-10-01" , "/project/webClick/preLog"};*    *Configuration conf = new Configuration();*  *//conf.set("fs.defaultFS", "hdfs://node1:8020");*  *//创建一个主类对象：*  *WeblogPreProcess preProcess = new WeblogPreProcess();*    *int status = ToolRunner.run(conf, preProcess , args);*    *System.exit(status);*  *}*  *}* |

## 5.3 shell脚本：执行预处理程序 log\_preprocess.sh

|  |
| --- |
| ***#!/bin/bash***  ***#set java env***  ***export JAVA\_HOME=/usr/java/default***  ***export JRE\_HOME=${JAVA\_HOME}/jre***  ***export CLASSPATH=.:${JAVA\_HOME}/lib:${JRE\_HOME}/lib***  ***export PATH=${JAVA\_HOME}/bin:$PATH***  ***#set hadoop env***  ***export HADOOP\_HOME=/home/bigData/hadoop-2.6.5/***  ***export PATH=${HADOOP\_HOME}/bin:${HADOOP\_HOME}/sbin:$PATH***  ***#预处理程序类名***  ***preprocess\_class="com.syx.hive.mr.pre.WeblogPreProcess"***  ***#只输出valid记录的预处理程序类名***  ***pre\_valid\_class="com.syx.hive.mr.pre.WeblogPreValid"***  ***#待处理日志存放的目录 : hdfs目录，flume生成的日志目录***  ***log\_pre\_input=/flume/webClick/preprocessDir/input***  ***#预处理输出结果(raw)目录***  ***log\_pre\_output=/flume/webClick/preprocessDir/output***  ***#预处理输出结果(valid)目录***  ***log\_pre\_valid\_output=/flume/webClick/preprocessDir/valid\_output***  ***#获取时间信息（前一天） ： day\_01 该变量下面多处引用***  ***day\_01=`date -d'-1 day' +%Y-%m-%d`***  ***# 年***  ***syear=`date --date=$day\_01 +%Y`***  ***# 月***  ***smonth=`date --date=$day\_01 +%m`***  ***# 日***  ***sday=`date --date=$day\_01 +%d`***  ***#读取日志文件的目录，判断是否有当日待处理的目录(如：2016-03-18)***  ***files=`hadoop fs -ls $log\_pre\_input | grep $day\_01 | wc -l`***  ***#如果files变量有数据：***  ***if [ $files -gt 0 ]; then***  ***#提交mr任务job运行（把代码打包成jar包。weblog.jar）***  ***echo "running.. hadoop jar weblog.jar $preprocess\_class $log\_pre\_input/$day\_01 /$log\_pre\_output/$day\_01"***  ***hadoop jar weblog.jar $preprocess\_class $log\_pre\_input/$day\_01 $log\_pre\_output/$day\_01***  ***fi***  ***echo "raw预处理运行结果： $?"***  ***#1、如果执行成功：***  ***if [ $? -eq 0 ];then***  ***#提交mr任务job运行***  ***echo "running.. hadoop jar weblog.jar $pre\_valid\_class $log\_pre\_input $day\_01 /$log\_pre\_valid\_output/$day\_01"***  ***hadoop jar weblog.jar $pre\_valid\_class $log\_pre\_input/$day\_01 $log\_pre\_valid\_output/$day\_01***  ***fi***  ***#2、如果执行失败：***  ***#发送邮件或短信，人为来干预*** |

## 5.4 用户画像模型构建代码实现：

**由于大量的指标统计从用户画像模型中更容易得出，所以在预处理阶段，可以使用mr程序来生成用户画像模型的数据；**

### 代码1：pageviews模型bean

|  |
| --- |
| */\*\**  *\* <p>Title: PageViewsBean</p>*  *\* <p>Description: 页面用户画像模型Pageviews模型（每一个页面的数据）</p>*  *\* @author 优逸客大数据开发团队*  *\* @date 2018年10月1日*  *\*/*  *public class PageViewsBean implements Writable{*  *private String session; //会话id*  *private String remote\_addr; //记录客户端的ip地址*  *private String remote\_user; // 记录客户端用户名称,忽略属性"-"*  *private String timestr; //时间*  *private String request; //记录请求的url、http协议*  *private int step; //访问页面数*  *private String staylong; //停留*  *private String referal; //来源url*  *private String useragent; //浏览器信息*  *private String bytes\_send; //发送内容的大小*  *private String status; //状态码*    */\**  *\* 总体设置：*  *\*/*  *public void set(String session, String remote\_addr, String remote\_user , String timestr, String request, int step, String staylong, String referal, String useragent, String bytes\_send, String status){*  *this.session = session;*  *this.remote\_addr = remote\_addr;*  *this.remote\_user = remote\_user;*  *this.timestr = timestr;*  *this.request = request;*  *this.step = step;*  *this.staylong = staylong;*  *this.referal = referal;*  *this.useragent = useragent;*  *this.bytes\_send = bytes\_send;*  *this.status = status;*  *}*      *@Override*  *public void readFields(DataInput in) throws IOException {*    *this.session = in.readUTF();*  *this.remote\_addr = in.readUTF();*  *this.remote\_user = in.readUTF();*  *this.timestr = in.readUTF();*  *this.request = in.readUTF();*  *this.step = in.readInt();*  *this.staylong = in.readUTF();*  *this.referal = in.readUTF();*  *this.useragent = in.readUTF();*  *this.bytes\_send = in.readUTF();*  *this.status = in.readUTF();*  *}*  *@Override*  *public void write(DataOutput out) throws IOException {*  *out.writeUTF(session);*  *out.writeUTF(remote\_addr);*  *out.writeUTF(remote\_user);*  *out.writeUTF(timestr);*  *out.writeUTF(request);*  *out.writeInt(step);*  *out.writeUTF(staylong);*  *out.writeUTF(referal);*  *out.writeUTF(useragent);*  *out.writeUTF(bytes\_send);*  *out.writeUTF(status);*  *}*  *public String getSession() {*  *return session;*  *}*  *public void setSession(String session) {*  *this.session = session;*  *}*  *public String getRemote\_addr() {*  *return remote\_addr;*  *}*  *public void setRemote\_addr(String remote\_addr) {*  *this.remote\_addr = remote\_addr;*  *}*  *public String getTimestr() {*  *return timestr;*  *}*  *public void setTimestr(String timestr) {*  *this.timestr = timestr;*  *}*  *public String getRequest() {*  *return request;*  *}*  *public void setRequest(String request) {*  *this.request = request;*  *}*  *public int getStep() {*  *return step;*  *}*  *public void setStep(int step) {*  *this.step = step;*  *}*  *public String getStaylong() {*  *return staylong;*  *}*  *public void setStaylong(String staylong) {*  *this.staylong = staylong;*  *}*  *public String getReferal() {*  *return referal;*  *}*  *public void setReferal(String referal) {*  *this.referal = referal;*  *}*  *public String getUseragent() {*  *return useragent;*  *}*  *public void setUseragent(String useragent) {*  *this.useragent = useragent;*  *}*  *public String getBytes\_send() {*  *return bytes\_send;*  *}*  *public void setBytes\_send(String bytes\_send) {*  *this.bytes\_send = bytes\_send;*  *}*  *public String getStatus() {*  *return status;*  *}*  *public void setStatus(String status) {*  *this.status = status;*  *}*  *public String getRemote\_user() {*  *return remote\_user;*  *}*  *public void setRemote\_user(String remote\_user) {*  *this.remote\_user = remote\_user;*  *}*  *}* |

### 代码2：pageviews模型计算

|  |
| --- |
| */\*\**  *\* <p>Title: ClickStreamPageview</p>*  *\* <p>Description:*  *\* 输入数据是ETL（WeblogPreProcess）之后的数据；*  *\* 将清洗之后的日志梳理出用户画像pageviews模型数据：*  *\* 1）区分出每一次会话，给每一次visit（session）增加了session-id(随机uuid)*  *\* 2）梳理出每一次会话中所访问的每个页面（请求时间，url，停留时长，以及该页面在这次session中的序号）*  *\* 3）保留referral\_url, body\_bytes\_send. useragent*  *\* </p>*  *\* @author 优逸客大数据开发团队*  *\* @date 2018年10月1日*  *\*/*  *public class ClickStreamPageview extends Configured implements Tool{*    *private static class MapTask extends Mapper<LongWritable, Text, Text, WebLogBean>{*    *Text ip = new Text();*  *WebLogBean bean = new WebLogBean();*    *@Override*  *protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {*    *String line = value.toString();*  *String[] fields = line.split("\001");*  *//非法校验：*  *if(fields.length < 9){*  *return;*  *}*    *//将切分出来的各字段set到weblogbean中:*  *boolean valid = "true".equals(fields[0]) ? true : false;*  *String remote\_addr = fields[1];*  *String remote\_user = fields[2];*  *String time\_local = fields[3];*  *String request = fields[4];*  *String status = fields[5];*  *String body\_bytes\_sent = fields[6];*  *String http\_referer = fields[7];*  *String http\_user\_agent = fields[8];*    *//封装对象：*  *bean.set(valid, remote\_addr, remote\_user, time\_local, request, status, body\_bytes\_sent, http\_referer, http\_user\_agent);*    *//有效记录才进入后续处理*  *if(bean.isValid()){*    *ip.set(bean.getRemote\_addr());*  *//key = ip value=bean*  *context.write(ip , bean);*  *}*  *}*  *}*      */\**  *\* 按照提取设计好的跑批时间，reduce聚合统计的是一个时间段内的数据*  *\*/*  *private static class ReduceTask extends Reducer<Text, WebLogBean, NullWritable, Text>{*    *Text totalData = new Text();*    *@Override*  *protected void reduce(Text userIp, Iterable<WebLogBean> iterator,Context context) throws IOException, InterruptedException {*    *List<WebLogBean> beans = new ArrayList<WebLogBean>();*  *// 将一个用户的访问记录全部获取，放到集合中, 然后进行排序：*  *try{*  *for(WebLogBean bean : iterator){*    *WebLogBean webLogBean = new WebLogBean();*  *try {*  *//内存优化：使用复制出来的新的对象来替代旧的对象，从而减少对内存的使用：*  *BeanUtils.copyProperties(webLogBean, bean);*  *} catch (Exception e) {*  *e.printStackTrace();*  *}*  *//将bean添加到list中：目的：bean之间的排序*  *beans.add(webLogBean);*  *}*    *//将所有bean按时间先后顺序升序排序：*  *Collections.sort(beans, new Comparator<WebLogBean>() {*  *@Override*  *public int compare(WebLogBean o1, WebLogBean o2) {*  *try {*  *Date d1 = toDate(o1.getTime\_local());*  *Date d2 = toDate(o2.getTime\_local());*  *if (d1 == null || d2 == null) return 0;*    *return d1.compareTo(d2); //时间默认升序*  *} catch (Exception e) {*  *e.printStackTrace();*  *return 0;*  *}*  *}*  *});*      */\*\**  *\* 以下逻辑为：从有序bean中分辨出各次visit，并对一次visit中所访问的page按顺序标号step*  *\* 1）当只有一条*  *\* 2）当有多条*  *\* 当第一条 continue*  *\* 当不是第一条，时间在30分钟内吗?在, 写出去, step+1, 不在就step =1, 重置sessionid*  *\* 当最后一条，写出去, 设置默认停留时长为60s*  *\*/*  *int step = 1; //默认访问第一个页面：*    *String session = UUID.randomUUID().toString();//随机生成uuid，作为session；*    *for(int i=0;i<beans.size();i++){*    *WebLogBean bean = beans.get(i);*    *// 1、如果一个用户只有一条记录； 如果仅有1条数据，则直接输出:*  *if (1 == beans.size()) {*    *// 设置默认停留时长为60s*  *totalData.set(session+"\001"+userIp.toString()+"\001"+bean.getRemote\_user() + "\001" + bean.getTime\_local() + "\001" + bean.getRequest() + "\001" +*  *step + "\001" + (60) + "\001" + bean.getHttp\_referer() + "\001" + bean.getHttp\_user\_agent() + "\001" + bean.getBody\_bytes\_sent() + "\001" + bean.getStatus());*    *context.write(NullWritable.get(), totalData);*  *break;*  *}*  *//2、如果不止1条数据，则将第一条跳过不输出，遍历第二条时再输出:*  *if (i == 0) {*  *continue;*  *}*  *//当第二条时，开始执行下面逻辑：*  *// 求近本次（第二条）---上次（第一条）之间的时间差*  *long timeDiff = timeDiff(toDate(bean.getTime\_local()), toDate(beans.get(i - 1).getTime\_local()));*    */\*\**  *\* 如果本次-上次时间差<30分钟，说明两次访问时长是在同一个session内，*  *\* 则输出前一次（第一条）的页面访问信息*  *\*/*  *if (timeDiff < 30 \* 60 \* 1000) {*    *totalData.set(session+"\001"+userIp.toString()+"\001"+beans.get(i - 1).getRemote\_user() + "\001" + beans.get(i - 1).getTime\_local() + "\001" + beans.get(i - 1).getRequest() + "\001" + step + "\001" + (timeDiff / 1000) + "\001" + beans.get(i - 1).getHttp\_referer() + "\001"*  *+ beans.get(i - 1).getHttp\_user\_agent() + "\001" + beans.get(i - 1).getBody\_bytes\_sent() + "\001" + beans.get(i - 1).getStatus());*  *//输出上次页面访问信息：*  *context.write(NullWritable.get(), totalData);*    *step++; // 访问页面数加1；*  *} else {*    */\*\**  *\* 如果本次-上次时间差>30分钟，说明两次访问时长已经不在同一个session内，*  *\* 则输出前一次的页面访问信息，同时将step重置，以分隔为新的visit*  *\*/*  *totalData.set(session+"\001"+userIp.toString()+"\001"+beans.get(i - 1).getRemote\_user() + "\001" + beans.get(i - 1).getTime\_local() + "\001" + beans.get(i - 1).getRequest() + "\001" + (step) + "\001" + (60) + "\001" + beans.get(i - 1).getHttp\_referer() + "\001"*  *+ beans.get(i - 1).getHttp\_user\_agent() + "\001" + beans.get(i - 1).getBody\_bytes\_sent() + "\001" + beans.get(i - 1).getStatus());*  *//输出上次页面访问信息：*  *context.write(NullWritable.get(), totalData);*    *// 输出完上一条之后，重置step编号、session*  *step = 1;*  *session = UUID.randomUUID().toString();*  *}*    *//3、如果此次遍历的是最后一条，则将本条直接输出:*  *if (i == beans.size() - 1) {*    *// 设置默认停留时长为60s*  *totalData.set(session+"\001"+userIp.toString()+"\001"+bean.getRemote\_user() + "\001" + bean.getTime\_local() + "\001" + bean.getRequest() + "\001"*  *+ step + "\001" + (60) + "\001" + bean.getHttp\_referer() + "\001" + bean.getHttp\_user\_agent() + "\001" + bean.getBody\_bytes\_sent() + "\001" + bean.getStatus());*    *context.write(NullWritable.get(), totalData);*  *}*  *}*  *}catch (Exception e) {*  *e.printStackTrace();*  *}*  *}*    *private Date toDate(String timeStr) throws Exception {*  *SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.US);*  *return df.parse(timeStr);*  *}*  *private long timeDiff(Date time1, Date time2) throws Exception {*  *return time1.getTime() - time2.getTime();*  *}*  *}*      *@Override*  *public int run(String[] arg0) throws Exception {*    *Configuration conf = getConf();*  *Job job = Job.getInstance(conf , "pageview模型");*  *job.setJarByClass(ClickStreamPageview.class);*    *job.setMapperClass(MapTask.class);*  *job.setMapOutputKeyClass(Text.class);*  *job.setMapOutputValueClass(WebLogBean.class);*  *job.setReducerClass(ReduceTask.class);*  *job.setOutputKeyClass(Text.class);*  *job.setOutputValueClass(Text.class);*    *FileInputFormat.addInputPath(job, new Path(arg0[0]));*  *FileOutputFormat.setOutputPath(job, new Path(arg0[1]));*    *return job.waitForCompletion(true) ? 0 : 1;*  *}*      *public static void main(String[] args) throws Exception {*    *//本地测试：*  *//args = new String[] {"testData/access.log" , "testData/preLog"};*    *//集群测试：*  *//args = new String[] {"/project/webClick/preLog" , "/project/webClick/pageviewsModel"};*    *Configuration conf = new Configuration();*  *//conf.set("fs.defaultFS", "hdfs://node1:8020");*    *//创建一个主类对象：*  *ClickStreamPageview pageview = new ClickStreamPageview();*    *int status = ToolRunner.run(conf, pageview , args);*    *System.exit(status);*  *}*  *}* |

### 代码3：Visits模型bean

|  |
| --- |
| */\*\**  *\* <p>Title: VisitBean</p>*  *\* <p>Description: 用户画像模型Visits模型：按照session聚合统计的页面数据：</p>*  *\* @author 优逸客大数据开发团队*  *\* @date 2018年10月1日*  *\*/*  *public class VisitBean implements Writable{*    *private String session; //用户session*  *private String remote\_addr; //userIp*  *private String inTime; //起始时间*  *private String outTime; //结束时间*  *private String inPage; //起始页面*  *private String outPage; //结束页面*  *private String referal; //来访的url*  *private int pageVisits; //访问的总页数*    *public void set(String session, String remote\_addr, String inTime, String outTime, String inPage, String outPage, String referal, int pageVisits) {*  *this.session = session;*  *this.remote\_addr = remote\_addr;*  *this.inTime = inTime;*  *this.outTime = outTime;*  *this.inPage = inPage;*  *this.outPage = outPage;*  *this.referal = referal;*  *this.pageVisits = pageVisits;*  *}*  *@Override*  *public void readFields(DataInput in) throws IOException {*  *this.session = in.readUTF();*  *this.remote\_addr = in.readUTF();*  *this.inTime = in.readUTF();*  *this.outTime = in.readUTF();*  *this.inPage = in.readUTF();*  *this.outPage = in.readUTF();*  *this.referal = in.readUTF();*  *this.pageVisits = in.readInt();*  *}*  *@Override*  *public void write(DataOutput out) throws IOException {*  *out.writeUTF(session);*  *out.writeUTF(remote\_addr);*  *out.writeUTF(inTime);*  *out.writeUTF(outTime);*  *out.writeUTF(inPage);*  *out.writeUTF(outPage);*  *out.writeUTF(referal);*  *out.writeInt(pageVisits);*  *}*    *//格式化数据输出：*  *@Override*  *public String toString() {*  *return session + "\001" + remote\_addr + "\001" + inTime + "\001" + outTime + "\001" + inPage + "\001" + outPage + "\001" + referal + "\001" + pageVisits;*  *}*  *public String getSession() {*  *return session;*  *}*  *public void setSession(String session) {*  *this.session = session;*  *}*  *public String getRemote\_addr() {*  *return remote\_addr;*  *}*  *public void setRemote\_addr(String remote\_addr) {*  *this.remote\_addr = remote\_addr;*  *}*  *public String getInTime() {*  *return inTime;*  *}*  *public void setInTime(String inTime) {*  *this.inTime = inTime;*  *}*  *public String getOutTime() {*  *return outTime;*  *}*  *public void setOutTime(String outTime) {*  *this.outTime = outTime;*  *}*  *public String getInPage() {*  *return inPage;*  *}*  *public void setInPage(String inPage) {*  *this.inPage = inPage;*  *}*  *public String getOutPage() {*  *return outPage;*  *}*  *public void setOutPage(String outPage) {*  *this.outPage = outPage;*  *}*  *public String getReferal() {*  *return referal;*  *}*  *public void setReferal(String referal) {*  *this.referal = referal;*  *}*  *public int getPageVisits() {*  *return pageVisits;*  *}*  *public void setPageVisits(int pageVisits) {*  *this.pageVisits = pageVisits;*  *}*  *}* |

### 代码4：Visits模型计算

|  |
| --- |
| */\*\**  *\* <p>Title: ClickStreamVisit</p>*  *\* <p>Description: 从pageviews模型（ClickStream）中进一步提炼出的visit模型；</p>*  *\* @author 优逸客大数据开发团队*  *\* @date 2018年10月1日*  *\*/*  *public class ClickStreamVisit extends Configured implements Tool{*  *private static class MapTask extends Mapper<LongWritable, Text, Text, PageViewsBean>{*    *PageViewsBean pageView = new PageViewsBean();*  *Text sessionId = new Text();*    *@Override*  *protected void map(LongWritable key, Text value,Context context) throws IOException, InterruptedException {*    *String line = value.toString();*  *String[] tokens = line.split("\001");*  *String session = tokens[0];*  *String remote\_addr = tokens[1];*  *String remote\_user = tokens[2];*  *String time\_local = tokens[3];*  *String request = tokens[4];*  *Integer step = Integer.valueOf(tokens[5]); //获取访问页面数*  *String staylong = tokens[6];*  *String referal = tokens[7];*  *String useragent = tokens[8];*  *String bytes\_send = tokens[9];*  *String status = tokens[10];*    *pageView.set(session, remote\_addr, remote\_user, time\_local, request, step, staylong, referal, useragent, bytes\_send, status);*    *sessionId.set(pageView.getSession());*    */\**  *\* 一次会话，访问的所有页面的数据:*  *\* sessionId 作为key; value = 每个页面数据*  *\*/*  *context.write(sessionId , pageView);*  *}*  *}*      *private static class ReduceTask extends Reducer<Text, PageViewsBean, NullWritable, VisitBean>{*    *@Override*  *protected void reduce(Text sessionid, Iterable<PageViewsBean> iterator,Context context) throws IOException, InterruptedException {*    *List<PageViewsBean> beans = new ArrayList<PageViewsBean>();*    *for(PageViewsBean pvBean : iterator){*    *PageViewsBean pageViewsBean = new PageViewsBean();*    *try {*  *BeanUtils.copyProperties(pageViewsBean, pvBean);*  *//将bean添加到list中：目的：bean之间的排序*  *beans.add(pageViewsBean);*  *} catch (Exception e) {*  *e.printStackTrace();*  *}*  *}*  */\**  *\* 按照访问页面的顺序降序排序：*  *\* 比如：一个ip :第一个访问的页面 step = 1 www.baidu.com*  *\* 第二个访问的页面 step = 2 www.jd.com*  *\* .........*  *\* 统计目录：可以按照访问的先后顺序来进行页面热度的业务指标分析；*  *\*/*  *Collections.sort(beans , new Comparator<PageViewsBean>() {*  *@Override*  *public int compare(PageViewsBean o1, PageViewsBean o2) {*  *return o1.getStep() > o2.getStep() ? 1 : -1;*  *}*  *});*    *// 取这次visit的首尾pageview记录，将数据放入VisitBean中:*  *VisitBean visitBean = new VisitBean();*    *//当前session中访问的第一个页面信息：请求方式、请求url、http协议；*  *visitBean.setInPage(beans.get(0).getRequest());*  *//当前session中访问的第一个页面的时间；*  *visitBean.setInTime(beans.get(0).getTimestr());*    *//当前session中访问的最后一个页面信息*  *visitBean.setOutPage(beans.get(beans.size() - 1).getRequest());*  *//当前session中访问的最后一个页面的时间；*  *visitBean.setOutTime(beans.get(beans.size() - 1).getTimestr());*    *// visit访问的页面数*  *visitBean.setPageVisits(beans.size());*    *// 来访者的ip*  *visitBean.setRemote\_addr(beans.get(0).getRemote\_addr());*    *// 本次visit的referal*  *visitBean.setReferal(beans.get(0).getReferal());*    *visitBean.setSession(sessionid.toString());*    */\**  *\* key = null , value = visit模型数据*  *\*/*  *context.write(NullWritable.get(), visitBean);*  *}*  *}*      *public static void main(String[] args) throws Exception {*    *//本地测试：*  *//args = new String[] {"testData/access.log" , "testData/preLog"};*    *//集群测试：*  *//args = new String[] {"/project/webClick/pageviewsModel" , "/project/webClick/visitModel"};*    *Configuration conf = new Configuration();*  *//conf.set("fs.defaultFS", "hdfs://node1:8020");*    *//创建一个主类对象：*  *ClickStreamVisit visit = new ClickStreamVisit();*    *int status = ToolRunner.run(conf, visit , args);*    *System.exit(status);*  *}*  *@Override*  *public int run(String[] arg0) throws Exception {*  *Configuration conf = getConf();*    *Job job = Job.getInstance(conf , "visit模型");*  *job.setJarByClass(ClickStreamVisit.class);*  *job.setMapperClass(MapTask.class);*  *job.setReducerClass(ReduceTask.class);*  *job.setMapOutputKeyClass(Text.class);*  *job.setMapOutputValueClass(PageViewsBean.class);*  *job.setOutputKeyClass(NullWritable.class);*  *job.setOutputValueClass(VisitBean.class);*  *FileInputFormat.setInputPaths(job, new Path(arg0[0]));*  *FileOutputFormat.setOutputPath(job, new Path(arg0[1]));*    *return job.waitForCompletion(true) ? 0 : 1;*  *}*  *}* |

# 模块开发三：数据仓库设计及开发

## 6.1 数据仓库设计：

通过对业务需求分析，在需求确立阶段就要对数据仓库进行建模；

注：采用星型模型（一张事实表与多张维表进行关联（维表是事实表中的字段），然后各种维度关联查询；）

雪花模型：对星型模型进行扩展，外层再增加维表进行多级关联。

### 6.1.1 事实表：

|  |  |  |  |
| --- | --- | --- | --- |
| ***原始数据表:t\_origin\_weblog (贴源表)*** | | | |
| ***valid*** | ***string*** | | ***是否有效*** |
| ***remote\_addr*** | ***string*** | | ***访客ip*** |
| ***remote\_user*** | ***string*** | | ***访客用户信息*** |
| ***time\_local*** | ***string*** | | ***请求时间*** |
| ***request*** | ***string*** | | ***请求url*** |
| ***status*** | ***string*** | | ***响应码*** |
| ***body\_bytes\_sent*** | ***string*** | | ***响应字节数*** |
| ***http\_referer*** | ***string*** | | ***来源url*** |
| ***http\_user\_agent*** | ***string*** | | ***访客终端信息*** |
|  |  | |  |
| ***ETL中间表：t\_etl\_referurl 在贴源表的基础上对某些字段进行再次抽取*** | | | |
| ***valid*** | ***string*** | ***是否有效*** | |
| ***remote\_addr*** | ***string*** | ***访客ip*** | |
| ***remote\_user*** | ***string*** | ***访客用户信息*** | |
| ***time\_local*** | ***string*** | ***请求时间*** | |
| ***request*** | ***string*** | ***请求url*** | |
| ***request\_host*** | ***string*** | ***请求的域名*** | |
| ***status*** | ***string*** | ***响应码*** | |
| ***body\_bytes\_sent*** | ***string*** | ***响应字节数*** | |
| ***http\_referer*** | ***string*** | ***来源url*** | |
| ***http\_user\_agent*** | ***string*** | ***访客终端信息*** | |
| ***valid*** | ***string*** | ***是否有效*** | |
| ***remote\_addr*** | ***string*** | ***访客ip*** | |
| ***remote\_user*** | ***string*** | ***访客用户信息*** | |
| ***time\_local*** | ***string*** | ***请求时间*** | |
| ***request*** | ***string*** | ***请求url*** | |
| ***status*** | ***string*** | ***响应码*** | |
| ***body\_bytes\_sent*** | ***string*** | ***响应字节数*** | |
| ***http\_referer*** | ***string*** | ***外链url*** | |
| ***http\_user\_agent*** | ***string*** | ***访客终端信息*** | |
| ***host*** | ***string*** | ***外链url的域名*** | |
| ***path*** | ***string*** | ***外链url的路径*** | |
| ***query*** | ***string*** | ***外链url的参数*** | |
| ***query\_id*** | ***string*** | ***外链url的参数值*** | |
|  |  |  | |
| ***对贴源表进行更细粒度的字段后，形成一张能够代表当前完整业务的大表；（在数据仓库中，建表时，一般不考虑数据的冗余（就是数据的重复），和表与表之间的约束）***  ***形成一张明细宽表，接下来我们所有的查询操作，都以这张大表来进行操作；*** |  |  | |
| ***访问日志明细宽表：t\_ods\_access\_detail 对所有字段进行二次三次抽取*** | | | |
| ***valid*** | ***string*** | ***是否有效*** | |
| ***remote\_addr*** | ***string*** | ***访客ip*** | |
| ***remote\_user*** | ***string*** | ***访客用户信息*** | |
| ***time\_local*** | ***string*** | ***请求时间*** | |
| ***request*** | ***string*** | ***请求url整串*** | |
| ***request\_level1*** | ***string*** | ***请求的一级栏目*** | |
| ***request\_level2*** | ***string*** | ***请求的二级栏目*** | |
| ***request\_level3*** | ***string*** | ***请求的三级栏目*** | |
| ***status*** | ***string*** | ***响应码*** | |
| ***body\_bytes\_sent*** | ***string*** | ***响应字节数*** | |
| ***http\_referer*** | ***string*** | ***来源url*** | |
| ***http\_user\_agent*** | ***string*** | ***访客终端信息*** | |
| ***valid*** | ***string*** | ***是否有效*** | |
| ***remote\_addr*** | ***string*** | ***访客ip*** | |
| ***remote\_user*** | ***string*** | ***访客用户信息*** | |
| ***time\_local*** | ***string*** | ***请求时间*** | |
| ***request*** | ***string*** | ***请求url*** | |
| ***status*** | ***string*** | ***响应码*** | |
| ***body\_bytes\_sent*** | ***string*** | ***响应字节数*** | |
| ***http\_referer*** | ***string*** | ***外链url*** | |
| ***http\_user\_agent*** | ***string*** | ***访客终端信息整串*** | |
| ***http\_user\_agent\_browser*** | ***string*** | ***访客终端浏览器*** | |
| ***http\_user\_agent\_sys*** | ***string*** | ***访客终端操作系统*** | |
| ***http\_user\_agent\_dev*** | ***string*** | ***访客终端设备*** | |
| ***host*** | ***string*** | ***外链url的域名*** | |
| ***path*** | ***string*** | ***外链url的路径*** | |
| ***query*** | ***string*** | ***外链url的参数*** | |
| ***query\_id*** | ***string*** | ***外链url的参数值*** | |
| ***daystr*** | ***string*** | ***日期整串*** | |
| ***tmstr*** | ***string*** | ***时间整串*** | |
| ***month*** | ***string*** | ***月份*** | |
| ***day*** | ***string*** | ***日*** | |
| ***hour*** | ***string*** | ***时*** | |
| ***minute*** | ***string*** | ***分*** | |
| ***##*** | ***##*** | ***##*** | |
| ***mm*** | ***string*** | ***分区字段--月*** | |
| ***dd*** | ***string*** | ***分区字段--日*** | |

### 6.1.2 维度表

|  |
| --- |
| *时间维度 v\_year\_month\_date* |
| *year* |
| *month* |
| *day* |
| *hour* |
| *minute* |

|  |
| --- |
| ***访客地域维度t\_dim\_area*** |
| ***北京*** |
| ***上海*** |
| ***广州*** |
| ***深圳*** |
| ***河北*** |
| ***河南*** |

|  |
| --- |
| ***终端类型维度t\_dim\_termination*** |
| ***uc*** |
| ***firefox*** |
| ***chrome*** |
| ***safari*** |
| ***ios*** |
| ***android*** |

|  |
| --- |
| ***网站栏目维度 t\_dim\_section*** |
| ***跳蚤市场*** |
| ***房租信息*** |
| ***休闲娱乐*** |
| ***建材装修*** |
| ***本地服务*** |
| ***人才市场*** |

## 6.2 数据仓库hive开发：

**该项目的数据分析过程在hadoop集群上实现，主要应用hive数据仓库工具，因此，采集并经过预处理后的数据，需要加载到hive数据仓库中，以进行后续的挖掘分析。**

### 6.2.1 创建一个项目的数据库；

|  |
| --- |
| *Create database webClick;* |

### 6.2.2 创建原始数据表（贴源表）：

在hive仓库中构建事实表；

#### 1、贴源数据表：ods\_weblog\_origin.sql

|  |
| --- |
| ***drop table if exists webClick.ods\_weblog\_origin;***  ***create table webClick.ods\_weblog\_origin(***  ***valid string,***  ***remote\_addr string,***  ***remote\_user string,***  ***time\_local string,***  ***request string,***  ***status string,***  ***body\_bytes\_sent string,***  ***http\_referer string,***  ***http\_user\_agent string***  ***)***  ***partitioned by (datestr string)***  ***row format delimited***  ***fields terminated by '\001';*** |



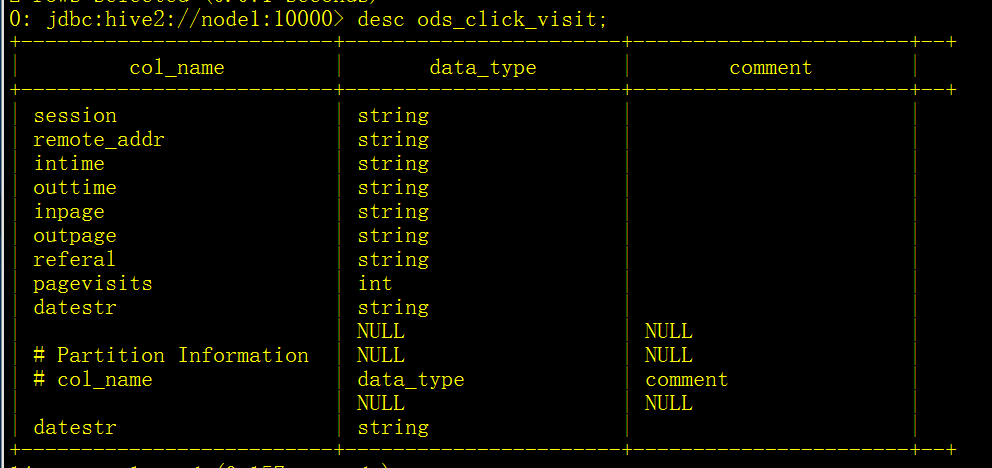
#### 2、用户画像模型pageviews表 ：ods\_click\_pageviews.sql

|  |
| --- |
| ***drop table if exists webClick.ods\_click\_pageviews;***  ***create table webClick.ods\_click\_pageviews(***  ***Session string,***  ***remote\_addr string,***  ***remote\_user string,***  ***time\_local string,***  ***request string,***  ***visit\_step string,***  ***page\_staylong string,***  ***http\_referer string,***  ***http\_user\_agent string,***  ***body\_bytes\_sent string,***  ***status string***  ***)***  ***partitioned by (datestr string)***  ***row format delimited***  ***fields terminated by '\001';*** |



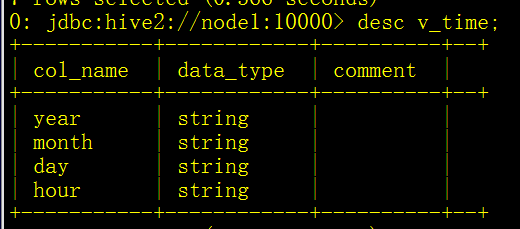
#### 3、用户画像visit模型表：ods\_click\_stream\_visit.sql

|  |
| --- |
| ***drop table if exists webClick.ods\_click\_stream\_visit;***  ***create table webClick.ods\_click\_stream\_visit(***  ***session string,***  ***remote\_addr string,***  ***inTime string,***  ***outTime string,***  ***inPage string,***  ***outPage string,***  ***referal string,***  ***pageVisits int***  ***)***  ***partitioned by (datestr string)***  ***row format delimited***  ***fields terminated by '\001';*** |



#### 4、创建时间维表：dim\_time.sql

|  |
| --- |
| ***create table webClick.dim\_time(***  ***year string,***  ***month string,***  ***day string,***  ***hour string***  ***)***  ***row format delimited***  ***fields terminated by ',';*** |



|  |
| --- |
| *load data local inpath '/root/uek/project/webClick/hiveHqlShell/dim\_time.txt' into table dim\_time;*  *dim\_time.txt:*  *2013,09,18,01*  *2013,09,18,02*  *2013,09,18,03*  *2013,09,18,04*  *2013,09,18,05*  *2013,09,18,06*  *2013,09,18,07*  *2013,09,18,08*  *2013,09,18,09*  *2013,09,18,10*  *2013,09,18,11*  *2013,09,18,12*  *2013,09,18,13*  *2013,09,18,14*  *2013,09,18,15*  *2013,09,18,16*  *2013,09,18,17*  *2013,09,18,18*  *2013,09,18,19*  *2013,09,18,20*  *2013,09,18,21*  *2013,09,18,22*  *2013,09,18,23*  *2013,09,18,24* |

### 6.2.3 导入数据：loadData.sql

|  |
| --- |
| * ***导入清洗结果数据到贴源数据表ods\_weblog\_origin***   ***load data inpath '/project/webClick/preLog/part-m-00000' into table ods\_weblog\_origin partition(datestr='2013-09-18');***  ***（最好多创建一个分区数据，和时间维表关联查询）***  ***0: jdbc:hive2://node1:10000> select count(\*) from ods\_weblog\_origin;***  ***+--------+--+***  ***| \_c0 |***  ***+--------+--+***  ***| 76 |***  ***+--------+--+***   * ***导入用户画像模型pageviews数据到ods\_click\_pageviews表***   ***load data inpath '/project/webClick/pageviewsModel/part-r-00000' into table ods\_click\_pageviews partition(datestr='2013-09-18');***  ***0: jdbc:hive2://node1:10000> select count(\*) from ods\_click\_pageviews;***  ***+------+--+***  ***| \_c0 |***  ***+------+--+***  ***| 76 |***  ***+------+--+***   * ***导入用户画像模型visit数据到click\_stream\_visit表***   ***load data inpath '/project/webClick/visitModel/part-r-00000' into table ods\_click\_stream\_visit partition(datestr='2013-09-18');***  ***0: jdbc:hive2://node1:10000> select count(\*) from ods\_click\_stream\_visit;***  ***+------+--+***  ***| \_c0 |***  ***+------+--+***  ***| 57 |***  ***+------+--+*** |

### 6.2.4 生成ODS层明细宽表：

#### 6.2.4.1 需求概述：

整个数据分析的过程是按照数据仓库的层次分层进行的，总体来说，是从ODS原始数据中整理出一些中间表（比如，为后续分析方便，将原始数据中的时间、url等非结构化数据作结构化抽取，将各种字段信息进行细化，形成明细表），然后再在中间表的基础之上统计出各种指标数据；

#### 6.2.4.2 ETL实现：

经过业务分析，来设计一个多维度的明细表；

建表——明细表ods\_weblog\_detail (源：ods\_weblog\_origin) （目标：ods\_weblog\_detail）

“对贴源表，做更细致的字段抽取。

1. **对来访的URL，做拆分，获取host、path、query、query\_id等维度；**

**（通过函数：*parse\_url\_tuple(regexp\_replace(http\_referer, "\"", ""), 'HOST', 'PATH','QUERY', 'QUERY:id')***

***regexp\_replace(http\_referer, "\"", "") ： 把字符串中的引号去掉。***

***parse\_url\_tuple ： 从url字符串中抽取出各个维度；属于UDAF函数（一进多出）***

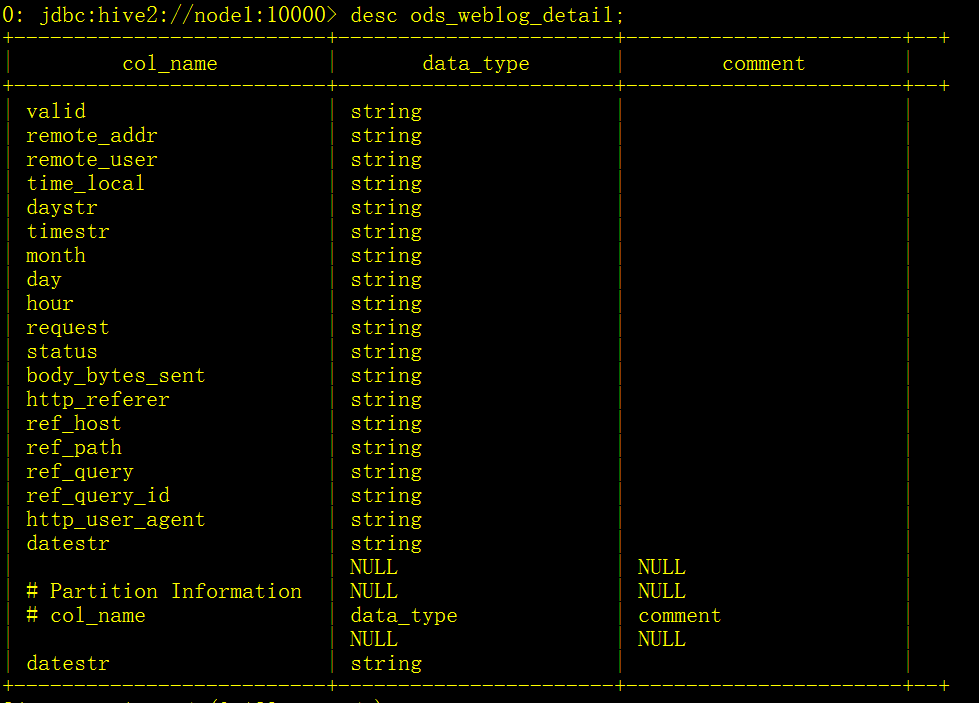
***LATERAL VIEW ： 将解析出来的多个字段转换为多个列（行转列操作）*）**

|  |
| --- |
| ***实验：***  ***测试数据：***  ***url1 http://facebook.com/path1/p.php?k1=v1&k2=v2#Ref1***  ***url2 https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF#LanguageManualUDF-getjsonobjec***  ***turl3 https://www.google.com.hk/#hl=zh-CN&newwindow=1&safe=strict&q=hive+translate+example&oq=hive+trans***  ***late+example&gs\_l=serp.3...10174.11861.6.12051.8.8.0.0.0.0.132.883.0j7.7.0...0.0...1c.1j4.8.serp.0B9C1T\_n0Hs&bav=on.2,or.&bvm=bv.44770516,d.aGc&fp=e13e41a6b9dab3f6&biw=1241&bih=589***  ***Hql:***  ***create table if not exists t\_url(***  ***f1 string,***  ***f2 string)***  ***row format delimited***  ***fields TERMINATED BY ' ' ;***  ***load data local inpath '/root/uek/bigdataApplication/hive-2.3.3/testDir/url.txt' into table t\_url;***  ***SELECT f1, b.\****  ***FROM t\_url***  ***LATERAL VIEW***  ***parse\_url\_tuple(f2, 'HOST', 'PATH', 'QUERY', 'QUERY:k1') b as host, path, query, query\_id;***  ***查询结果：***  ***f1 b.host b.path b.query b.query\_id***  ***url1 facebook.com /path1/p.php k1=v1&k2=v2 v1***  ***url2 cwiki.apache.org /confluence/display/Hive/LanguageManual+UDF NULL NULL***  ***url3 www.google.com.hk / NULL NULL*** |

1. **对来访时间，做拆分，获取年月日时的维度。”**

##### 1、最终构建的明细宽表：ods\_weblog\_detail.sql

|  |
| --- |
| ***drop table webClick.ods\_weblog\_detail;***  ***create table webClick.ods\_weblog\_detail(***  ***valid string, --有效标识***  ***remote\_addr string, --来源IP***  ***remote\_user string, --用户标识***  ***time\_local string, --访问完整时间***  ***daystr string, --访问日期***  ***timestr string, --访问时间***  ***month string, --访问月***  ***day string, --访问日***  ***hour string, --访问时***  ***request string, --请求的url***  ***status string, --响应码***  ***body\_bytes\_sent string, --传输字节数***  ***http\_referer string, --来源url （对多个utl参数进行切分成以下几个字段）***  ***ref\_host string, --来源的host***  ***ref\_path string, --来源的路径***  ***ref\_query string, --来源参数query***  ***ref\_query\_id string, --来源参数query的值***  ***http\_user\_agent string --客户终端标识***  ***)***  ***partitioned by(datestr string)***  ***row format delimited***  ***fields terminated by '\001';*** |



##### 2、中间表：

可以分开实现，创建几张中间表；

1. 抽取refer\_url到中间表 "t\_ods\_tmp\_referurl"
2. 将来访url分离出host path query query id等列,然后添加到贴源表的列中；

***webClick.ods\_tmp\_referurl.sql***

|  |
| --- |
| ***select regexp\_replace(http\_referer,"\"","") from ods\_weblog\_origin;***  ***select b.\****  ***from webClick.ods\_weblog\_origin a***  ***lateral view***  ***parse\_url\_tuple(regexp\_replace(http\_referer,"\"","") , 'HOST', 'PATH', 'QUERY', 'QUERY:url') b***  ***as host , path ,query,query\_url;***  ***drop table if exists webClick.ods\_tmp\_referurl;***  ***create table webClick.ods\_tmp\_referurl***  ***as***  ***select a.\* , b.\****  ***from webClick.ods\_weblog\_origin a***  ***lateral view***  ***parse\_url\_tuple(regexp\_replace(http\_referer,"\"","") , 'HOST', 'PATH', 'QUERY', 'QUERY:url') b***  ***as host , path ,query,query\_url;*** |



***ods\_tmp\_detail.sql：***

*--抽取转换time\_local字段到中间表明细表 ”t\_ ods \_detail”*

*(在将来访时间拆分，把抽取出来的多个维度，添加到上一步新生成的url表的列中。然后把该表的数据插入到webClick.ods\_weblog\_detail表中)*

|  |
| --- |
| ***2013-09-18 07:10:50***  ***select***  ***substring(time\_local,1,10) as daystr,***  ***substring(time\_local,12) as tmstr,***  ***substring(time\_local,6,2) as month,***  ***substring(time\_local,9,2) as day,***  ***substring(time\_local,12,2) as hour***  ***from***  ***webClick.ods\_tmp\_referurl b;***  ***drop table if exists webClick.ods\_tmp\_detail;***  ***create table webClick.ods\_tmp\_detail***  ***as***  ***Select b.\* ,***  ***substring(time\_local,1,10) as daystr,***  ***substring(time\_local,12) as tmstr,***  ***substring(time\_local,6,2) as month,***  ***substring(time\_local,9,2) as day,***  ***substring(time\_local,12,2) as hour***  ***from***  ***webClick.ods\_tmp\_referurl b;*** |



*Load\_ods\_weblog\_detail.sql：*

**3张中间表的建表整合，语句可以改写成：**

|  |
| --- |
| ***INSERT INTO TABLE webClick.ods\_weblog\_detail PARTITION (datestr = '2013-09-18')***  ***SELECT***  ***c.valid,***  ***c.remote\_addr,***  ***c.remote\_user,***  ***c.time\_local,***  ***substring(time\_local,1,10) as daystr,***  ***substring(time\_local,12) as tmstr,***  ***substring(time\_local,6,2) as month,***  ***substring(time\_local,9,2) as day,***  ***substring(time\_local,12,2) as hour,***  ***c.request,***  ***c. STATUS,***  ***c.body\_bytes\_sent,***  ***c.http\_referer,***  ***c.ref\_host,***  ***c.ref\_path,***  ***c.ref\_query,***  ***c.ref\_query\_url,***  ***c.http\_user\_agent***  ***FROM***  ***(***  ***SELECT***  ***a.valid,***  ***a.remote\_addr,***  ***a.remote\_user,***  ***a.time\_local,***  ***a.request,***  ***a. STATUS,***  ***a.body\_bytes\_sent,***  ***a.http\_referer,***  ***a.http\_user\_agent,***  ***b.ref\_host,***  ***b.ref\_path,***  ***b.ref\_query,***  ***b.ref\_query\_url***  ***FROM***  ***webClick.ods\_weblog\_origin a***  ***LATERAL VIEW parse\_url\_tuple (***  ***regexp\_replace (http\_referer, "\"", ""),***  ***'HOST',***  ***'PATH',***  ***'QUERY',***  ***'QUERY:url'***  ***) b***  ***AS***  ***ref\_host,***  ***ref\_path,***  ***ref\_query,***  ***ref\_query\_url***  ***) c;*** |

***(当最终的报表指标有明显的不合法指标，这时就需要数据质量检查：写脚本来查看最后的结果表中某些字段的值是否合法)***

# 七、模块开发四：业务指标统计

***注：每一种统计指标都可以跟各维度表进行叉乘，从而得出各个维度的统计结果篇幅限制，叉乘的代码及注释信息详见项目工程代码文件***

***为了在前端展示时速度更快，每一个指标都事先算出各维度结果存入mysql***

**在实际生产中，究竟需要哪些统计指标通常由相关数据需求部门人员提出，而且会不断有新的统计需求产生，以下为网站流量分析中的一些典型指标示例。**

## 7.1 PV统计

### 7.1.1 多维度统计PV总量

#### 1、时间维度：

|  |
| --- |
| ***一、第一种：***  ***1、计算指定的某个小时pvs***  ***select count(\*),month,day,hour from webClick.ods\_weblog\_detail group by month,day,hour;***  ***\_c0 month day hour***  ***4 09 18 00***  ***2 09 18 02***  ***5 09 18 03***  ***1 09 18 04***  ***1 09 18 05***  ***13 09 18 07***  ***11 09 18 08***  ***14 09 18 09***  ***3 09 18 10***  ***2 09 18 11***  ***4 09 18 12***  ***5 09 18 13***  ***1 09 18 14***  ***2 09 18 15***  ***1 09 18 16***  ***4 09 18 17***  ***1 09 18 19***  ***1 09 18 22***  ***1 09 18 23***  ***2、结果表： 计算该处理批次（一天）中的各小时pvs***  ***drop table webClick.dw\_pvs\_hour;***  ***create table webClick.dw\_pvs\_hour(***  ***month string,***  ***day string,***  ***hour string,***  ***pvs bigint)***  ***partitioned by(datestr string);***  ***3、把查询结果插入到结果表中***  ***insert into table webClick.dw\_pvs\_hour partition(datestr='2013-09-18')***  ***select a.month as month,a.day as day,a.hour as hour,count(1) as pvs from webClick.ods\_weblog\_detail a***  ***where a.datestr='2013-09-18' group by a.month,a.day,a.hour;***  ***dw\_pvs\_hour.month dw\_pvs\_hour.day dw\_pvs\_hour.hour dw\_pvs\_hour.pvs dw\_pvs\_hour.datestr***  ***09 18 00 4 2013-09-18***  ***09 18 02 2 2013-09-18***  ***09 18 03 5 2013-09-18***  ***09 18 04 1 2013-09-18***  ***09 18 05 1 2013-09-18***  ***09 18 07 13 2013-09-18***  ***09 18 08 11 2013-09-18***  ***09 18 09 14 2013-09-18***  ***09 18 10 3 2013-09-18***  ***09 18 11 2 2013-09-18***  ***09 18 12 4 2013-09-18***  ***09 18 13 5 2013-09-18***  ***09 18 14 1 2013-09-18***  ***09 18 15 2 2013-09-18***  ***09 18 16 1 2013-09-18***  ***09 18 17 4 2013-09-18***  ***09 18 19 1 2013-09-18***  ***09 18 22 1 2013-09-18***  ***09 18 23 1 2013-09-18***  ***二、第二种：***  ***或者用时间维表关联，时间维度(按日、时为单位统计)***  ***drop table webClick.dw\_pvs\_day;***  ***create table webClick.dw\_pvs\_day(pvs bigint,month string,day string);***  ***select count(1) as pvs,a.month as month,a.day as day from webClick.dim\_time a***  ***join***  ***webClick.ods\_weblog\_detail b***  ***on***  ***b.daystr="2013-09-18" and a.month=b.month and a.day=b.day***  ***group by a.month,a.day;***  ***insert into table webClick.dw\_pvs\_day***  ***select count(1) as pvs,a.month as month,a.day as day from webClick.dim\_time a***  ***join***  ***webClick.ods\_weblog\_detail b***  ***on***  ***b.daystr="2013-09-18" and a.month=b.month and a.day=b.day***  ***group by a.month,a.day;***  ***Insert into table webClick.dw\_pvs\_day***  ***Select sum(pvs) as pvs,month,day from dw\_pvs\_hour group by month,day having day=18;***   1. **时间维度：按月统计：**   ***select count(\*) as pvs,a.month from dim\_time a***  ***join ods\_weblog\_detail b on a.month=b.month group by a.month;***  ***pvs a.month***   1. ***09***   ***drop table t\_display\_pv\_month;***  ***create table t\_display\_pv\_month (pvs bigint,month string);***  ***insert into table t\_display\_pv\_month***  ***select count(\*) as pvs,a.month from dim\_time a***  ***join ods\_weblog\_detail b on a.month=b.month group by a.month;***  ***t\_display\_pv\_month.pvs t\_display\_pv\_month.month***  ***1824 09*** |

#### 2、按终端维度统计pv总量：

**注：探索数据中的终端类型：**

|  |
| --- |
| ***select distinct(http\_referer) from ods\_weblog\_detail where http\_user\_agent like '%Mozilla%' ;***  ***Select count(\*) as pvs , a.month from ods\_weblog\_detail a where http\_user\_agent like '%Mozilla%' group by a.month;***  ***pvs a.month***   1. ***09***   ***Select count(\*) as pvs , a.month from ods\_weblog\_detail a where http\_user\_agent like '%Chrome%' group by a.month;***  ***pvs a.month***  ***30 09***  ***Select count(\*) as pvs , a.month from ods\_weblog\_detail a where http\_user\_agent like '%Safari%' group by a.month;***  ***pvs a.month***  ***32 09*** |

**终端维度：uc （新浪UC）**

|  |
| --- |
| ***drop table uc;***  ***create table uc (pvs int,month string) partitioned by(datestr string);***  ***insert into table uc partition(datestr='2013-09')***  ***Select count(\*) as pvs , a.month from ods\_weblog\_detail a where http\_user\_agent like '%Mozilla%' group by a.month;*** |

**终端维度：chrome （浏览器）**

|  |
| --- |
| ***drop table chrome;***  ***create table chrome (pvs bigint,month string) partitioned by(datestr string);***  ***insert into table chrome partition(datestr='2013-09')***  ***Select count(\*) as pvs , a.month from ods\_weblog\_detail a where http\_user\_agent like '%Chrome%' group by a.month;*** |

**终端维度：safari （Safari浏览器 苹果）**

|  |
| --- |
| ***drop table safari;***  ***create table safari (pvs bigint,month string) partitioned by(datestr string);***  ***insert into table safari partition(datestr='2013-09')***  ***Select count(\*) as pvs , a.month from ods\_weblog\_detail a where http\_user\_agent like '%Safari%' group by a.month;*** |

#### 3、 按栏目维度统计pv总量

**栏目维度：job**

**栏目维度：news**

**栏目维度：bargin**

**栏目维度：lane**

### 7.1.2 人均浏览页数：

**需求描述：比如，今日所有来访者，平均请求的页面数；**

**人均浏览页数 = 总页面请求数/去重总人数**

|  |
| --- |
| ***drop table dw\_avgpv\_user\_d;***  ***create table dw\_avgpv\_user\_d(***  ***day string,***  ***avgpv string);***  ***select***  ***'2013-09-18' ,***  ***sum(b.pvs)/count(b.remote\_addr)***  ***from***  ***(select remote\_addr , count(1) as pvs from ods\_weblog\_detail where datestr='2013-09-18'***  ***group by remote\_addr) b;***  ***\_c0 \_c1***  ***2013-09-18 1.4339622641509433***  ***insert into table dw\_avgpv\_user\_d***  ***select '2013-09-18',sum(b.pvs)/count(b.remote\_addr) from***  ***(select remote\_addr,count(1) as pvs from ods\_weblog\_detail where datestr='2013-09-18' group by remote\_addr) b;*** |

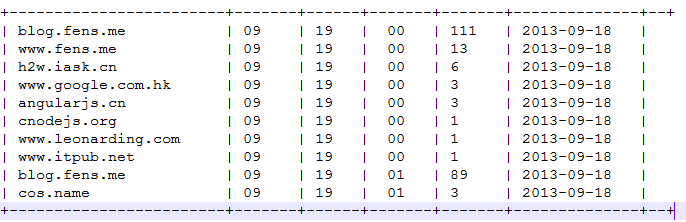
### 7.1.3 按referer维度统计pv总量：

**需求：按照来源及时间维度统计PVS，并按照PV大小倒序排序；**

**-- 按照小时粒度统计，查询结果存入：( "*dw\_pvs\_referer\_h*" )**

|  |
| --- |
| ***drop table dw\_pvs\_referer\_h;***  ***create table dw\_pvs\_referer\_h(***  ***referer\_url string,***  ***referer\_host string,***  ***month string,***  ***day string,***  ***hour string,***  ***pv\_referer\_cnt bigint)***  ***partitioned by(datestr string);***  ***select***  ***http\_referer,***  ***ref\_host,***  ***month,***  ***day,***  ***hour,***  ***count(1) as pv\_referer\_cnt***  ***from ods\_weblog\_detail***  ***group by http\_referer,ref\_host,month,day,hour***  ***having ref\_host is not null***  ***order by hour asc,day asc,month asc,pv\_referer\_cnt desc;***  ***http\_referer ref\_host month day hour pv\_referer\_cnt***  ***"https://www.google.com.hk/" www.google.com.hk 09 18 00 1***  ***"http://h2w.iask.cn/jump.php?url=http%3A%2F%2Fblog.fens.me%2Fhadoop-mahout-roadmap%2F" h2w.iask.cn 0***   1. ***18 00 1***   ***"http://blog.fens.me/series-it-finance/" blog.fens.me 09 18 02 1***  ***"https://www.google.com/" www.google.com 09 18 02 1***  ***"http://f.dataguru.cn/thread-175501-1-1.html" f.dataguru.cn 09 18 03 1***  ***"http://blog.fens.me/hadoop-family-roadmap/" blog.fens.me 09 18 03 1***  ***"http://blog.fens.me/hadoop-mahout-roadmap/" blog.fens.me 09 18 03 1***  ***"http://blog.fens.me/nodejs-crawler-douban/" blog.fens.me 09 18 03 1***  ***"http://blog.fens.me/series-hadoop-family/" blog.fens.me 09 18 07 2***  ***"https://www.google.com.hk/" www.google.com.hk 09 18 07 2***  ***insert into table dw\_pvs\_referer\_h partition(datestr='2013-09-18')***  ***select http\_referer,ref\_host,month,day,hour,count(1) as pv\_referer\_cnt***  ***from ods\_weblog\_detail***  ***group by http\_referer,ref\_host,month,day,hour***  ***having ref\_host is not null***  ***order by hour asc,day asc,month asc,pv\_referer\_cnt desc;*** |

***注：还可以按来源地域维度、访客终端维度等计算***

****

### 7.1.4 统计pv总量最大的来源TOPN (取分组TOP)：

**需求描述：按照时间维度，比如，统计一天内各小时产生最多pvs的来源topN**

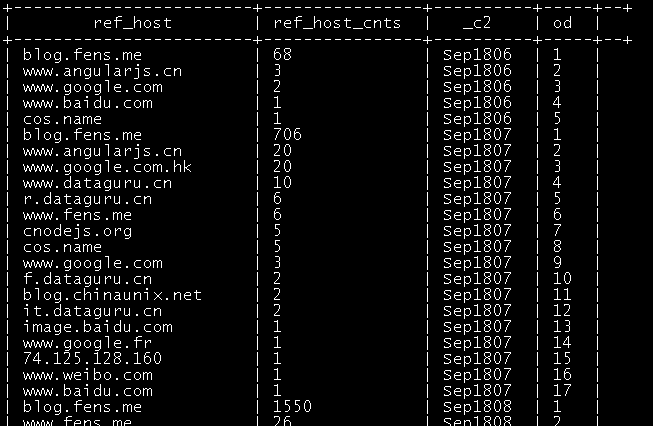
**需要用到row\_number函数；**

|  |
| --- |
| **Row\_nomber 实验：**   1. **数据：**   **用户信息表：**  **1,man,18,打游戏**  **2,man,20,谈恋爱**  **3,gril,16,学习**  **4,gril,20,打扮**  **5,man,30,工作**  **需求：每种性别，年龄最大的两个人；**  **分析：**   1. **先按照性别分组：**   **Select id,sex,age,like from userInfo group by sex order by age;**  *row\_number() over (partition by 分组条件字段 order by 排序条件字段) as 新增列名：*  *按照分组条件字段进行标记序号，同时按照排序条件字段进行排序；*  **1,man,18,打游戏 1**  **2,man,20,谈恋爱 2**  **5,man,30,工作 3**  **3,gril,16,学习 1**  **4,gril,20,打扮 2**   1. **hql语句：**   ***CREATE TABLE t\_userInfo (***  ***id INT,***  ***sex String,***  ***age INT,***  ***likes string***  ***) ROW format delimited FIELDS TERMINATED BY ',';***  ***LOAD DATA LOCAL inpath '/root/uek/bigdataApplication/hive-2.3.3/testDir/userInfo.txt' INTO TABLE t\_userInfo;***  ***SELECT***  ***id,***  ***sex,***  ***age,***  ***likes,***  ***row\_number () over (***  ***PARTITION BY sex***  ***ORDER BY***  ***age DESC***  ***) AS number***  ***FROM***  ***t\_userInfo;***  ***SELECT***  ***id,***  ***sex,***  ***age,***  ***likes***  ***FROM***  ***(***  ***SELECT***  ***id,***  ***sex,***  ***age,***  ***likes,***  ***row\_number () over (***  ***PARTITION BY sex***  ***ORDER BY***  ***age DESC***  ***) AS number***  ***FROM***  ***t\_userInfo***  ***) t\_tmp***  ***WHERE***  ***t\_tmp.number <= 2;***   1. ***结果：***   ***id sex age likes***  ***4 gril 20 打扮***  ***3 gril 16 学习***  ***5 man 30 工作***  ***2 man 20 谈恋爱*** |

**以下语句对每个小时内的来访host次数倒序排序标号：**

|  |
| --- |
| ***create table dw\_pvs\_referer\_h(***  ***referer\_url string,***  ***referer\_host string,***  ***month string,***  ***day string,***  ***hour string,***  ***pv\_referer\_cnt bigint)***  ***partitioned by(datestr string);***  *select*  ***referer\_host****,*  ***pv\_referer\_cnt****,*  *concat(month,day,hour), --字符串连接函数*  *row\_number() over (partition by concat(month,day,hour) order by* ***pv\_referer\_cnt*** *desc) as od*  *from* ***dw\_pvs\_referer\_h;***  ***referer\_host pv\_referer\_cnt \_c2 od***  ***h2w.iask.cn 1 0918 00 1***  ***www.google.com.hk 1 0918 00 2***  ***blog.fens.me 1 0918 02 1***  ***www.google.com 1 0918 02 2***  ***blog.fens.me 1 0918 03 1***  ***f.dataguru.cn 1 0918 03 2***  ***blog.fens.me 1 0918 03 3***  ***blog.fens.me 1 0918 03 4***  ***blog.fens.me 2 0918 07 1***  ***www.google.com.hk 2 0918 07 2***  ***f.dataguru.cn 1 0918 07 3***  ***www.google.com.hk 1 0918 07 4***  ***www.google.com.hk 1 0918 07 5***  ***www.google.com 1 0918 07 6***  ***blog.fens.me 2 0918 08 1***  ***blog.fens.me 1 0918 08 2***  ***blog.fens.me 1 0918 08 3***  ***blog.fens.me 1 0918 08 4***  ***blog.fens.me 1 0918 08 5***  ***f.dataguru.cn 1 0918 08 6***  ***www.dataguru.cn 1 0918 08 7***  ***www.google.com.tw 1 0918 08 8***  ***www.google.com.hk 1 0918 08 9***  ***redir.yy.duowan.com 3 0918 09 1***  ***www.google.com.hk 2 0918 09 2***  ***www.baidu.com 1 0918 09 3***  ***www.baidu.com 1 0918 09 4***  ***www.dataguru.cn 1 0918 09 5***  ***blog.fens.me 1 0918 09 6***  ***blog.fens.me 1 0918 09 7***  ***blog.fens.me 1 0918 09 8***  ***blog.fens.me 1 0918 10 1***  ***f.dataguru.cn 1 0918 10 2***  ***www.google.com.hk 1 0918 10 3***  ***blog.fens.me 1 0918 11 1***  ***www.google.com.hk 1 0918 11 2***  ***blog.fens.me 1 0918 12 1***  ***f.dataguru.cn 1 0918 12 2***  ***blog.fens.me 1 0918 12 3***  ***f.dataguru.cn 1 0918 13 1***  ***blog.fens.me 1 0918 14 1***  ***www.google.com.tw 1 0918 19 1***  ***blog.fens.me 1 0918 23 1*** |

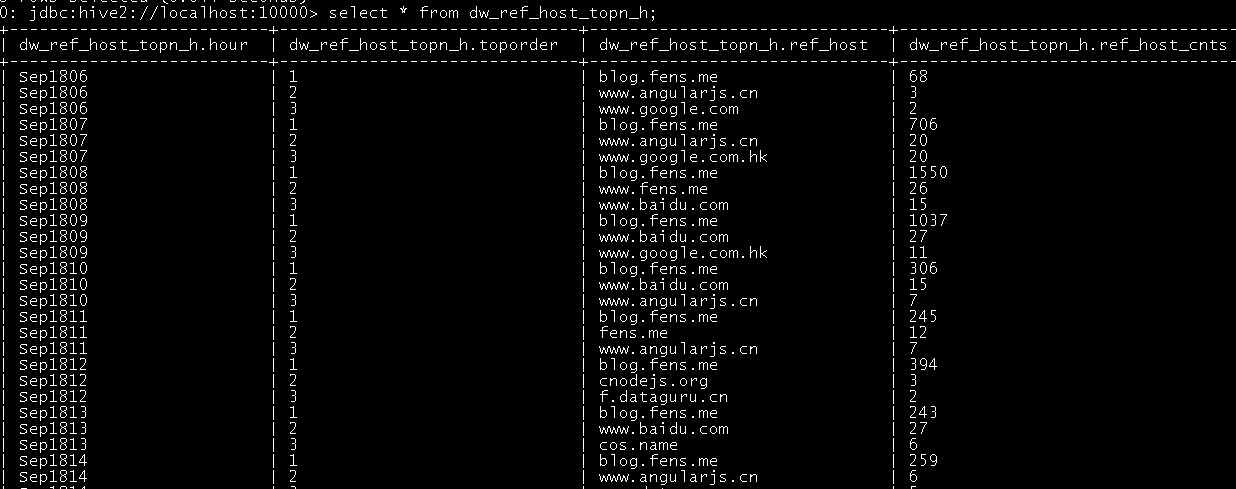
**效果如下：**

****

**根据上述row\_number的功能，可编写Hql取各小时的ref\_host访问次数topn**

|  |
| --- |
| ***drop table dw\_pvs\_refhost\_topn\_h;***  ***create table dw\_pvs\_refhost\_topn\_h(***  ***hour string,***  ***toporder string,***  ***ref\_host string,***  ***ref\_host\_cnts string***  ***) partitioned by(datestr string);***  ***select***  ***t.hour,***  ***t.od,***  ***t.ref\_host,***  ***t.ref\_host\_cnts***  ***From***  ***(select***  ***ref\_host,***  ***ref\_host\_cnts,***  ***concat(month,day,hour) as hour,***  ***row\_number() over (partition by concat(month,day,hour) order by ref\_host\_cnts desc) as od***  ***from dw\_pvs\_referer\_h) t***  ***where od<=3;***  ***select***  ***t.hour,***  ***t.od,***  ***t.referer\_host,***  ***t.pv\_referer\_cnt***  ***From***  ***(****select*  ***referer\_host****,*  ***pv\_referer\_cnt****,*  *concat(month,day,hour) as hour,*  *row\_number() over (partition by concat(month,day,hour) order by* ***pv\_referer\_cnt*** *desc) as od*  *from* ***dw\_pvs\_referer\_h***  ***) t***  ***where od<=3;***  ***t.hour t.od t.referer\_host t.pv\_referer\_cnt***  ***0918 00 1 h2w.iask.cn 1***  ***0918 00 2 www.google.com.hk 1***  ***0918 02 1 blog.fens.me 1***  ***0918 02 2 www.google.com 1***  ***0918 03 1 blog.fens.me 1***  ***0918 03 2 f.dataguru.cn 1***  ***0918 03 3 blog.fens.me 1***  ***0918 07 1 blog.fens.me 2***  ***0918 07 2 www.google.com.hk 2***  ***0918 07 3 f.dataguru.cn 1***  ***0918 08 1 blog.fens.me 2***  ***0918 08 2 blog.fens.me 1***  ***0918 08 3 blog.fens.me 1***  ***0918 09 1 redir.yy.duowan.com 3***  ***0918 09 2 www.google.com.hk 2***  ***0918 09 3 www.baidu.com 1***  ***0918 10 1 blog.fens.me 1***  ***0918 10 2 f.dataguru.cn 1***  ***0918 10 3 www.google.com.hk 1***  ***0918 11 1 blog.fens.me 1***  ***0918 11 2 www.google.com.hk 1***  ***0918 12 1 blog.fens.me 1***  ***0918 12 2 f.dataguru.cn 1***  ***0918 12 3 blog.fens.me 1***  ***0918 13 1 f.dataguru.cn 1***  ***0918 14 1 blog.fens.me 1***  ***0918 19 1 www.google.com.tw 1***  ***0918 23 1 blog.fens.me 1***  ***insert into table webClick.dw\_pvs\_refhost\_topn\_h partition(datestr='2013-09-18')***  ***select***  ***t.hour,***  ***t.od,***  ***t.referer\_host,***  ***t.pv\_referer\_cnt***  ***From***  ***(****select*  ***referer\_host****,*  ***pv\_referer\_cnt****,*  *concat(month,day,hour) as hour,*  *row\_number() over (partition by concat(month,day,hour) order by* ***pv\_referer\_cnt*** *desc) as od*  *from* ***dw\_pvs\_referer\_h***  ***) t***  ***where od<=3;*** |

**结果如下：**

****

***注：还可以按来源地域维度、访客终端维度等计算***

## 7.2 受访分析:

**统计每日最热门的页面top10:**

|  |
| --- |
| *drop table dw\_pvs\_d;*  *create table dw\_pvs\_d(day string,url string,pvs string);*  *SELECT*  *'2013-09-18',*  *a.request,*  *a.request\_counts*  *FROM*  *(*  *SELECT*  *request AS request,*  *count(request) AS request\_counts*  *FROM*  *ods\_weblog\_detail*  *WHERE*  *datestr = '2013-09-18'*  *GROUP BY*  *request*  *HAVING*  *request IS NOT NULL*  *) a*  *ORDER BY*  *a.request\_counts DESC*  *LIMIT 10;*  *2013-09-18 /hadoop-mahout-roadmap/ 20*  *2013-09-18 /hadoop-hive-intro/ 14*  *2013-09-18 /finance-rhive-repurchase/ 13*  *2013-09-18 /hadoop-family-roadmap/ 13*  *2013-09-18 /hadoop-zookeeper-intro/ 6*  *2013-09-18 /about 5*  *2013-09-18 /cassandra-clustor/ 3*  *2013-09-18 /black-ip-list/ 2*  *insert into table dw\_pvs\_d*  *SELECT*  *'2013-09-18',*  *a.request,*  *a.request\_counts*  *FROM*  *(*  *SELECT*  *request AS request,*  *count(request) AS request\_counts*  *FROM*  *ods\_weblog\_detail*  *WHERE*  *datestr = '2013-09-18'*  *GROUP BY*  *request*  *HAVING*  *request IS NOT NULL*  *) a*  *ORDER BY*  *a.request\_counts DESC*  *LIMIT 10;* |

***注：还可继续得出各维度交叉结果***

## 7.3 访客分析

### 7.3.1 独立访客:

**需求描述：按照时间维度比如小时来统计独立访客及其产生的pvCnts;**

***对于独立访客的识别，如果在原始日志中有用户标识，则根据用户标识即很好实现;***

***此处，由于原始日志中并没有用户标识，以访客IP来模拟，技术上是一样的，只是精确度相对较低:***

**时间维度：时**

|  |
| --- |
| *DROP TABLE dw\_user\_dstc\_ip\_h;*  *CREATE TABLE dw\_user\_dstc\_ip\_h (*  *remote\_addr string,*  *pvs BIGINT,*  *HOUR string*  *);*  *SELECT*  *remote\_addr,*  *count(1) AS pvs,*  *concat(MONTH, DAY, HOUR) AS HOUR*  *FROM*  *ods\_weblog\_detail*  *WHERE*  *datestr = '2013-09-18'*  *GROUP BY*  *concat(MONTH, DAY, HOUR),*  *remote\_addr;*  *remote\_addr pvs hour*  *113.90.232.163 1 0918 00*  *123.116.73.157 1 0918 00*  *208.115.113.90 1 0918 00*  *221.179.193.202 1 0918 00*  *122.95.27.176 1 0918 02*  *220.181.126.4 1 0918 02*  *220.181.89.174 1 0918 03*  *222.70.152.149 3 0918 03*  *71.206.247.97 1 0918 03*  *220.181.89.174 1 0918 04*  *157.56.92.147 1 0918 05*  *1.80.249.223 1 0918 07*  *111.193.224.9 1 0918 07*  *114.221.170.42 1 0918 07*  *121.8.153.6 1 0918 07*  *14.17.29.86 1 0918 07*  *162.105.203.28 1 0918 07*  *183.60.9.217 1 0918 07*  *202.99.27.196 2 0918 07*  *211.155.231.180 1 0918 07*  *61.148.244.188 1 0918 07*  *66.249.66.84 1 0918 07*  *72.46.128.140 1 0918 07*  *112.65.193.16 1 0918 08*  *116.24.236.137 3 0918 08*  *159.226.251.170 1 0918 08*  *204.89.152.91 2 0918 08*  *218.241.103.162 1 0918 08*  *60.10.8.5 2 0918 08*  *60.247.77.253 1 0918 08*  *101.226.167.201 1 0918 09*  *101.226.167.205 1 0918 09*  *113.107.237.31 1 0918 09*  *123.125.199.104 1 0918 09*  *180.153.163.190 1 0918 09*  *183.60.177.228 5 0918 09*  *218.18.232.228 1 0918 09*  *61.135.190.103 1 0918 09*  *61.135.190.104 1 0918 09*  *61.135.190.222 1 0918 09*  *101.226.169.215 1 0918 10*  *118.195.70.197 1 0918 10*  *58.246.170.210 1 0918 10*  *182.148.111.151 1 0918 11*  *58.100.63.173 1 0918 11*  *111.161.17.104 1 0918 12*  *115.236.11.194 1 0918 12*  *183.12.96.73 1 0918 12*  *66.249.66.84 1 0918 12*  *110.211.10.14 1 0918 13*  *174.120.8.226 4 0918 13*  *113.90.83.205 1 0918 14*  *220.181.89.174 2 0918 15*  *220.181.89.174 1 0918 16*  *208.43.225.84 3 0918 17*  *220.181.89.174 1 0918 17*  *195.110.138.105 1 0918 19*  *66.249.66.84 1 0918 22*  *113.90.48.158 1 0918 23*  *INSERT INTO TABLE dw\_user\_dstc\_ip\_h*  *SELECT*  *remote\_addr,*  *count(1) AS pvs,*  *concat(MONTH, DAY, HOUR) AS HOUR*  *FROM*  *ods\_weblog\_detail*  *WHERE*  *datestr = '2013-09-18'*  *GROUP BY*  *concat(MONTH, DAY, HOUR),*  *remote\_addr;* |

**在此结果表之上，可以进一步统计出，每小时独立访客总数，每小时请求次数topn访客等;**

**每小时独立访客总数：**

|  |
| --- |
| *SELECT*  *count(1) AS dstc\_ip\_cnts,*  *HOUR*  *FROM*  *dw\_user\_dstc\_ip\_h*  *GROUP BY*  *HOUR;*  *dstc\_ip\_cnts hour*  *4 0918 00*  *2 0918 02*  *3 0918 03*  *1 0918 04*  *1 0918 05*  *12 0918 07*  *7 0918 08*  *10 0918 09*  *3 0918 10*  *2 0918 11*  *4 0918 12*  *2 0918 13*  *1 0918 14*  *1 0918 15*  *1 0918 16*  *2 0918 17*  *1 0918 19*  *1 0918 22*  *1 0918 23* |

**时间维度：月**

|  |
| --- |
| *SELECT*  *remote\_addr,*  *count(1) AS counts,*  *MONTH*  *FROM*  *ods\_weblog\_detail*  *GROUP BY*  *MONTH,*  *remote\_addr;*  *remote\_addr counts month*  *1.80.249.223 1 09*  *101.226.167.201 1 09*  *101.226.167.205 1 09*  *101.226.169.215 1 09*  *110.211.10.14 1 09*  *111.161.17.104 1 09*  *111.193.224.9 1 09*  *112.65.193.16 1 09*  *113.107.237.31 1 09*  *113.90.232.163 1 09*  *113.90.48.158 1 09*  *113.90.83.205 1 09*  *114.221.170.42 1 09*  *115.236.11.194 1 09*  *116.24.236.137 3 09*  *118.195.70.197 1 09*  *121.8.153.6 1 09*  *122.95.27.176 1 09*  *123.116.73.157 1 09*  *123.125.199.104 1 09*  *14.17.29.86 1 09*  *157.56.92.147 1 09*  *159.226.251.170 1 09*  *162.105.203.28 1 09*  *174.120.8.226 4 09*  *180.153.163.190 1 09*  *182.148.111.151 1 09*  *183.12.96.73 1 09*  *183.60.177.228 5 09*  *183.60.9.217 1 09*  *195.110.138.105 1 09*  *202.99.27.196 2 09*  *204.89.152.91 2 09*  *208.115.113.90 1 09*  *208.43.225.84 3 09*  *211.155.231.180 1 09*  *218.18.232.228 1 09*  *218.241.103.162 1 09*  *220.181.126.4 1 09*  *220.181.89.174 6 09*  *221.179.193.202 1 09*  *222.70.152.149 3 09*  *58.100.63.173 1 09*  *58.246.170.210 1 09*  *60.10.8.5 2 09*  *60.247.77.253 1 09*  *61.135.190.103 1 09*  *61.135.190.104 1 09*  *61.135.190.222 1 09*  *61.148.244.188 1 09*  *66.249.66.84 3 09*  *71.206.247.97 1 09*  *72.46.128.140 1 09* |

**时间维度：日**

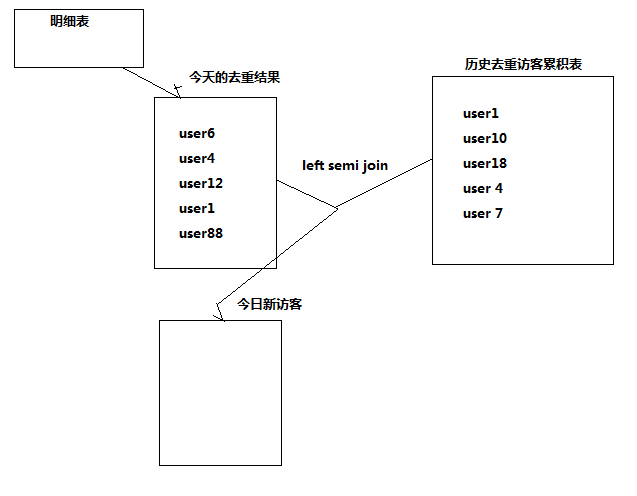
|  |
| --- |
| *SELECT*  *remote\_addr,*  *count(1) AS counts,*  *concat(MONTH, DAY) AS DAY*  *FROM*  *ods\_weblog\_detail*  *WHERE*  *datestr='2013-09-18'*  *GROUP BY*  *concat(MONTH, DAY),*  *remote\_addr;*  *remote\_addr counts day*  *1.80.249.223 1 0918*  *101.226.167.201 1 0918*  *101.226.167.205 1 0918*  *101.226.169.215 1 0918*  *110.211.10.14 1 0918*  *111.161.17.104 1 0918*  *111.193.224.9 1 0918*  *112.65.193.16 1 0918*  *113.107.237.31 1 0918*  *113.90.232.163 1 0918*  *113.90.48.158 1 0918*  *113.90.83.205 1 0918*  *114.221.170.42 1 0918*  *115.236.11.194 1 0918*  *116.24.236.137 3 0918*  *118.195.70.197 1 0918*  *121.8.153.6 1 0918*  *122.95.27.176 1 0918*  *123.116.73.157 1 0918*  *123.125.199.104 1 0918*  *14.17.29.86 1 0918*  *157.56.92.147 1 0918*  *159.226.251.170 1 0918*  *162.105.203.28 1 0918*  *174.120.8.226 4 0918*  *180.153.163.190 1 0918*  *182.148.111.151 1 0918*  *183.12.96.73 1 0918*  *183.60.177.228 5 0918*  *183.60.9.217 1 0918*  *195.110.138.105 1 0918*  *202.99.27.196 2 0918*  *204.89.152.91 2 0918*  *208.115.113.90 1 0918*  *208.43.225.84 3 0918*  *211.155.231.180 1 0918*  *218.18.232.228 1 0918*  *218.241.103.162 1 0918*  *220.181.126.4 1 0918*  *220.181.89.174 6 0918*  *221.179.193.202 1 0918*  *222.70.152.149 3 0918*  *58.100.63.173 1 0918*  *58.246.170.210 1 0918*  *60.10.8.5 2 0918*  *60.247.77.253 1 0918*  *61.135.190.103 1 0918*  *61.135.190.104 1 0918*  *61.135.190.222 1 0918*  *61.148.244.188 1 0918*  *66.249.66.84 3 0918*  *71.206.247.97 1 0918*  *72.46.128.140 1 0918* |

***注：还可以按来源地域维度、访客终端维度等计算***

### 7.3.2 每日新访客

**需求描述：将每天的新访客统计出来**

**实现思路：创建一个去重访客累积表，然后将每日访客对比累积表**

****

**时间维度：日**

|  |
| --- |
| *--历日去重访客累积表*  *drop table dw\_user\_dsct\_history;*  *create table dw\_user\_dsct\_history(*  *ip string ,*  *day string*  *)*  *partitioned by(datestr string);*  *Insert into table dw\_user\_dsct\_history partition(datestr='2013-09-18')*  *SELECT*  *DISTINCT(remote\_addr) ,*  *"2013-09-18" AS DAY*  *FROM*  *ods\_weblog\_detail*  *WHERE*  *datestr = "2013-09-18"*  *Limit 10;*  *历史用户：*  *SELECT*  *today. DAY AS DAY,*  *today.remote\_addr AS today\_addr,*  *old.ip AS old\_addr*  *FROM*  *(*  *SELECT*  *DISTINCT(remote\_addr) ,*  *"2013-09-18" AS DAY*  *FROM*  *ods\_weblog\_detail*  *WHERE*  *datestr = "2013-09-18"*  *) today*  *LEFT OUTER JOIN dw\_user\_dsct\_history old*  *ON today.remote\_addr = old.ip;*  *新用户：*  *SELECT*  *tmp. DAY AS DAY,*  *tmp.today\_addr AS new\_ip*  *FROM*  *(*  *SELECT*  *today. DAY AS DAY,*  *today.remote\_addr AS today\_addr,*  *old.ip AS old\_addr*  *FROM*  *(*  *SELECT*  *DISTINCT(remote\_addr) ,*  *"2013-09-18" AS DAY*  *FROM*  *ods\_weblog\_detail*  *WHERE*  *datestr = "2013-09-18"*  *) today*  *LEFT OUTER JOIN dw\_user\_dsct\_history old*  *ON today.remote\_addr = old.ip*  *) tmp*  *WHERE*  *tmp.old\_addr IS NULL;*  *--每日新用户追加到累计表:*  *INSERT INTO TABLE dw\_user\_dsct\_history PARTITION (datestr = '2013-09-19')*  *SELECT*  *tmp. DAY AS DAY,*  *tmp.today\_addr AS new\_ip*  *FROM*  *(*  *SELECT*  *today. DAY AS DAY,*  *today.remote\_addr AS today\_addr,*  *old.ip AS old\_addr*  *FROM*  *(*  *SELECT*  *DISTINCT(remote\_addr) ,*  *"2013-09-18" AS DAY*  *FROM*  *ods\_weblog\_detail*  *WHERE*  *datestr = "2013-09-18"*  *) today*  *LEFT OUTER JOIN dw\_user\_dsct\_history old*  *ON today.remote\_addr = old.ip*  *) tmp*  *WHERE*  *tmp.old\_addr IS NULL;* |

**验证：**

|  |
| --- |
| *select count(distinct remote\_addr) from ods\_weblog\_detail;*  *-- 1005*  *select count(1) from dw\_user\_dsct\_history where prtflag\_day='18/Sep/2013';*  *--845*  *select count(1) from dw\_user\_dsct\_history where prtflag\_day='19/Sep/2013';*  *--160* |

**时间维度：月**

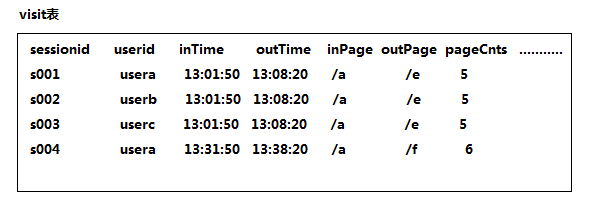
|  |
| --- |
| ***类似日粒度算法*** |

***注：还可以按来源地域维度、访客终端维度等计算***

## 7.4 Visit分析（用户画像模型）

### 7.4.1 回头/单次访客统计

**需求描述：查询今日所有回头访客及其访问次数:**

****

**实现思路：上表中出现次数>1的访客，即回头访客；反之，则为单次访客:**

|  |
| --- |
| *DROP TABLE dw\_user\_returning;*  *CREATE TABLE dw\_user\_returning (*  *DAY string,*  *remote\_addr string,*  *acc\_cnt string*  *) partitioned BY (datestr string);*  *每个ip下所产生的session数量；*  *SELECT*  *'2013-09-18' AS DAY,*  *remote\_addr,*  *count(SESSION) AS acc\_cnt*  *FROM*  *ods\_click\_stream\_visit*  *GROUP BY*  *remote\_addr;*  *每个ip下所产生的session数量大于1，就代表是回头客；*  *SELECT*  *tmp. DAY,*  *tmp.remote\_addr,*  *tmp.acc\_cnt*  *FROM*  *(*  *SELECT*  *'2013-09-18' AS DAY,*  *remote\_addr,*  *count(SESSION) AS acc\_cnt*  *FROM*  *ods\_click\_stream\_visit*  *GROUP BY*  *remote\_addr*  *) tmp*  *WHERE*  *tmp.acc\_cnt > 1;*  *tmp.day tmp.remote\_addr tmp.acc\_cnt*  *2013-09-18 220.181.89.174 3*  *2013-09-18 66.249.66.84 3*  *INSERT overwrite TABLE dw\_user\_returning PARTITION (datestr = '2013-09-18')*  *SELECT*  *tmp. DAY,*  *tmp.remote\_addr,*  *tmp.acc\_cnt*  *FROM*  *(*  *SELECT*  *'2013-09-18' AS DAY,*  *remote\_addr,*  *count(SESSION) AS acc\_cnt*  *FROM*  *ods\_click\_stream\_visit*  *GROUP BY*  *remote\_addr*  *) tmp*  *WHERE*  *tmp.acc\_cnt > 1;* |

### 7.4.2 人均访问频次

**需求：统计出每天所有用户访问网站的平均次数（visit）**

**总pv数/去重总用户数**

|  |
| --- |
| *select sum(pagevisits)/count(distinct remote\_addr) from ods\_click\_stream\_visit;*  *\_c0*  *1.4339622641509433* |

## 7.5 关键路径转化率分析——漏斗模型

### 7.5.0 数据生成：

|  |
| --- |
| *#!/bin/python*  *#coding=utf-8*  *import time*  *import datetime*  *import random*  *# 日志格式*  *# valid*  *# remote-addr*  *# remote-user*  *# time\_local*  *# request*  *# status*  *# byte\_sent*  *# refer*  *# user-agent*  *# 本程序用于生成web访问日志 #######*  *# ''.join([(string.ascii\_letters+string.digits)[x] for x in random.sample(range(0,62),8)])*  *def random\_str(randomlength=8):*  *str = ''*  *chars = 'AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz0123456789'*  *length = len(chars) - 1*  *for i in range(randomlength):*  *str+=chars[random.randint(0, length)]*  *return str*  *# 页面：注册页、推荐页、相关页、商品页a、商品页b、商品页c、商品列表页、查询页、购物车页、订单查看页、订单提交页、首页、A类页面、B类页面、C类页面、D类页面；*  *pages=['/register','/recommand','/relation','/item/a','/item/b','/item/c','/list/','/search/','/cart','/order/getorder','/order/submitorder','/index','/category/a','/category/b','/category/c','/category/d']*  *for i in range(100):*  *pages.append("/item/"+random\_str(8))*  *refers=['http://www.google.com','www.baidu.com','www.sohu.com']*  *refers.extend(pages)*  *minutes=[]*  *for i in range(00,40):*  *t = i+random.randint(0,19)*  *minutes.append(str(t) if len(str(t))>1 else "0"+str(t))*  *print(minutes)*  *seconds=[]*  *for i in range(00,40):*  *t = i+random.randint(0,19)*  *seconds.append(str(t) if len(str(t))>1 else "0"+str(t))*  *print(minutes)*  *hours=[]*  *for i in range(00,24):*  *hours.append(str(i) if len(str(i))>1 else "0"+str(i))*  *print(hours)*  *ips = open('/root/uek/bigdataApplication/hive-2.3.3/testDir/000000\_0.ips')*  *file = ips.read()*  *ips = file.split('\n')*  *#*  *# morning=datetime.datetime.strptime('2013-09-18 00:00:00','%Y-%m-%d %H:%M:%S')*  *# evening=datetime.datetime.strptime('2013-09-18 23:59:59','%Y-%m-%d %H:%M:%S')*  *#*  *#*  *# print(morning)*  *# print(evening)*  *logfile = open("/root/uek/bigdataApplication/hive-2.3.3/testDir/mylog.log",mode='w')*  *for i in range(0,500000):*  *# seed = random.randint(0,7)*  *# print(pages[seed])*  *ip=random.choice(ips)*  *h=random.choice(hours)*  *m=random.choice(minutes)*  *s=random.choice(minutes)*  *page=random.choice(pages)*  *refer=random.choice(refers)*  *useragent = 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36'*  *body\_sent = '1800'*  *status='200'*  *valid = 'true'*  *user = '-'*  *day='2013-09-18'*  *time\_local= day + " " + h + ":" + m + ":" + s*  *logfile.write(valid + "\001" + ip + "\001"+ user+ "\001" + time\_local +"\001"+page+"\001"+ status +"\001"+ body\_sent + "\001"+refer+ "\001" + useragent +"\n")*  *logfile.close()* |

**(python脚本生成测试数据；mylog.txt)**

**转化：在一条指定的业务流程中，各个步骤的完成人数及相对上一个步骤的百分比**

### 7.5.1 需求分析：



### 7.5.2 模型设计：

**定义好业务流程中的页面标识，下例中的步骤为：**

|  |
| --- |
| ***Step1、 /item% 商品***  ***Step2、 /category 分类***  ***Step3、 /index 首页***  ***Step4、 /order 订单*** |

### 7.5.3 开发实现：

**分步骤开发：**

1. **查询每一个步骤的总访问人数：**

|  |
| --- |
| ***原始数据表：***  ***drop table if exists webClick.ods\_click\_pageviews\_loudoumoxing;***  ***create table webClick.ods\_click\_pageviews\_loudoumoxing(***  ***Session string, --session***  ***remote\_addr string, --ip***  ***remote\_user string, --用户信息***  ***time\_local string, --访问时间***  ***request string, --请求url***  ***visit\_step string, --step步骤***  ***page\_staylong string, --页面停留时间***  ***http\_referer string, --来访url***  ***http\_user\_agent string, --浏览器信息***  ***body\_bytes\_sent string, --访问内容大小***  ***status string --状态***  ***)***  ***partitioned by (datestr string)***  ***row format delimited***  ***fields terminated by '\001';***  ***load data local inpath '/root/uek/bigdataApplication/hive-2.3.3/testDir/mylog.log' into table webClick.ods\_click\_pageviews\_loudoumoxing partition(datestr='2013-09-18');***  **每一个步骤的总访问人数表：**  *Drop table dw\_oute\_numbs;*  *CREATE TABLE dw\_oute\_numbs AS*  *SELECT*  *'step1' AS step,*  *count(DISTINCT remote\_addr) AS numbs*  *FROM*  ***ods\_click\_pageviews\_loudoumoxing***  *WHERE*  *request LIKE '/item%'*  *UNION*  *SELECT*  *'step2' AS step,*  *count(DISTINCT remote\_addr) AS numbs*  *FROM*  ***ods\_click\_pageviews\_loudoumoxing***  *WHERE*  *request LIKE '/category%'*  *UNION*  *SELECT*  *'step3' AS step,*  *count(DISTINCT remote\_addr) AS numbs*  *FROM*  ***ods\_click\_pageviews\_loudoumoxing***  *WHERE*  *request LIKE '/order%'*  *UNION*  *SELECT*  *'step4' AS step,*  *count(DISTINCT remote\_addr) AS numbs*  *FROM*  ***ods\_click\_pageviews\_loudoumoxing***  *WHERE*  *request LIKE '/index%';*  *\_u3.step \_u3.numbs*  *step1 1029*  *step2 1029*  *step3 1029*  *step4 1022* |

1. **查询每一步骤相对于第一个步骤的比例:**

**思路：利用join , 自连接；**

|  |
| --- |
| *SELECT*  *rn.step AS rnstep,*  *rn.numbs AS rnnumbs,*  *rr.step AS rrstep,*  *rr.numbs AS rrnumbs*  *FROM*  *dw\_oute\_numbs rn*  *INNER JOIN dw\_oute\_numbs rr;*  *hive.strict.checks.cartesian.product = false; 该属性不允许笛卡尔积操作, 默认为true ,需要修改为false;*  *rnstep rnnumbs rrstep rrnumbs*  *Step1 1029 step1 1029 cast 函数：把字符串转换成int类型；*  *step2 1029 step1 1029*  *step3 1029 step1 1029*  *step4 1022 step1 1029*  *step1 1029 step2 1029*  *step2 1029 step2 1029*  *step3 1029 step2 1029*  *step4 1022 step2 1029*  *step1 1029 step3 1029*  *step2 1029 step3 1029*  *step3 1029 step3 1029*  *step4 1022 step3 1029*  *step1 1029 step4 1022*  *step2 1029 step4 1022*  *step3 1029 step4 1022*  *step4 1022 step4 1022* |

|  |
| --- |
| ***SELECT***  ***tmp.rnstep,***  ***tmp.rnnumbs / tmp.rrnumbs AS ratio***  ***FROM***  ***(***  ***SELECT***  ***rn.step AS rnstep,***  ***rn.numbs AS rnnumbs,***  ***rr.step AS rrstep,***  ***rr.numbs AS rrnumbs***  ***FROM***  ***dw\_oute\_numbs rn***  ***INNER JOIN dw\_oute\_numbs rr***  ***) tmp***  ***WHERE***  ***tmp.rrstep = 'step1';***  *tmp.rnstep ratio*  *step1 1.0*  *step2 1.0*  *step3 1.0*  *step4 0.9931972789115646* |

1. **查询每一步骤相对于上一步骤的漏出率**

|  |
| --- |
| *SELECT*  *tmp.rrstep AS rrstep,*  *tmp.rrnumbs / tmp.rnnumbs AS ration*  *FROM*  *(*  *SELECT*  *rn.step AS rnstep,*  *rn.numbs AS rnnumbs,*  *rr.step AS rrstep,*  *rr.numbs AS rrnumbs*  *FROM*  ***dw\_oute\_numbs*** *rn*  *INNER JOIN* ***dw\_oute\_numbs*** *rr*  *) tmp*  *WHERE*  *cast(substr(tmp.rnstep, 5, 1) AS INT) = cast(substr(tmp.rrstep, 5, 1) AS INT) - 1 当右边减1等于坐标*  *rrstep ration*  *step2 1.0*  *step3 1.0*  *step4 0.9931972789115646* |

1. **汇总以上两种指标**

|  |
| --- |
| *create table dw\_out\_ratio\_result*  *as*  *SELECT*  *abs.step,*  *abs.numbs,*  *abs.ratio AS abs\_ratio,*  *rel.ratio AS rel\_ratio*  *FROM*  *(*  *SELECT*  *tmp.rnstep AS step,*  *tmp.rnnumbs AS numbs,*  *tmp.rnnumbs / tmp.rrnumbs AS ratio*  *FROM*  *(*  *SELECT*  *rn.step AS rnstep,*  *rn.numbs AS rnnumbs,*  *rr.step AS rrstep,*  *rr.numbs AS rrnumbs*  *FROM*  ***dw\_oute\_numbs*** *rn*  *INNER JOIN* ***dw\_oute\_numbs*** *rr*  *) tmp*  *WHERE*  *tmp.rrstep = 'step1'*  *) abs*  *LEFT OUTER JOIN (*  *SELECT*  *tmp.rrstep AS step,*  *tmp.rrnumbs / tmp.rnnumbs AS ratio*  *FROM*  *(*  *SELECT*  *rn.step AS rnstep,*  *rn.numbs AS rnnumbs,*  *rr.step AS rrstep,*  *rr.numbs AS rrnumbs*  *FROM*  ***dw\_oute\_numbs*** *rn*  *INNER JOIN* ***dw\_oute\_numbs*** *rr*  *) tmp*  *WHERE*  *cast(substr(tmp.rnstep, 5, 1) AS INT) = cast(substr(tmp.rrstep, 5, 1) AS INT) - 1*  *) rel ON abs.step = rel.step;*  *abs.step abs.numbs abs\_ratio rel\_ratio*  *step1 1029 1.0 NULL*  *step2 1029 1.0 1.0*  *step3 1029 1.0 1.0*  *step4 1022 0.9931972789115646 0.9931972789115646* |

# 八、模块开发五：业务指标结果导出：

**报表统计结果，由sqoop从hive表中导出到mysql中：**

**hiveToMysql01.opt**

|  |
| --- |
| *export*  *--connect*  *jdbc:mysql://bigdata:3306/webClick*  *--username*  *root*  *--password*  *root*  *--export-dir*  */hive2.3.3\_workSpace/warehouse/webclick.db/dw\_pvs\_hour/datestr=2013-09-18*  *--table*  *dw\_pvs\_hour*  *--num-mappers*  *1*  *--input-fields-terminated-by*  *'\001'* |
| *sqoop --options-file* **hiveToMysql01.opt** | |

# 模块开发六：工作流调度：

|  |
| --- |
| *1、数据采集脚本：*  *脚本名：dataCollect.job*  *#数据采集:*  *type=command*  *command=/root/uek/bigdataApplication/flume-1.8.0/bin/flume-ng agent -n a1 -f dataCollect.conf -D flume.root.logger=INFO,console*  *2、数据预处理：*  *脚本名：WeblogPreProcess.job*  *#数据预处理：*  *type=command*  *command=/root/uek/bigdataApplication/hadoop-3.1.1/bin/hadoop jar preLog.jar com.webclick.mr.pre.WeblogPreProcess /project/webClick/logs/18-10-01 /project/webClick/preLog*  *脚本名：ClickStreamPageview.job*  *#pageviews模型构建：*  *type=command*  *dependencies=WeblogPreProcess*  *command=/root/uek/bigdataApplication/hadoop-3.1.1/bin/hadoop jar pageView.jar com.webclick.mr.PageviewMode.ClickStreamPageview /project/webClick/preLog /project/webClick/pageviewsModel*  *脚本名：ClickStreamVisit.job*  *#visit模型构建：*  *type=command*  *dependencies=ClickStreamPageview*  *command=/root/uek/bigdataApplication/hadoop-3.1.1/bin/hadoop jar visitView.jar com.webclick.mr.VisitMode.ClickStreamVisit /project/webClick/pageviewsModel /project/webClick/visitModel*  *3、数据仓库处理：*  *脚本名：loadDataToTable.job*  *#给事实表中添加数据：*  *type=command*  *command=/root/uek/bigdataApplication/hive-2.3.3/bin/hive -f loadDataToTable.sql*  *脚本名：loadDataToOds\_weblog\_detail.job*  *#给明细宽表中添加数据：*  *type=command*  *dependencies=loadDataToTable*  *command=/root/uek/bigdataApplication/hive-2.3.3/bin/hive -f loadDataToOds\_weblog\_detail.sql*  *脚本名：dw\_pvs\_hour.job*  *#统计每小时pvs：*  *type=command*  *dependencies=loadDataToOds\_weblog\_detail*  *command=/root/uek/bigdataApplication/hive-2.3.3/bin/hive -f dw\_pvs\_hour.sql*  *4、sqoop数据迁移：*  *脚本名：sqoop\_dw\_pvs\_hour.job*  *#统计每小时pvs：*  *type=command*  *command=/root/uek/bigdataApplication/sqoop-1.4.7/bin/sqoop --options-file sqoop\_dw\_pvs\_hour.opt* |

# 十、模块开发七：数据展示：

**在企业的数据分析系统中，前端展现工具有很多，**

* **独立部署专门系统的方式：以Business Objects(BO,Crystal Report),Heperion(Brio),Cognos等国外产品为代表的，它们的服务器是单独部署的，与应用程序之间通过某种协议沟通信息**
* **有WEB程序展现方式：通过独立的或者嵌入式的java web系统来读取报表统计结果，以网页的形式对结果进行展现，如，100%纯Java的润乾报表**

**本日志分析项目采用自己开发web程序展现的方式**

* **Web展现程序采用的技术框架：**

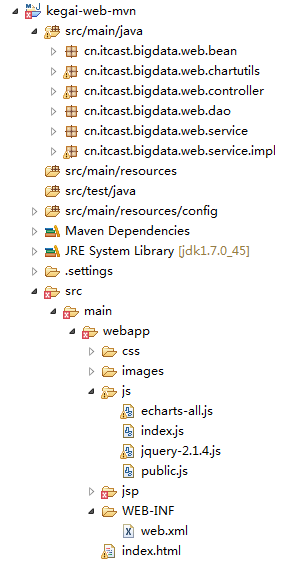
**Jquery + Echarts + springmvc + spring + mybatis + mysql**

* **展现的流程：**

1. **使用ssh从mysql中读取要展现的数据**
2. **使用json格式将读取到的数据返回给页面**
3. **在页面上用echarts对json解析并形成图标**

## Web程序工程结构

**采用maven管理工程，引入SSH框架依赖及jquery+echarts的js库**

****

## Web程序的实现代码

**采用典型的MVC架构实现**

|  |  |
| --- | --- |
| **页面** | **HTML + JQUERY + ECHARTS** |
| **Controller** | **SpringMVC** |
| **Service** | **Service** |
| **DAO** | **Mybatis** |
| **数据库** | **Mysql** |

***代码详情见项目工程***

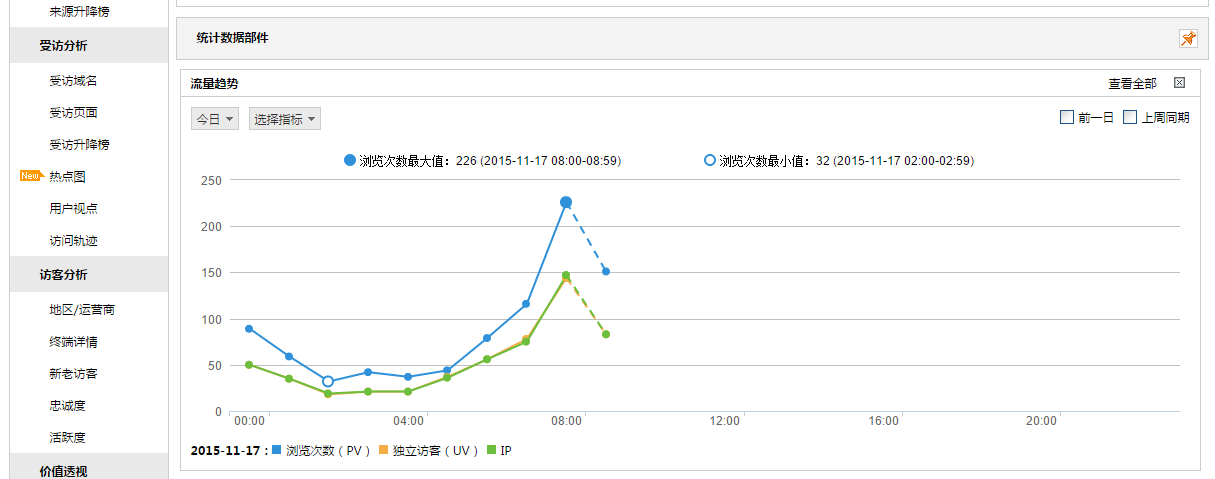
***代码示例：*ChartServiceImpl**

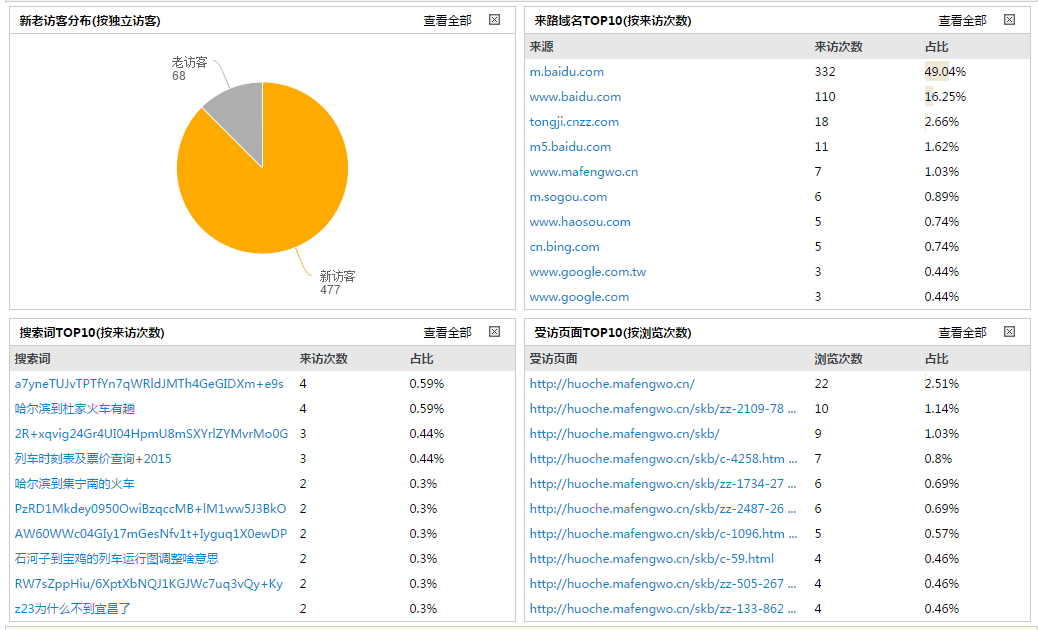
|  |
| --- |
| **@Service("chartService")**  **public class ChartServiceImpl implements IChartService {**  **@Autowired**  **IEchartsDao iEchartsDao;**  **public EchartsData getChartsData() {**  **List<Integer> xAxiesList = iEchartsDao.getXAxiesList("");**  **List<Integer> pointsDataList = iEchartsDao.getPointsDataList("");**      **EchartsData data = new EchartsData();**  **ToolBox toolBox = EchartsOptionUtil.getToolBox();**  **Serie serie = EchartsOptionUtil.getSerie(pointsDataList);**  **ArrayList<Serie> series = new ArrayList<Serie>();**  **series.add(serie);**      **List<XAxi> xAxis = EchartsOptionUtil.getXAxis(xAxiesList);**  **List<YAxi> yAxis = EchartsOptionUtil.getYAxis();**    **HashMap<String, String> title = new HashMap<String, String>();**  **title.put("text", "pvs");**  **title.put("subtext", "超级pvs");**  **HashMap<String, String> tooltip = new HashMap<String, String>();**  **tooltip.put("trigger", "axis");**    **HashMap<String, String[]> legend = new HashMap<String, String[]>();**  **legend.put("data", new String[]{"pv统计"});**      **data.setTitle(title);**  **data.setTooltip(tooltip);**  **data.setLegend(legend);**  **data.setToolbox(toolBox);**  **data.setCalculable(true);**  **data.setxAxis(xAxis);**  **data.setyAxis(yAxis);**  **data.setSeries(series);**  **return data;**  **}**    **public List<HashMap<String, Integer>> getGaiKuangList(String date) throws ParseException{**    **HashMap<String, Integer> gaiKuangToday = iEchartsDao.getGaiKuang(date);**  **SimpleDateFormat sf = new SimpleDateFormat("MMdd");**  **Date parse = sf.parse(date);**  **Calendar calendar = Calendar.getInstance();**  **calendar.setTime(parse);**  **calendar.add(Calendar.DAY\_OF\_MONTH, -1);**  **Date before = calendar.getTime();**  **String beforeString = sf.format(before);**  **System.out.println(beforeString);**    **HashMap<String, Integer> gaiKuangBefore = iEchartsDao.getGaiKuang(beforeString);**    **ArrayList<HashMap<String, Integer>> gaiKuangList = new ArrayList<HashMap<String, Integer>>();**  **gaiKuangList.add(gaiKuangToday);**  **gaiKuangList.add(gaiKuangBefore);**    **return gaiKuangList;**    **}**    **public static void main(String[] args) {**  **ChartServiceImpl chartServiceImpl = new ChartServiceImpl();**  **EchartsData chartsData = chartServiceImpl.getChartsData();**  **Gson gson = new Gson();**  **String json = gson.toJson(chartsData);**  **System.out.println(json);**    **}**  **}** |

## Web程序的展现效果

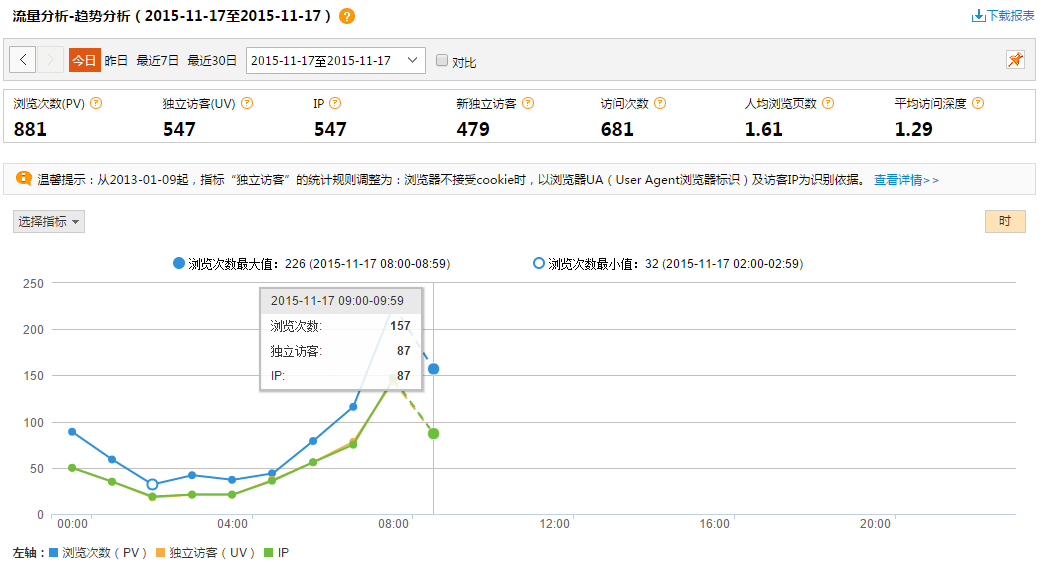
#### **网站概况**

****

****

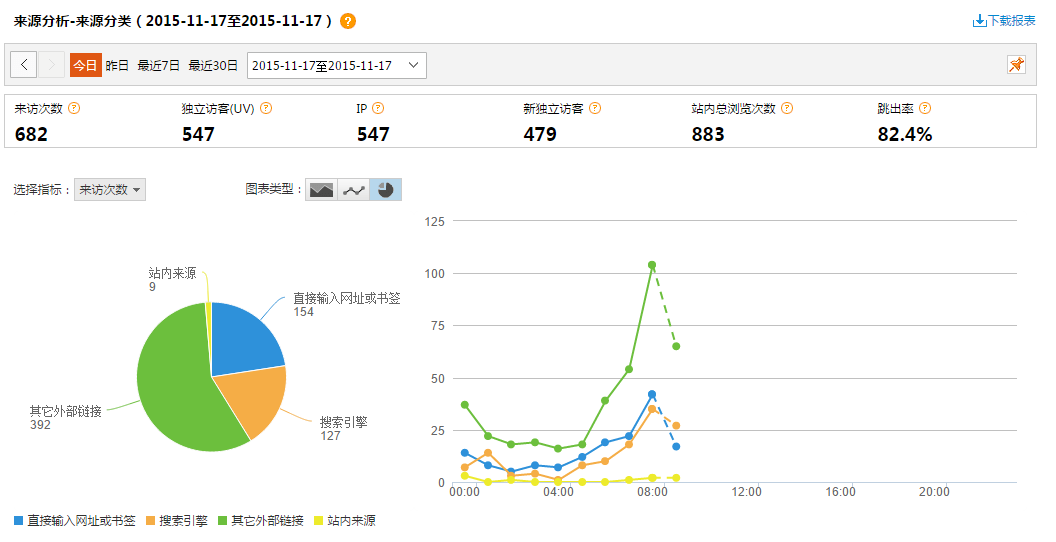
****

#### **流量分析**

****

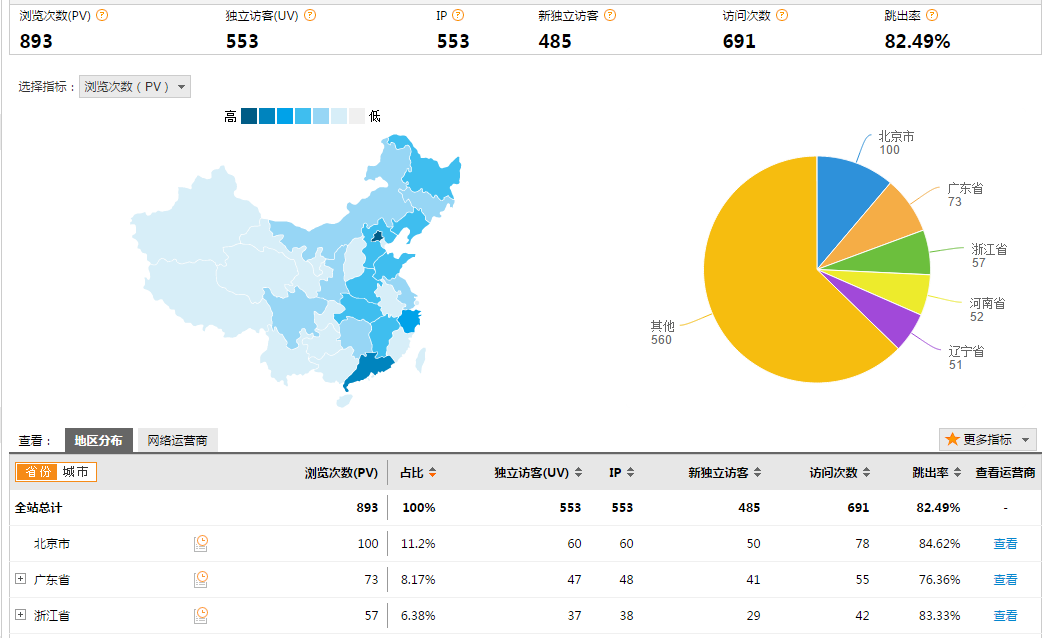
****

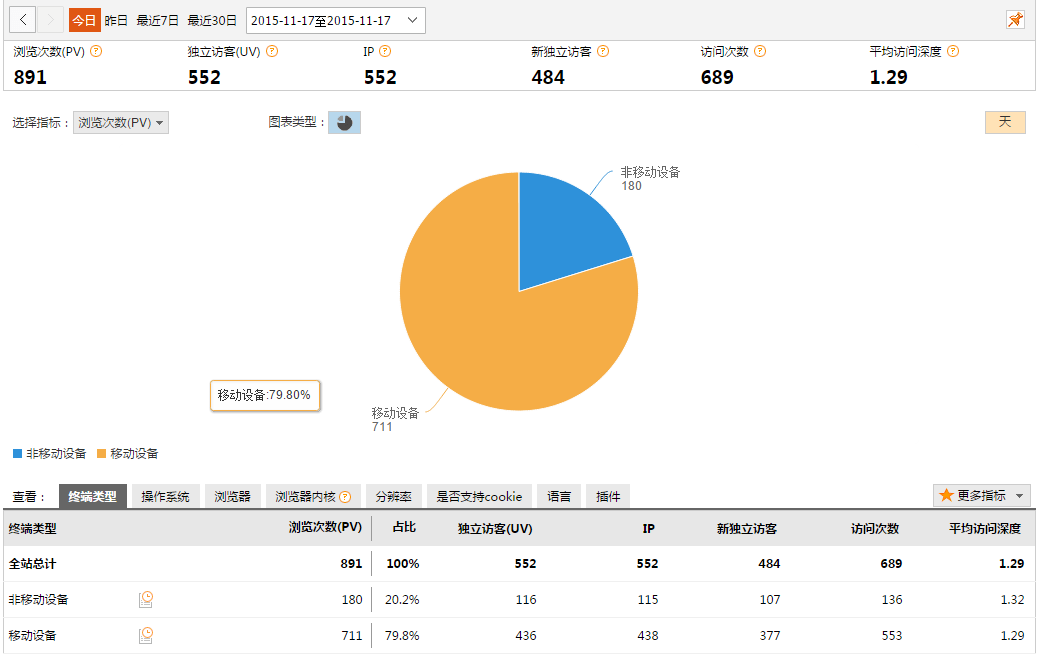
#### **来源分析**

****

****

#### **访客分析**

****

****