

网站日志分析（Data IDE 篇）实验手册

1 实验目标

1.1 实验背景介绍

本实验基于一份真实的日志数据，了解通过 Data IDE 操作 MaxCompute 来构建数据仓库。完成各种数据分析需求。

1.2 实验环境架构

本实验需要使用的环境有：大数据计算服务（ODPS）、大数据开发（Data IDE）、BI 报表。

2 实验内容

2.1 开通 Data IDE

在基础课程中已经有详细介绍，再此不再赘述。本实验中 Data IDE 项目名称对应课程编号_账号后三位数，如 aca21107_801_dev 为开发项目，对应的生产项目为 aca21107_801_prod。

2.2 实验：网站日志分析

2.2.1 实验：场景和数据说明

该示例基于一份真实的数据集，数据来源于酷壳（CoolShell.cn）网站上的

HTTP 访问日志数据。这份数据是 2014/2/12 一天的访问日志 (coolshell_20140212.log), 该日志数据是存在于 FTP 上。

数据格式如下：

```
$remote_addr - $remote_user [$time_local] "$request" $status  
$body_bytes_sent "$http_referer" "$http_user_agent" [unknown_content];
```

序号	字段名称	字段说明
1	\$remote_addr	发送请求的客户端 IP 地址。
2	\$remote_user	客户端登录名
3	\$time_local	服务器本地时间
4	\$request	请求，包括 HTTP 请求类型+请求 URL+HTTP 协议版本号
5	\$status	服务端返回状态码
6	\$body_bytes_sent	返回给客户端的字节数 (不含 header)
7	\$http_referer	该请求的来源 URL
8	\$http_user_agent	发送请求的客户端信息，如使用的浏览器等

一条真实的数据如下：

```
18.111.79.172 - - [12/Feb/2014:03:15:52 +0800] "GET /articles/4914.html HTTP/1.1"  
200 37666  
"http://coolshell.cn/articles/6043.html" "Mozilla/5.0 (Windows NT 6.2; WOW64)  
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.107 Safari/537.36" -
```

2.2.2 实验：需求分析

基于这份网络日志，完成如下分析需求：

1.统计网站的 PV (浏览次数)、UV (独立访客)，按用户的终端类型 (如

Android、iPad、iPhone、PC 等) 分别统计 , 并给出这一天的统计报表 ;

2. 网站的访问来源 , 了解网站的流量从哪里来。

【说明】网站统计指标 :

浏览次数 (PV) 和独立访客 (UV) 是衡量网站流量的两项最基本指标。用户每打开一个网站页面 , 记录一个 PV , 多次打开同一个页面 PV 累计多次。独立访客是指一天内 , 访问网站的不重复用户数 , 一天内同一个访客多次访问网站只计算一次。

Referer 表示该请求日志从哪里来 , 它可以分析完整的访问来源 , 还可以用于分析用户和偏好等 , 是网站广告投放评估的重要指标。

要实现这两个需求 , 整个流程如下 :

1) 日志数据解析、导入到 MaxCompute (原 ODPS 表) , 从数据仓库角度该表属于 ODS 层 , 因此导入的 ODPS 表名为 ods_log_tacker ;

2) 数据加工。从数据说明章节中我们看到日志数据中 \$request 字段包含 “HTTP 请求类型+请求 URL+HTTP 协议版本号” , 由于后续分析中经常会分别查询统计如 GET 的请求统计、URL 进行分析等 , 所以我们需要把原始表的 request 字段拆解 , 并把拆解后的数据写入表 dw_log_parser (数据仓库层表)。

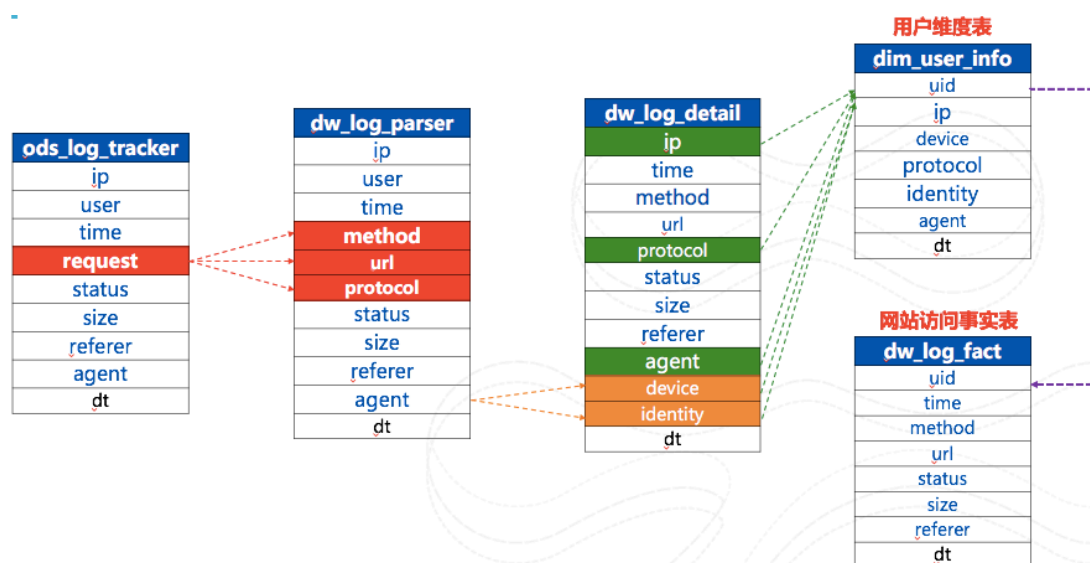
3) 数据分析。一般网站日志数据按用户身份可以划分为真实用户请求和程序发送请求 (订阅程序、爬虫等) 两大类 , 在统计流量 (PV、UV) 等指标时往往是基于真实用户请求的日志进行统计 , 此外 , 用户访问页面时 , 往往会有除页面本身请求外的其他如 js、图片发送的请求 , 这些都是做真实统计时需要过滤的请求。因此需要把这些分析的处理结果写到新表 dw_log_detail (数据仓库层表)。

4) 在数据仓库中 , 随着数据分析的深入 , 往往会构建维度表和事实表 , 本

实验中,我们可以构建用户维度表 dim_user_info 和网站访问事实表 dw_log_fact。

5) 根据数据仓库层中的维度表和事实表,生成基于终端设备信息的 PV 和 UV 统计表 adm_user_measures (应用数据集市层) 生成网站请求的来源 URL 统计表 adm_refer_info (应用数据集市层)。

以上 1-4 过程涉及到的各个表逻辑关系如下图:



2.2.3 实验：创建数据源

本实验案例,我们假设源数据是在 FTP 的文件 coolshell_20140212.log 中,即需要我们将 FTP 上的源数据表同步到 ODPS 的 ods_log_tracker,那么我们首先要在数据开发平台上给测试项目配置一个 FTP 数据源。具体配置步骤如下:

1) 项目管理员身份,浏览器中输入地址 <https://data.aliyun.com/console> 进入数加管理控制台,点击左侧导航“数据开发”,在项目列表中点击“测试项目”这个项目后面的数据开发:



2) 点击导航上的项目管理-数据源配置-新增数据源



弹出数据源配置表框，填写相关配置项：

【说明】FTP 配置信息：

Host：120.26.50.205

Port：21

用户名/密码：aca21107/DataIDE123

2.2.4 实验：创建数据表

创建 ods_log_tracker 表，用来导入日志数据。

1) 大数据开发平台页面导航上点击“数据开发”进入数据开发页面，点击工作区右上角新建选择下列列表中的“新建表”



进入建表页面：



2) 设计表，有两种方式，一种是使用图形界面，逐个添加字段，另一种是DDL 建表。在此我们使用 DDL 方式。点击建表页面右上角“DDL 建表”，在弹出的输入框里输入建表语句，添加提交：

DDL建表

```
1 CREATE TABLE IF NOT EXISTS ods_log_tracker(  
2     ip STRING COMMENT 'client ip address',  
3     user STRING,  
4     time DATETIME,  
5     request STRING COMMENT 'HTTP request type + requested path without args + HTTP  
protocol version',  
6     status BIGINT COMMENT 'HTTP reponse code from server',  
7     size BIGINT,  
8     referer STRING,  
9     agent STRING)  
10 COMMENT 'Log from coolshell.cn'  
11 PARTITIONED BY (dt STRING);
```

取消

提交

选择项目名 dev 项目（如果分开发/生产项目，建表需要建两次，每次分别选择开发/生产项目名），若找不到可以点击刷新按钮刷新项目列表。

设置表生命周期，为了便于案例数据保存更长久，该表生命周期设置为永久。

基础信息

字段和分区信息

新建成功!

1 基本信息设置

DDL建表

* 表名:

ods_log_tracker

别名:

请输入中文名

* 项目名:

odps.dataplus_private_test_4_dev

所属类目:

请选择类目

描述:

log from coolshell.cn

2 存储生命周期设置

* 生命周期:

永久

取消

下一步

点击下一步核对字段和分区，若无误直接点击提交。

4 是否设置分区： ☐ 否 ☒ 是

分区信息设置

字段英文名	字段类型	描述	操作
dt	STRING		编辑 删除

[+新增分区](#)

[取消](#) [上一步](#) [提交](#)

提交成功则表创建成功，用同样的方式继续创建其他表。

或者通过数据开发新建脚本执行建表方式建表，各个表建表语句请看附件。

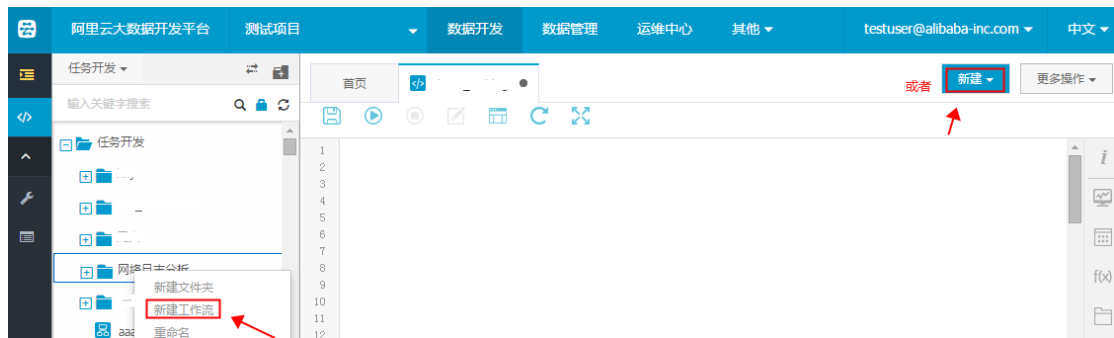
2.2.5 实验：创建工作流

进入数据开发页面，创建工作流 coolshell_log。

1) 创建工作流文件目录。文件目录树切换到“数据开发”新建目录——网络日志分析：



2) 目录文件夹上右键新建工作流或者点击工作区右上角新建按钮，下拉列表中选择新建工作流。



输入工作流名称和描述，并选择调度类型为“周期调度”。考虑到该工作流是需要后续每日自动调度生产每日报表，所以选择周期调度。

新建工作流

*工作流名称：

*描述：

*调度类型： ☐ 一次性调度 ☒ 周期调度

选择目录：

+

...

+

...

+

...

+

网络日志分析

+

...

点击“创建”按钮，成功创建工作流。

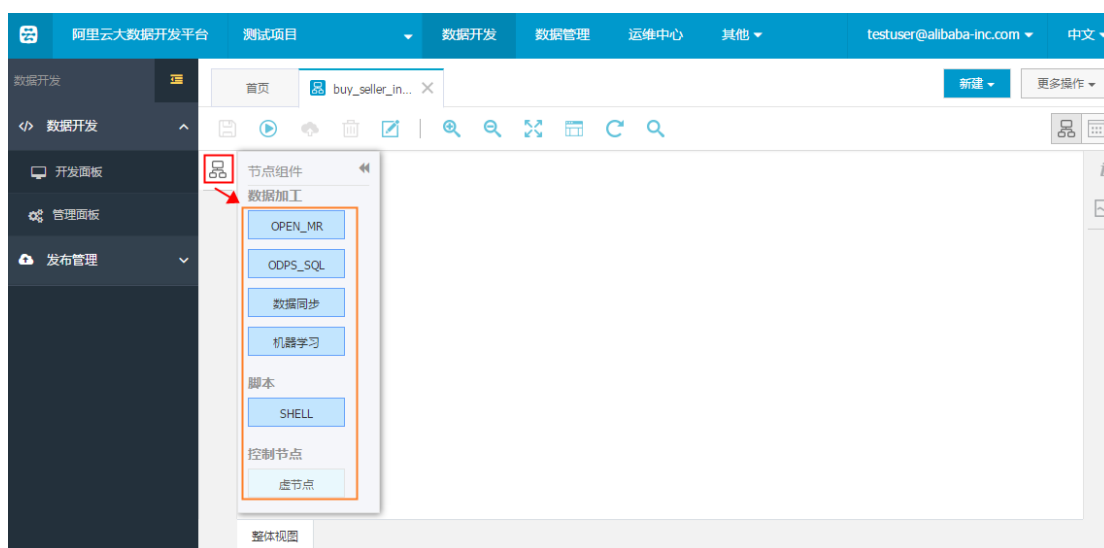
配置工作流调度时间属性：



2.2.6 实验：设计 workflow 节点

工作流创建好后，进入工作流设计面板添加节点进行节点设计。

1) 在工作流开发面板整体视图界面，点击左边节点组件图标，展开平台所支持的节点类型：



2) 鼠标双击节点组件或者拖拽节点组件到右边画布，依次添加以下节点：

- **SHELL**：利用 shell 调用某台机器上的 parse.py 文件来解析 coolshell_20140212.log 文件，生成 outputlog.log 文件。

- **同步任务**：将解析后的 output.log 日志文件同步支持 ODPS 表中 (ods_log_tracker)。

■ **数据加工 (ODPS SQL):** 节点名 dw_log_parser , 数据导入之后 , 对数据进行进一步的 ETL 过程 (request 字段拆解), 并将数据写入 dw_log_parser。

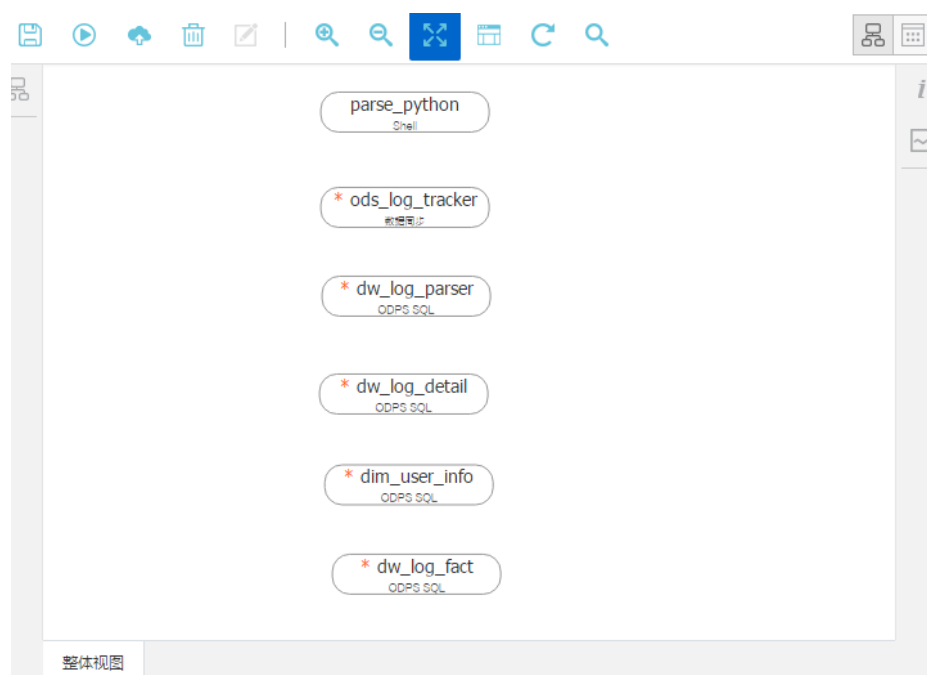
■ **数据分析 (ODPS SQL):** 节点名 dw_log_detail , 对 dw_log_parser 表进行进一步的数据分析、加工 , 得到 dw_log_detail。

■ **数据分析 (ODPS SQL):** 基于 dw_log_detail 表构建用户维度表 (dim_user_info) 和网站访问事实表 (dw_log_fact), 对应节点名称 dim_user_info 和 dw_log_fact。

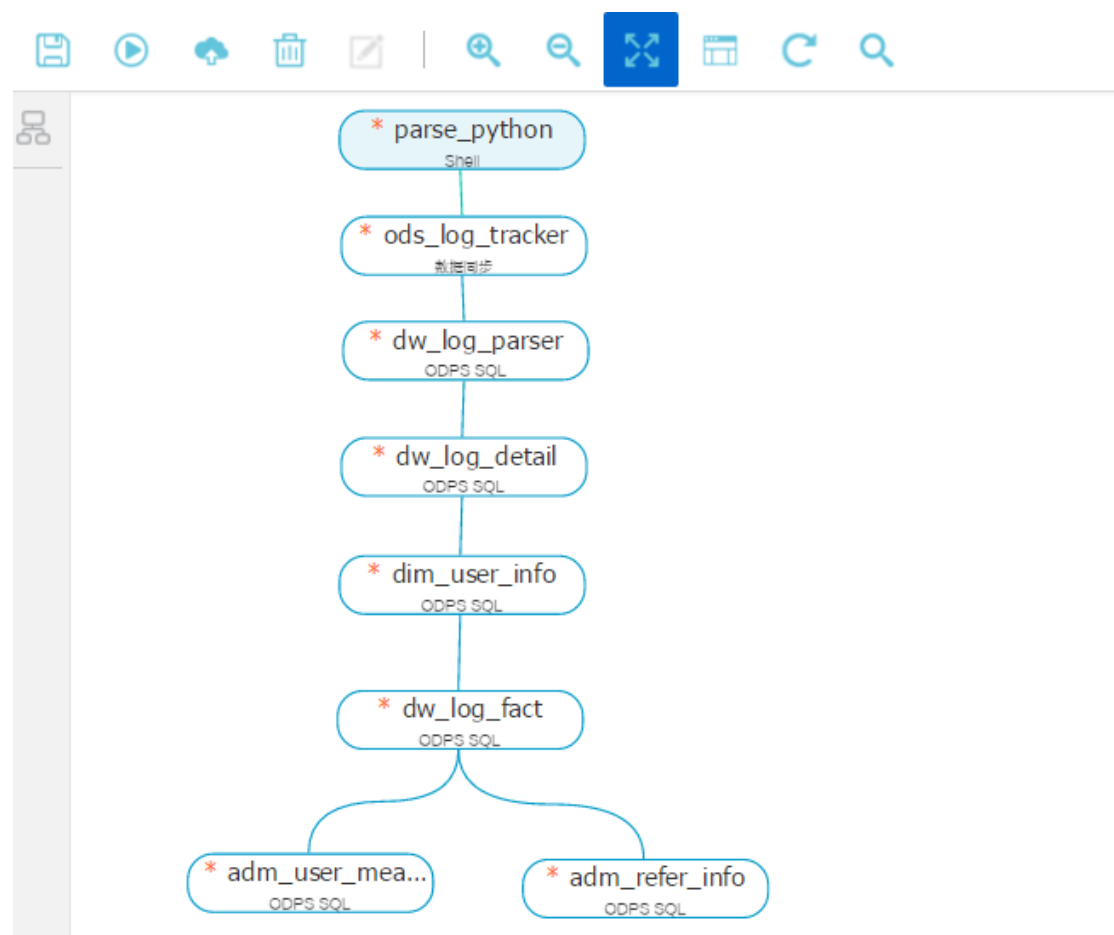
■ **数据应用 (ODPS SQL):** 基于前面的用户维度表和网站访问事实表 , 完成本实验 “需求分析” 中提出的业务需求 , 产出按用户终端类型统计网站的 PV/UV 表 (adm_user_measures) 和网站访问来源表 (adm_refer_info)。

数据应用面向业务需求 , 正常情况下 , 有可能这一层开发会是另一个团队同学 , 任何会在另外项目另外工作流里 , 本实验为了方便完整的走通 , 这一层的两个任务 dm_user_measures、 adm_refer_info 也放在这个工作流中。

这些节点在画布上散布如图 :



3) 需要根据节点内在的逻辑，通过连线体现出相互之间关系。鼠标移到节点上时该节点下沿中部有一个小半圆，然后把鼠标移到这个半圆上，鼠标即变为十字形，此时，按下左键可以划出连线，把鼠标拖至下一个节点，即形成了两节点之间的连线。连线体现了节点的依赖关系，箭头体现的是顺序。对节点进行连线，连线后点击保存按钮，保存我们的设计。连线后各节点的情况如下图：



2.2.7 实验：配置 SHELL 任务

Shell 任务主要是初步清洗原始 log 文件，需要清洗脚本直接到 log 文件存放的机器上运行，所以事先要把服务器自定义加为组织资源组，shell 任务将指定用该资源组运行。资源组添加具体可参考文档

:https://help.aliyun.com/document_detail/shujia/data-develop/quick-start/use-shell.html

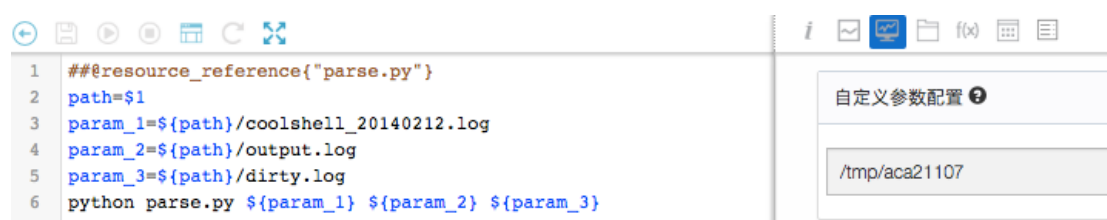
此处不再赘述具体资源组创建。

本实验中，shell 需要调用 python 脚本进行 log 文件处理，python 文件代码如附件，文件名为 parse.py，我们需要先将该文件添加为资源。

1) 工作区右上角新建>上传资源，把 parse.py 上传并注册为 odps 资源，资源名即为 parse.py。



2) 回到数据开发面板 coolshell_log 工作流工作区，双击 shell 节点进入代码编辑页面编辑代码，并配置 shell 节点调度主机（自定义资源组主机）：



保存 shell 节点。

2.2.8 实验：配置数据导入同步任务

对数据同步节点 ods_log_tracker 双击进入同步配置页面。



选择要抽取的列，并映射到目标表字段

来源	位置/值	类型	批量编辑	目标表字段	类型	批量编辑
字段	0	string		ip	STRING	
字段	1	string		user	STRING	
字段	2	string		time	DATETIME	
字段	3	string		request	STRING	
字段	4	string		status	BIGINT	
字段	5	string		size	BIGINT	
字段	6	string		referer	STRING	
字段	7	string		agent	STRING	

数据抽取和加载控制

抽取控制		加载控制	
* 列分隔符	<input type="text" value="\\u007c"/>	* 分区信息	<div><div>dt</div><div>=</div><div>\$(bdp.system.bizdate)</div></div>
编码格式	<input type="text" value="UTF-8"/>	清理规则	<div><input checked="" type="radio"/> 写入前清理已有数据 <input type="radio"/> 写入前保留已有数据</div>
null值	<input type="text" value="表示null值的字符串"/>		
压缩格式	<input type="text" value="None"/>		
跳过表头	<input type="text" value="否"/>		

其中列分隔符填入u007c，即经过 python 处理过后的数据文件是以为列分隔符来存储的。

由于本实验只有 20140212 一天的日志数据，也可把 `bdp.system.bizdate` 替换成常量 20140212 来更快速的进行实践。其他配置保留默认配置，保存节点。

2.2.9 实验：配置 ODPS SQL 任务

在开发面板整体视图分别对所有 ODPS SQL 节点双击进入节点代码编辑区，输入对应的 sql 语句（具体 sql 语句请看附件），并完成对应参数配置。

几个 sql 节点代码没有用到自定义变量，所以参数不需要配置，默认即可。

参数配置

系统参数配置 ?

\${bdp.system.bizda...}

yyyyMMdd

自定义参数配置 ?

代码和参数都配置完成，保存节点。

2.2.10 实验：添加/修改资源组

以组织管理员身份进入到【组织管理】模块，然后进入调度资源进行添加调度资源。具体见：

https://help.aliyun.com/document_detail/30272.html?spm=5176.doc30276.6.149.fgfCJ7

其中调度资源中涉及到的添加服务器，配置信息如下：

服务器名称：iZ23lki486xZ

机器 IP：10.117.222.163

修改资源组：

以组织管理员身份进入运维中心，点击任务管理列表，选择工作流中的 shell 节点，并点击修改资源组，如下图所示：



添加任务到资源组

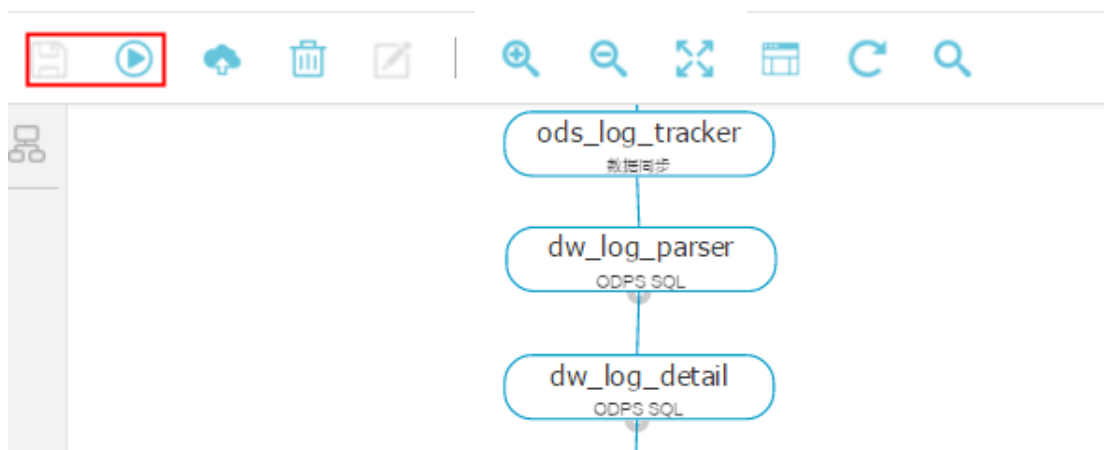
ftp_server_1

确定

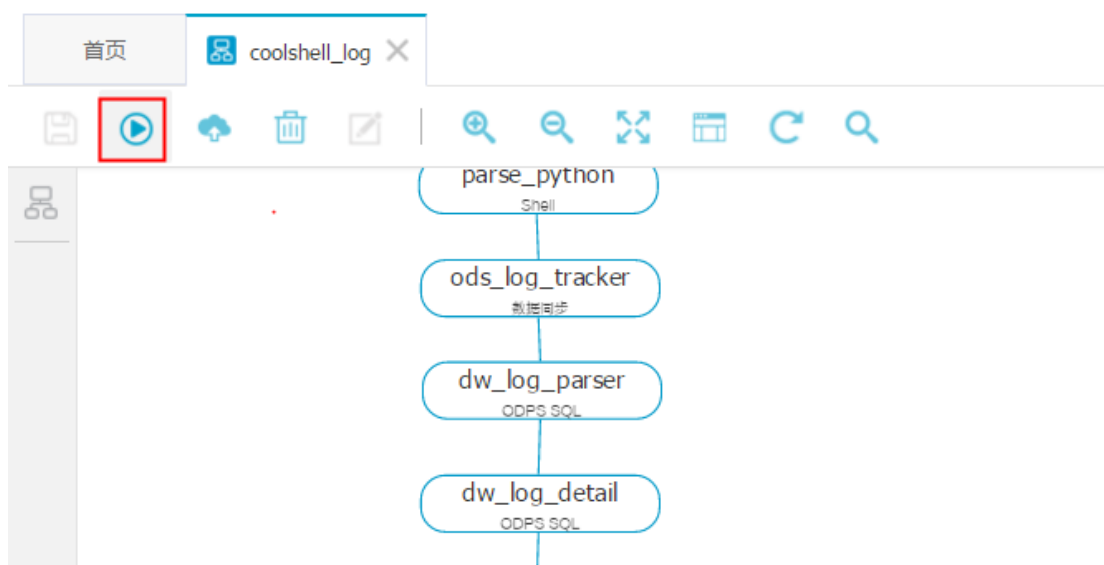
取消

2.2.11 实验：执行 workflow 节点

要执行整个工作流，需要先保存提交工作流。



工作流提交后，可以通过调度测试整个工作流所有节点运行情况。



本实验原始数据提供的是 20140212 一天的数据，所以在此我们业务日期选择 2014-02-12。

冒烟工作流

实例名称：

coolshell_log_2016_03_01

*业务日期：

2014-02-12

* 如果业务日期选择昨天之前，则立即执行任务。

* 如果业务日期选择昨天，则需要等到任务定时时间才能执行任务。

取消

创建

创建冒烟工作流后可调整到运维中心查看工作流测试具体情况：



双击工作流进入具体节点实例，看节点运行状态。

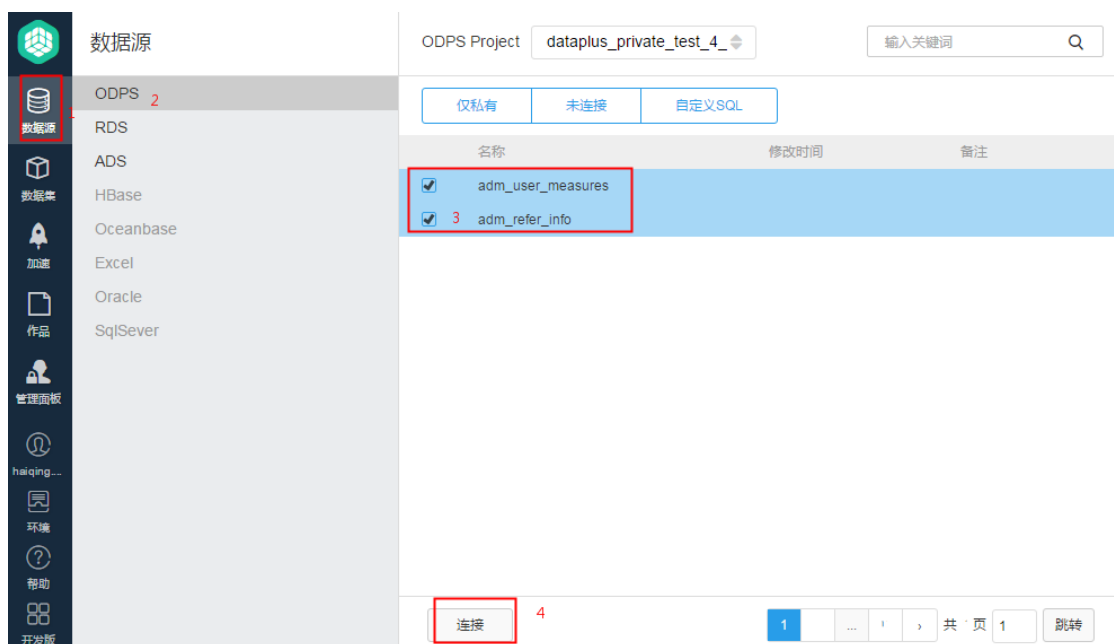
2.2.12 实验：BI 报表制作报表

应用数据层表产出后，我们可以直接通过“BI 报表”系统对数据进行报表统计，本实验我们将在 BI 报表上完成实验需求，产出简单的报表图。

回到数加管理控制台，左边导航点击 BI 报表，选择项目“测试项目”，进入 BI 报表页面。



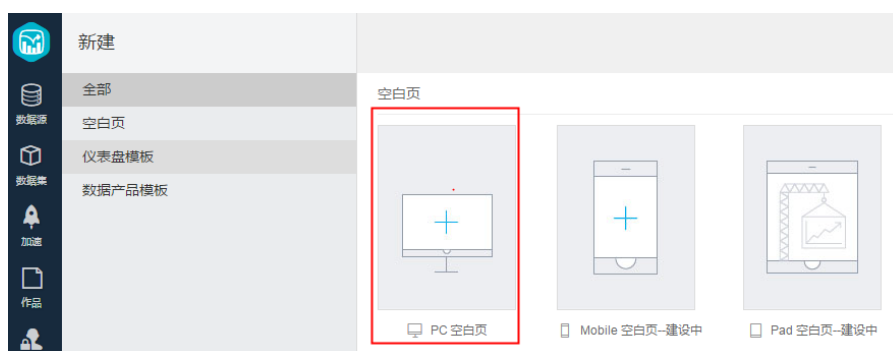
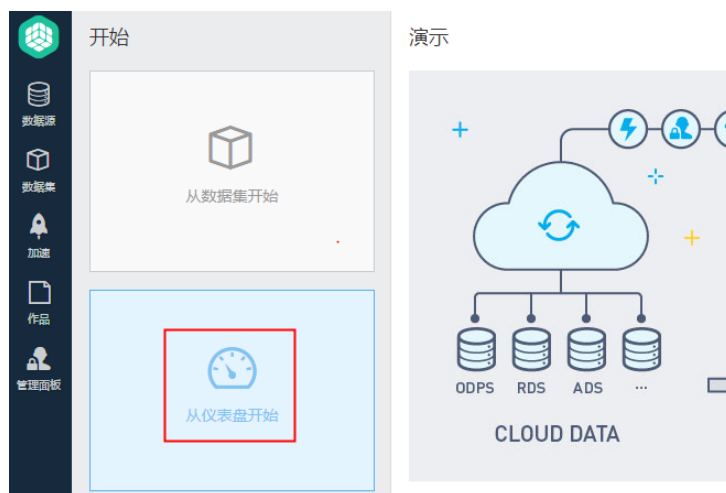
点击数据源，添加表 adm_user_measures 和表 adm_refer_info 为数据集。



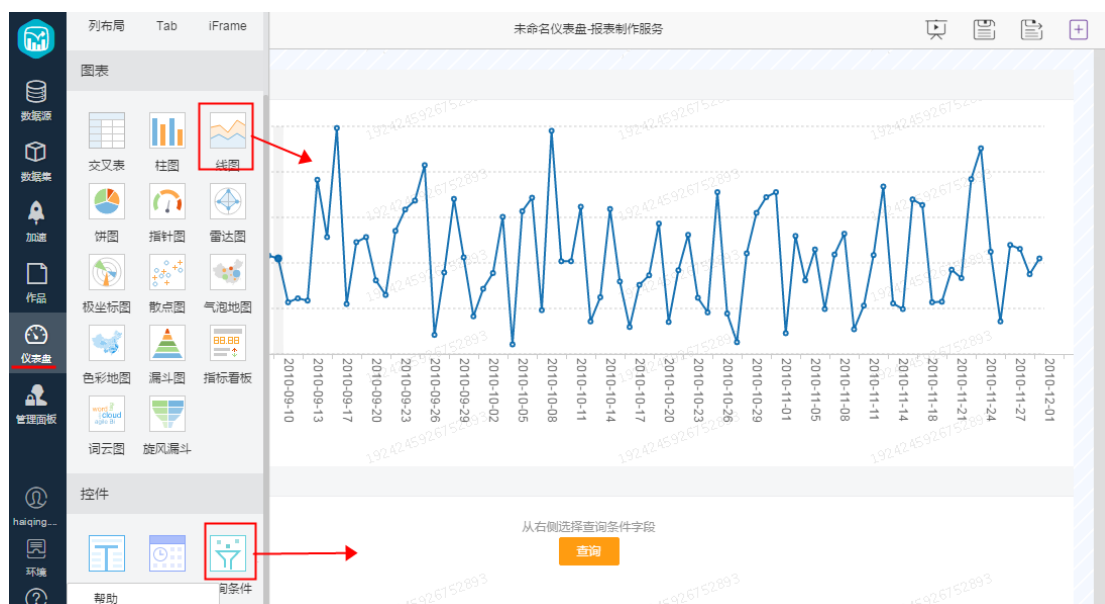
添加好后如下图：



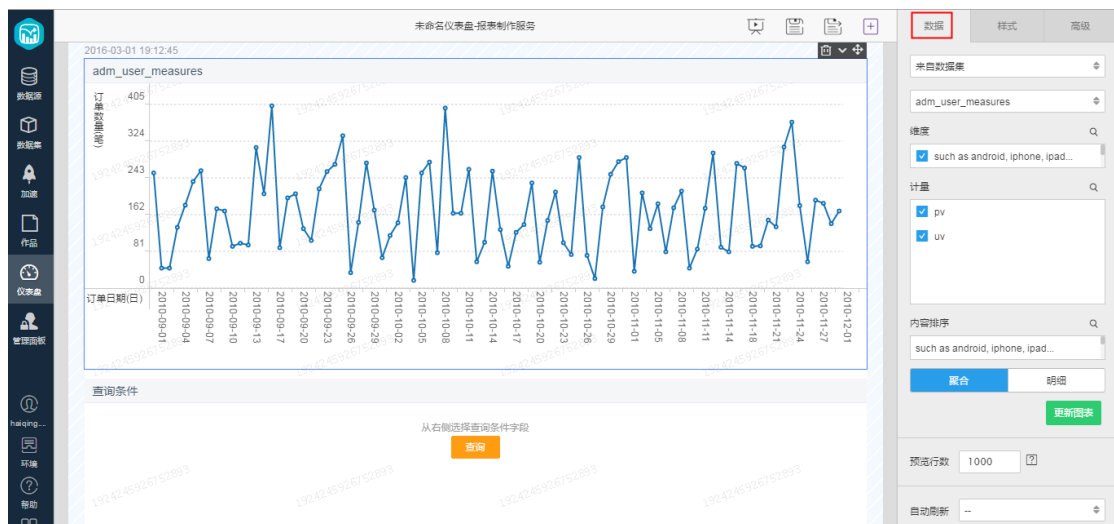
创建 coolshell 网站 PV/UV 统计图：回到数据分析服务首页，点击从仪表盘开始>PC 空白页，



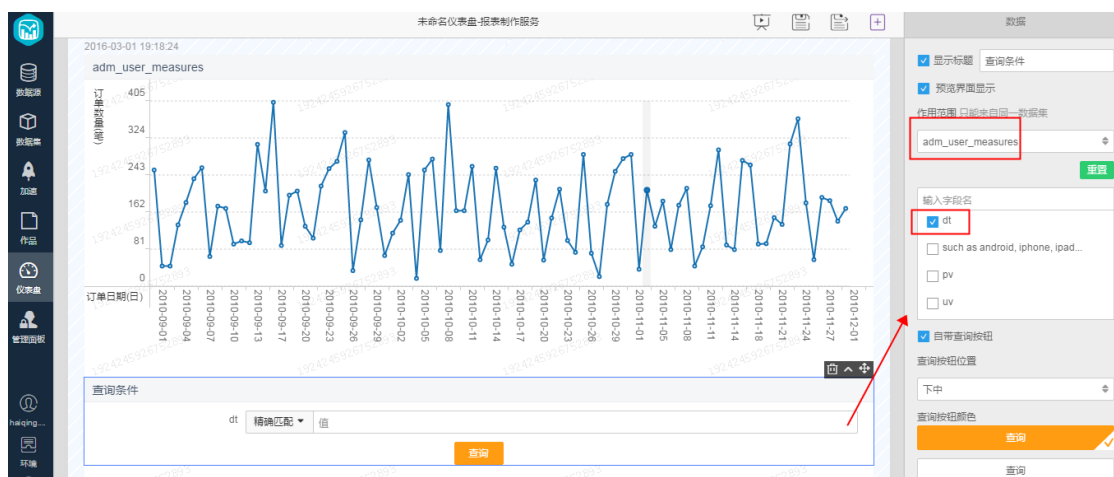
选择线图图表和查询条件控件，拖拽到工作区。



点击线图图表，右侧数据参数配置，数据源选来自数据集，选择表 adm_user_measures，选择好维度和计量如图，其他配置保持默认（本实验只做简单报表制作）。

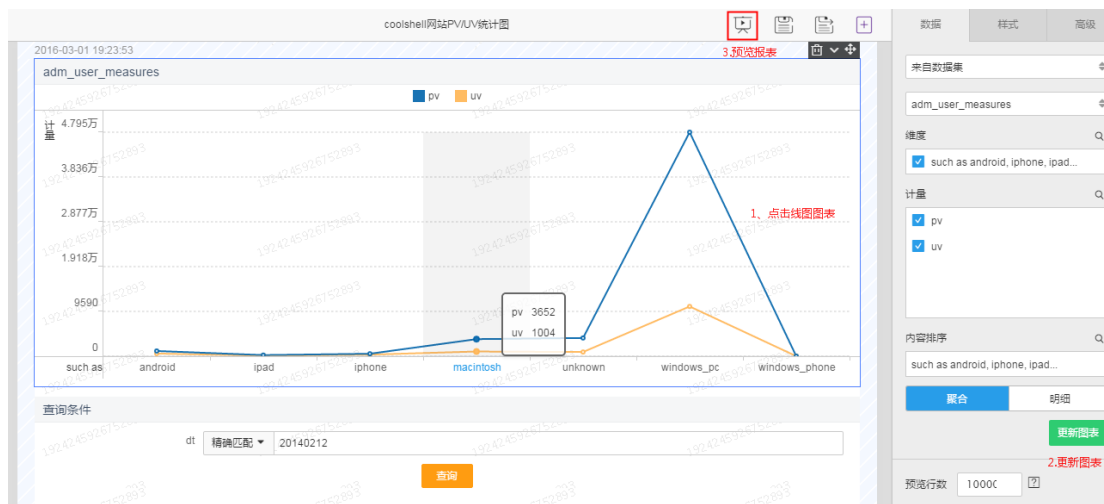


点击查询条件，右侧数据条件“请选择作用范围”选择 adm_user_measures 弹出的字段勾选 dt 分区字段，其他默认。

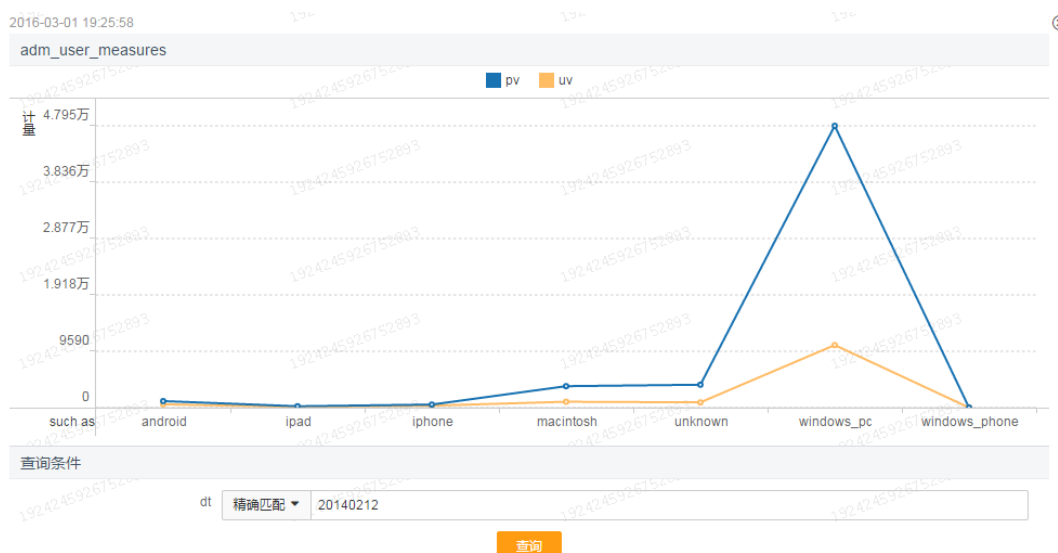


查询条件中，dt 需要精确匹配，本实验中我们的表分区为 20140212。

点击线图图表，右侧点击更新图表，预览报表。



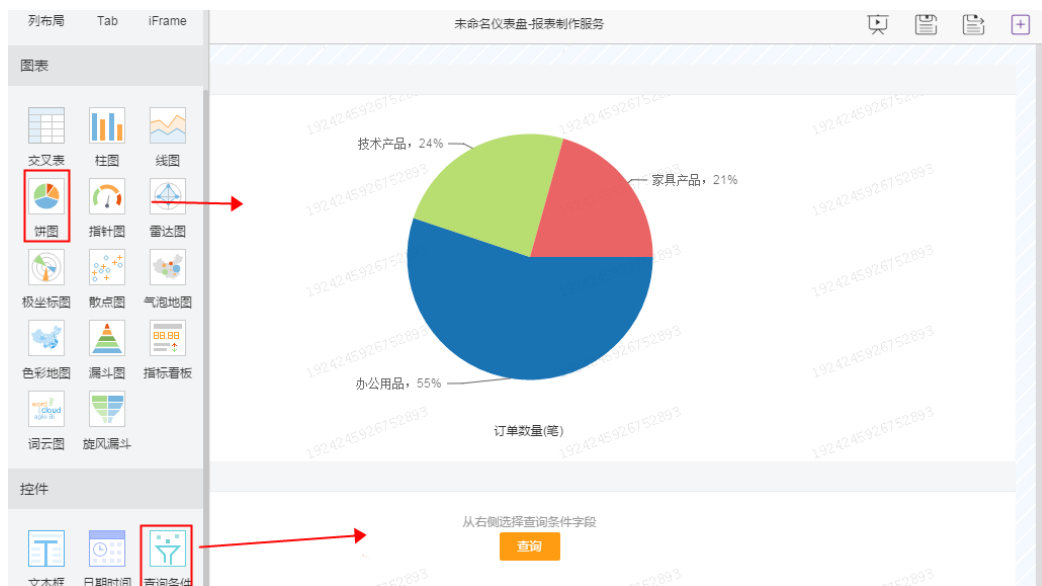
预览结果如下图



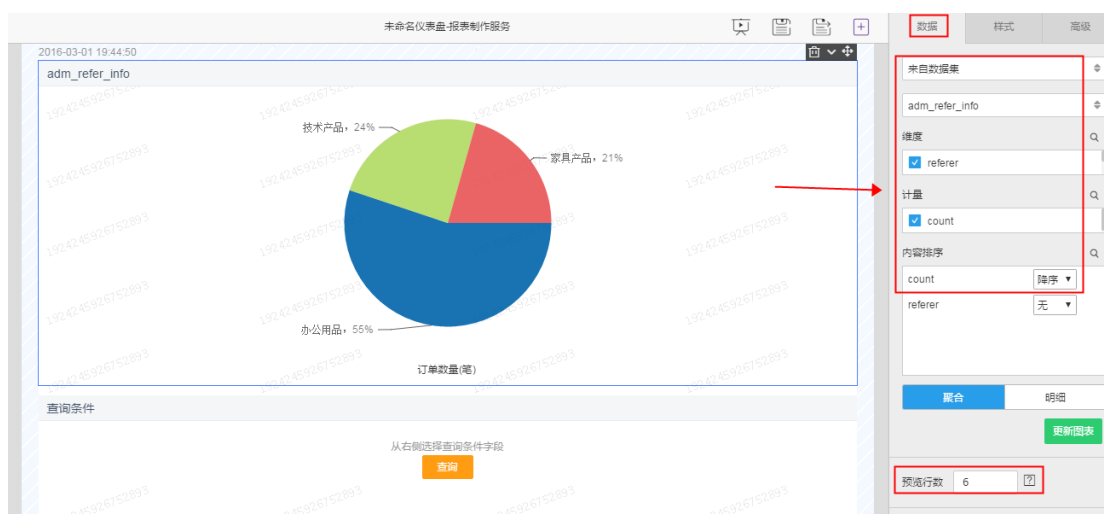
可以尝试 dt 输入其他分区值，如不存在的分区 20140101，看是否符合预期（没有数据），若符合点击预览页面右上角关闭按钮 ⊗ 回到编辑页，保存仪表盘



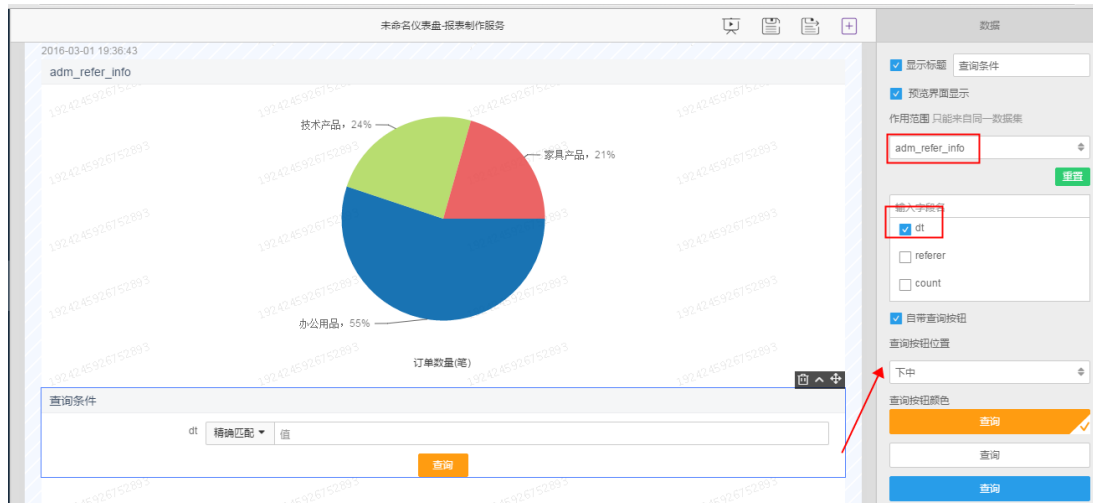
再点击新建按钮 + 新建空白仪表盘，选择饼图图表和查询条件控件拖拽到工作区：



单击饼图图表，右侧配置数据参数。数据源选择来自数据集，选择 adm_refer_info 数据集，维度选择 referer，计量选择 count，内容排序 count 选择降序，预览行数为 6（此处我们只看 top6），其他默认。

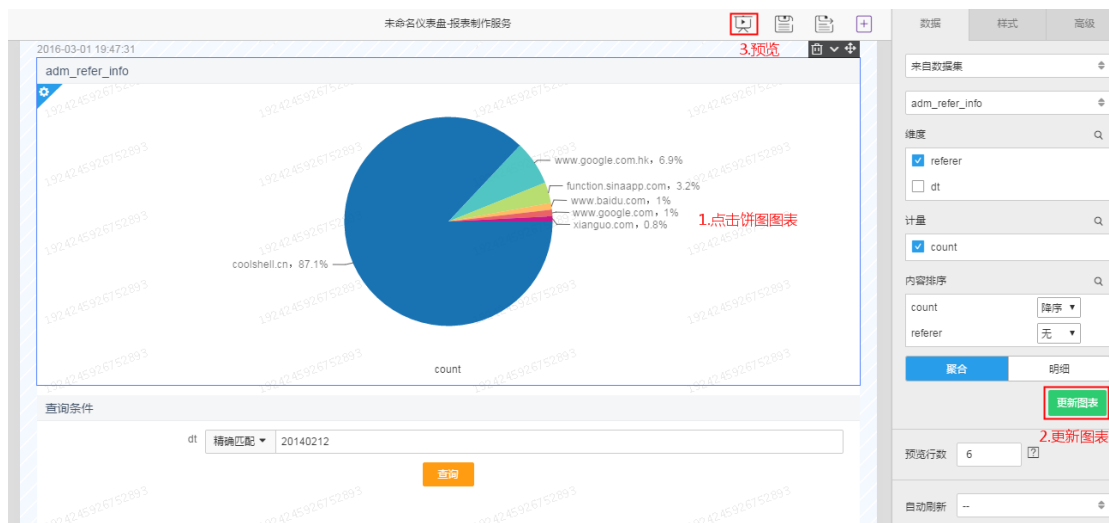


单击查询条件控件，右侧进行配置，作用范围选择 adm_refer_info，弹出的字段选择分区字段 dt，其他为默认。

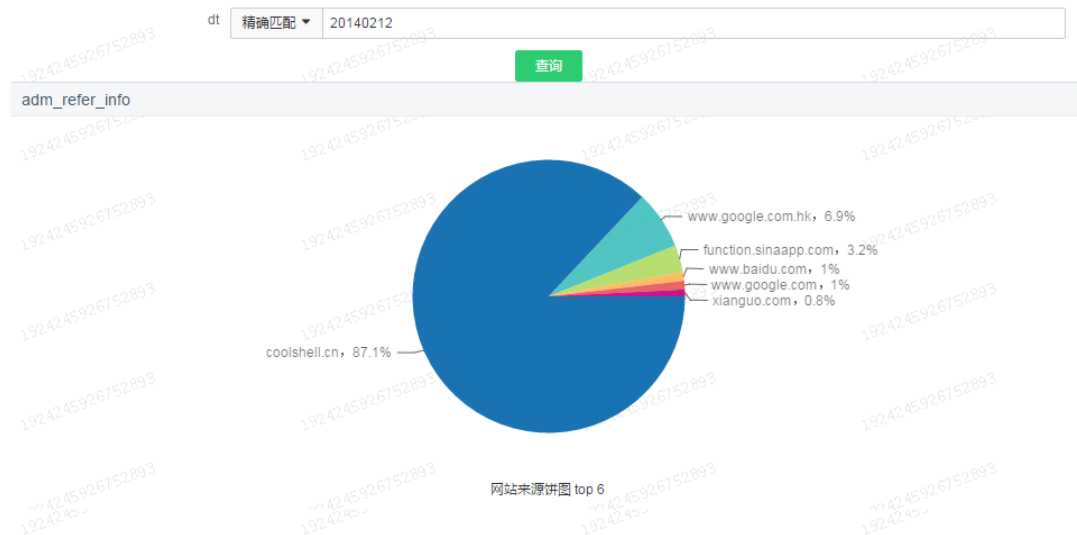


查询条件中，dt 需要精确匹配，本实验中我们的表分区为 20140212。


点击饼图图表，右侧点击更新图表，预览报表



预览结果如下图：



可以尝试 dt 输入不同分区看是否符合预期。

若符合点击预览页面右上角关闭按钮  回到编辑页，保存仪表盘



到此本实验两个需求报表就制作完成。

附件

ods_log_tracker_ddl

```
CREATE TABLE IF NOT EXISTS ods_log_tracker(  
  ip STRING COMMENT 'client ip address',  
  user STRING,  
  time DATETIME,  
  request STRING COMMENT 'HTTP request type + requested path without args  
+ HTTP protocol version',  
  status BIGINT COMMENT 'HTTP reponse code from server',  
  size BIGINT,  
  referer STRING,  
  agent STRING)  
COMMENT 'Log from coolshell.cn'  
PARTITIONED BY(dt STRING);
```


dw_log_parser_ddl

```
CREATE TABLE IF NOT EXISTS dw_log_parser(  
    ip STRING COMMENT 'client ip address',  
    user STRING,  
    time DATETIME,  
    method STRING COMMENT 'HTTP request type, such as GET POST...',  
    url STRING,  
    protocol STRING,  
    status BIGINT COMMENT 'HTTP reponse code from server',  
    size BIGINT,  
    referer STRING,  
    agent STRING)  
PARTITIONED BY(dt STRING);
```

dw_log_detail_ddl

```
CREATE TABLE IF NOT EXISTS dw_log_detail(  
    ip STRING COMMENT 'client ip address',  
    time DATETIME,  
    method STRING COMMENT 'HTTP request type, such as GET POST...',  
    url STRING,  
    protocol STRING,  
    status BIGINT COMMENT 'HTTP reponse code from server',  
    size BIGINT,  
    referer STRING COMMENT 'referer domain',  
    agent STRING,  
    device STRING COMMENT 'android|iphone|ipad...',  
    identity STRING COMMENT 'identify: user, crawler, feed')  
PARTITIONED BY(dt STRING);
```

dim_user_info_ddl

```
CREATE TABLE IF NOT EXISTS dim_user_info(  
    uid STRING COMMENT 'unique user id',  
    ip STRING COMMENT 'client ip address',  
    device STRING,  
    protocol STRING,  
    identity STRING COMMENT 'user, crawler, feed',  
    agent STRING)  
PARTITIONED BY(dt STRING);
```

dw_log_fact_ddl

```
CREATE TABLE IF NOT EXISTS dw_log_fact(  
    uid STRING COMMENT 'unique user id',  
    time DATETIME,  
    method STRING COMMENT 'HTTP request type, such as GET POST...',  
    url STRING,  
    status BIGINT COMMENT 'HTTP reponse code from server',  
    size BIGINT,  
    referer STRING)  
PARTITIONED BY(dt STRING);
```

adm_user_measures_ddl

```
CREATE TABLE IF NOT EXISTS adm_user_measures(  
    device STRING COMMENT 'such as android, iphone, ipad...',  
    pv BIGINT,  
    uv BIGINT)  
PARTITIONED BY(dt STRING);
```

adm_refer_info_ddl

```
CREATE TABLE adm_refer_info(  
    referer STRING,  
    count BIGINT)  
PARTITIONED BY(dt STRING);
```

parse.py 代码

```
-  
#!/usr/bin/python  
"""  
    A script to parse sample log.  
    [log_format]      $remote_addr - $remote_user [$time_local]  
                      "$request" $status $body_bytes_sent  
                      "$http_referer" "$http_user_agent" [unknown_content];  
    [output_format] ip,user,time,request,status,size,referrer,agent  
    """  
import sys
```

```

import re
import time

COL_DELIMITER = '|';

def convertTime(str):
    # convert: 12/Feb/2014:03:17:50 +0800
    #         to: 2014-02-12 03:17:50
    # [note] : timezone is not considered
    if str:
        return time.strftime('%Y-%m-%d %H:%M:%S',
                              time.strptime(str[:-6], '%d/%b/%Y:%H:%M:%S'))

def parseLog(inFile, outFile, dirtyFile):
    file    = open(inFile)
    output = open(outFile, "w")
    dirty   = open(dirtyFile, "w")
    items   = [
        r'(?P<ip>\S+)',           # ip
        r'\S+',                   # indent -, not used
        r'(?P<user>\S+)',         # user
        r'\[(?P<time>.\+)\]',     # time
        r'"(?P<request>.*)"',     # request
        r'(?P<status>[0-9]+)',    # status
        r'(?P<size>[0-9-]+)',     # size
        r'"(?P<referer>.*)"',     # referer
        r'"(?P<agent>.*)"',       # user agent
        r'(.*)',                 # unknown info
    ]
    pattern = re.compile(r'\s+'.join(items)+r'\s*\Z')
    for line in file:
        m = pattern.match(line)
        if not m:
            dirty.write(line)
        else:
            dict = m.groupdict()
            dict["time"] = convertTime(dict["time"])

            if dict["size"] == "-":
                dict["size"] = "0"

            for key in dict:
                if dict[key]=="-":
                    dict[key] = ""

```

```

        #ip,user,time,request,status,size,referrer
        output.write("%s\n" % (COL_DELIMITER.join(
            (dict["ip"],
              dict["user"],
              dict["time"],
              dict["request"],
              dict["status"],
              dict["size"],
              dict["referrer"],
              dict["agent"] )))))

    output.close()
    dirty.close()
    file.close()

if __name__ == "__main__":
    if len(sys.argv) != 4:
        print "Usage: %s <input_file> <output_file> <dirty_file>" % sys.argv[0]
        sys.exit(1)
    parseLog(sys.argv[1], sys.argv[2], sys.argv[3])i

```

dw_log_parser 节点代码

```

INSERT      OVERWRITE      TABLE      dw_log_parser      PARTITION
(dt=${bdp.system.bizdate})
SELECT ip
    , user
    , time
    , regexp_substr(request, '([^ ]+)') AS method
    , regexp_extract(request, '^[^ ]+ (.*) ([^ ]+)$') AS url
    , regexp_substr(request, '([^ ]+)$') AS protocol
    , status
    , size
    , referer
    , agent
FROM ods_log_tracker
WHERE dt = ${bdp.system.bizdate};

```

dw_log_detail 节点代码

```

INSERT      OVERWRITE      TABLE      dw_log_detail      PARTITION
(dt=${bdp.system.bizdate})
SELECT ip

```

```

, time
, method
, url
, protocol
, status
, size
, regexp_extract(referer, '^[^/]+://([^/]+){1}') AS referer
, agent
, CASE
    WHEN TOLOWER(agent) RLIKE 'android' THEN 'android'
    WHEN TOLOWER(agent) RLIKE 'iphone' THEN 'iphone'
    WHEN TOLOWER(agent) RLIKE 'ipad' THEN 'ipad'
    WHEN TOLOWER(agent) RLIKE 'macintosh' THEN 'macintosh'
    WHEN TOLOWER(agent) RLIKE 'windows phone' THEN 'windows_phone'
    WHEN TOLOWER(agent) RLIKE 'windows' THEN 'windows_pc'
    ELSE 'unknown'
END AS device
, CASE
    WHEN TOLOWER(agent) RLIKE '(bot|spider|crawler|slurp)' THEN
'crawler'
    WHEN TOLOWER(agent) RLIKE 'feed'
    OR url RLIKE 'feed' THEN 'feed'
    WHEN TOLOWER(agent) NOT RLIKE '(bot|spider|crawler|feed|slurp)'
    AND agent RLIKE '^[Mozilla|Opera]'
    AND url NOT RLIKE 'feed' THEN 'user'
    ELSE 'unknown'
END AS identity
FROM dw_log_parser
WHERE url NOT RLIKE '^[/]+wp-'
AND dt=${bdp.system.bizdate};

```

dim_user_info 节点代码

```

INSERT OVERWRITE TABLE dim_user_info PARTITION
(dt=${bdp.system.bizdate})
SELECT md5(concat(t1.ip, t1.device, t1.protocol, t1.identity, t1.agent))
, t1.ip
, t1.device
, t1.protocol
, t1.identity
, t1.agent
FROM (
    SELECT ip
    , protocol

```

```

        , agent
        , device
        , identity
FROM dw_log_detail
WHERE dt = ${bdp.system.bizdate}
GROUP BY ip,
        protocol,
        agent,
        device,
        identity
) t1;

```

dw_log_fact 节点代码

```

INSERT      OVERWRITE      TABLE      dw_log_fact      PARTITION
(dt=${bdp.system.bizdate})
SELECT u.uid
        , d.time
        , d.method
        , d.url
        , d.status
        , d.size
        , d.referer
FROM dw_log_detail d
JOIN dim_user_info u
ON (d.ip = u.ip
    AND d.protocol = u.protocol
    AND d.agent = u.agent) and d.dt = ${bdp.system.bizdate}    AND      u.dt
=${bdp.system.bizdate};

```

adm_user_measures 节点代码

```

INSERT      OVERWRITE      TABLE      adm_user_measures      PARTITION
(dt='${bdp.system.bizdate}')
SELECT u.device
        , COUNT(*) AS pv
        , COUNT(DISTINCT u.uid) AS uv
FROM dw_log_fact f
JOIN dim_user_info u
ON f.uid = u.uid
    AND u.identity = 'user'
    AND f.dt = '${bdp.system.bizdate}'

```

```
    AND u.dt = '${bdp.system.bizdate}'  
GROUP BY u.device;
```

adm_refer_info 节点代码

```
INSERT      OVERWRITE      TABLE      adm_refer_info      PARTITION  
(dt='${bdp.system.bizdate}')
```

```
SELECT referer  
      , COUNT(*) AS cnt  
FROM dw_log_fact  
WHERE LENGTH(referer) > 1  
      AND dt = '${bdp.system.bizdate}'  
GROUP BY referer;
```