电商数仓（用户行为数据仓库）

版本：V1.4

# 第1章 数仓分层概念

## 1.1 为什么要分层



## 1.2 数仓分层





## 1.3 关于区分数据集市与数据仓库

数据集市（Date Market），现在市面上的公司和书籍都对数据集市有不同的概念。

狭义上来讲数据集市，可以理解为数据仓库中为用户提供数据支撑的应用层，比如咱们前文说的ADS层。

广义上，数据集市，所有以主题划分的数据仓库中可供查阅的都可以成为数据集市，包括DWD，DWS，ADS层，甚至包括从Hadoop中同步到RDS的数据都可以成为数据集市。

比如订单主题，我可以提供使用者，从明细，聚合统计，比率分析等全部数据，提供给某个部门查询。那么除了订单还有用户、商品、供应商等等主题分别可以供不同的人员部门使用，这都可以称之为数据集市。

# 第2章 数仓搭建环境准备



集群规划

|  |  |  |  |
| --- | --- | --- | --- |
|  | 服务器hadoop102 | 服务器hadoop103 | 服务器hadoop104 |
| Hive | Hive |  |  |
| Mysql | Mysql |  |  |

## 2.1 Hive&MySQL安装

### 2.1.1 Hive&MySQL安装

详见： Hive



### 2.1.2 修改 hive-site.xml

1）关闭元数据检查

[byf@hadoop102 conf]$ pwd

/opt/module/hive/conf

[byf@hadoop102 conf]$ vim hive-site.xml

增加如下配置：

<property>

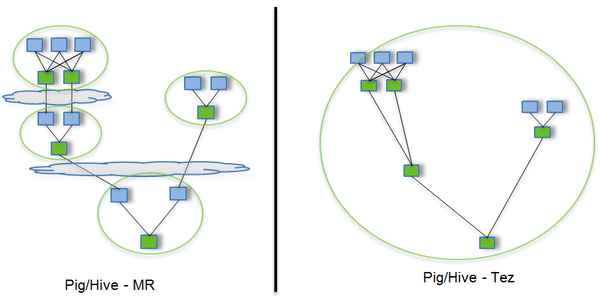
<name>hive.metastore.schema.verification</name>

<value>false</value>

</property>

## 2.2 Hive运行引擎Tez

Tez是一个Hive的运行引擎，性能优于MR。为什么优于MR呢？看下图。



用Hive直接编写MR程序，假设有四个有依赖关系的MR作业，上图中，绿色是Reduce Task，云状表示写屏蔽，需要将中间结果持久化写到HDFS。

Tez可以将多个有依赖的作业转换为一个作业，这样只需写一次HDFS，且中间节点较少，从而大大提升作业的计算性能。

### 2.2.1 安装包准备

1）下载tez的依赖包：<http://tez.apache.org>

2）拷贝apache-tez-0.9.1-bin.tar.gz到hadoop102的/opt/module目录

[byf@hadoop102 module]$ ls

apache-tez-0.9.1-bin.tar.gz

3）解压缩apache-tez-0.9.1-bin.tar.gz

[byf@hadoop102 module]$ tar -zxvf apache-tez-0.9.1-bin.tar.gz

4）修改名称

[byf@hadoop102 module]$ mv apache-tez-0.9.1-bin/ tez-0.9.1

### 2.2.2 在Hive中配置Tez

1）进入到Hive的配置目录：/opt/module/hive/conf

[byf@hadoop102 conf]$ pwd

/opt/module/hive/conf

2）在hive-env.sh文件中添加tez环境变量配置和依赖包环境变量配置

[byf@hadoop102 conf]$ vim hive-env.sh

添加如下配置

# Set HADOOP\_HOME to point to a specific hadoop install directory

export HADOOP\_HOME=/opt/module/hadoop-2.7.2

# Hive Configuration Directory can be controlled by:

export HIVE\_CONF\_DIR=/opt/module/hive/conf

# Folder containing extra libraries required for hive compilation/execution can be controlled by:

export TEZ\_HOME=/opt/module/**tez-0.9.1** #是你的tez的解压目录

export TEZ\_JARS=""

for jar in `ls $TEZ\_HOME |grep jar`; do

export TEZ\_JARS=$TEZ\_JARS:$TEZ\_HOME/$jar

done

for jar in `ls $TEZ\_HOME/lib`; do

export TEZ\_JARS=$TEZ\_JARS:$TEZ\_HOME/lib/$jar

done

export HIVE\_AUX\_JARS\_PATH=/opt/module/hadoop-2.7.2/share/hadoop/common/**hadoop-lzo-0.4.20.jar**$TEZ\_JARS

3）在hive-site.xml文件中添加如下配置，更改hive计算引擎

<property>

<name>hive.execution.engine</name>

<value>tez</value>

</property>

### 2.2.3 配置Tez

1）在Hive 的/opt/module/hive/conf下面创建一个tez-site.xml文件

[byf@hadoop102 conf]$ pwd

/opt/module/hive/conf

[byf@hadoop102 conf]$ vim tez-site.xml

添加如下内容

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>tez.lib.uris</name> <value>${fs.defaultFS}/tez/tez-0.9.1,${fs.defaultFS}/tez/tez-0.9.1/lib</value>

</property>

<property>

<name>tez.lib.uris.classpath</name> <value>${fs.defaultFS}/tez/tez-0.9.1,${fs.defaultFS}/tez/tez-0.9.1/lib</value>

</property>

<property>

<name>tez.use.cluster.hadoop-libs</name>

<value>true</value>

</property>

<property>

<name>tez.history.logging.service.class</name> <value>org.apache.tez.dag.history.logging.ats.ATSHistoryLoggingService</value>

</property>

</configuration>

### 2.2.4 上传Tez到集群

1）将/opt/module/tez-0.9.1上传到HDFS的/tez路径

[byf@hadoop102 conf]$ hadoop fs -mkdir /tez

[byf@hadoop102 conf]$ hadoop fs -put /opt/module/tez-0.9.1/ /tez

[byf@hadoop102 conf]$ hadoop fs -ls /tez

/tez/tez-0.9.1

### 2.2.5 测试

1）启动Hive

[byf@hadoop102 hive]$ bin/hive

2）创建LZO表

hive (default)> create table student(

id int,

name string);

3）向表中插入数据

hive (default)> insert into student values(1,"zhangsan");

4）如果没有报错就表示成功了

hive (default)> select \* from student;

1 zhangsan

### 2.2.6 小结

1）运行Tez时检查到用过多内存而被NodeManager杀死进程问题：

Caused by: org.apache.tez.dag.api.SessionNotRunning: TezSession has already shutdown. Application application\_1546781144082\_0005 failed 2 times due to AM Container for appattempt\_1546781144082\_0005\_000002 exited with exitCode: -103

For more detailed output, check application tracking page:http://hadoop103:8088/cluster/app/application\_1546781144082\_0005Then, click on links to logs of each attempt.

Diagnostics: Container [pid=11116,containerID=container\_1546781144082\_0005\_02\_000001] is running beyond virtual memory limits. Current usage: 216.3 MB of 1 GB physical memory used; 2.6 GB of 2.1 GB virtual memory used. Killing container.

这种问题是从机上运行的Container试图使用过多的内存，而被NodeManager kill掉了。

[摘录] The NodeManager is killing your container. It sounds like you are trying to use hadoop streaming which is running as a child process of the map-reduce task. The NodeManager monitors the entire process tree of the task and if it eats up more memory than the maximum set in mapreduce.map.memory.mb or mapreduce.reduce.memory.mb respectively, we would expect the Nodemanager to kill the task, otherwise your task is stealing memory belonging to other containers, which you don't want.

**解决方法：**

方案一：或者是关掉虚拟内存检查。我们选这个，修改yarn-site.xml

<property>

<name>yarn.nodemanager.vmem-check-enabled</name>

<value>false</value>

</property>

方案二：mapred-site.xml中设置Map和Reduce任务的内存配置如下：(value中实际配置的内存需要根据自己机器内存大小及应用情况进行修改)

<property>

　　<name>mapreduce.map.memory.mb</name>

　　<value>1536</value>

</property>

<property>

　　<name>mapreduce.map.java.opts</name>

　　<value>-Xmx1024M</value>

</property>

<property>

　　<name>mapreduce.reduce.memory.mb</name>

　　<value>3072</value>

</property>

<property>

　　<name>mapreduce.reduce.java.opts</name>

　　<value>-Xmx2560M</value>

</property>

# 第3章 数仓搭建之ODS & DWD



## 3.1 创建数据库

1）创建gmall数据库

hive (default)> create database gmall;

说明：如果数据库存在且有数据，需要强制删除时执行：drop database gmall cascade;

2）使用gmall数据库

hive (default)> use gmall;

## 3.2 ODS层

原始数据层，存放原始数据，直接加载原始日志、数据，数据保持原貌不做处理。

### 3.2.1 创建启动日志表ods\_start\_log



1）创建输入数据是lzo输出是text，支持json解析的分区表

hive (gmall)>

drop table if exists ods\_start\_log;

CREATE EXTERNAL TABLE ods\_start\_log (`line` string)

PARTITIONED BY (`dt` string)

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION '/warehouse/gmall/ods/ods\_start\_log';

说明Hive的LZO压缩：https://cwiki.apache.org/confluence/display/Hive/LanguageManual+LZO

2）加载数据

hive (gmall)>

load data inpath '/origin\_data/gmall/log/topic\_start/2019-02-10' into table gmall.ods\_start\_log partition(dt='2019-02-10');

注意：时间格式都配置成YYYY-MM-DD格式，这是Hive默认支持的时间格式

3）查看是否加载成功

hive (gmall)> select \* from ods\_start\_log limit 2;

### 3.2.2 创建事件日志表ods\_event\_log



1）创建输入数据是lzo输出是text，支持json解析的分区表

hive (gmall)>

drop table if exists ods\_event\_log;

CREATE EXTERNAL TABLE `ods\_event\_log`(`line` string)

PARTITIONED BY (`dt` string)

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION '/warehouse/gmall/ods/ods\_event\_log';

2）加载数据

hive (gmall)>

load data inpath '/origin\_data/gmall/log/topic\_event/2019-02-10' into table gmall.ods\_event\_log partition(dt='2019-02-10');

注意：时间格式都配置成YYYY-MM-DD格式，这是Hive默认支持的时间格式

3）查看是否加载成功

hive (gmall)> select \* from ods\_event\_log limit 2;

### 3.2.3 ODS层加载数据脚本

1）在hadoop102的/home/byf/bin目录下创建脚本

[byf@hadoop102 bin]$ vim ods\_log.sh

在脚本中编写如下内容

#!/bin/bash

# 定义变量方便修改

APP=gmall

hive=/opt/module/hive/bin/hive

# 如果是输入的日期按照取输入日期；如果没输入日期取当前时间的前一天

if [ -n "$1" ] ;then

do\_date=$1

else

do\_date=`date -d "-1 day" +%F`

fi

echo "===日志日期为 $do\_date==="

sql="

load data inpath '/origin\_data/gmall/log/topic\_start/$do\_date' into table "$APP".ods\_start\_log partition(dt='$do\_date');

load data inpath '/origin\_data/gmall/log/topic\_event/$do\_date' into table "$APP".ods\_event\_log partition(dt='$do\_date');

"

$hive -e "$sql"

说明1：

[ -n 变量值 ] 判断变量的值，是否为空

-- 变量的值，非空，返回true

-- 变量的值，为空，返回false

说明2：

查看date命令的使用，[byf@hadoop102 ~]$ date --help

2）增加脚本执行权限

[byf@hadoop102 bin]$ chmod 777 ods\_log.sh

3）脚本使用

[byf@hadoop102 module]$ ods\_log.sh 2019-02-11

4）查看导入数据

hive (gmall)>

select \* from ods\_start\_log where dt='2019-02-11' limit 2;

select \* from ods\_event\_log where dt='2019-02-11' limit 2;

5）脚本执行时间

企业开发中一般在每日凌晨30分~1点

## 3.3 DWD层数据解析

对ODS层数据进行清洗（去除空值，脏数据，超过极限范围的数据，行式存储改为列存储，改压缩格式）。

### 3.3.1 创建基础明细表

明细表用于存储ODS层原始表转换过来的明细数据。



1）创建启动日志基础明细表

hive (gmall)>

drop table if exists dwd\_base\_start\_log;

CREATE EXTERNAL TABLE dwd\_base\_start\_log(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`event\_name` string,

`event\_json` string,

`server\_time` string)

PARTITIONED BY (`dt` string)

stored as parquet

location '/warehouse/gmall/dwd/dwd\_base\_start\_log/';

其中event\_name和event\_json用来对应事件名和整个事件。这个地方将原始日志1对多的形式拆分出来了。操作的时候我们需要将原始日志展平，需要用到UDF和UDTF。

2）创建事件日志基础明细表

hive (gmall)>

drop table if exists dwd\_base\_event\_log;

CREATE EXTERNAL TABLE dwd\_base\_event\_log(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`event\_name` string,

`event\_json` string,

`server\_time` string)

PARTITIONED BY (`dt` string)

stored as parquet

location '/warehouse/gmall/dwd/dwd\_base\_event\_log/';

### 3.3.2 自定义UDF函数（解析公共字段）

1）创建一个maven工程：hivefunction

2）创建包名：com.byf.udf

3）在pom.xml文件中添加如下内容

<**properties**>  
 <**project.build.sourceEncoding**>UTF8</**project.build.sourceEncoding**>  
 <**hive.version**>1.2.1</**hive.version**>  
</**properties**>  
  
<**dependencies**>  
 *<!--添加hive依赖-->* <**dependency**>  
 <**groupId**>org.apache.hive</**groupId**>  
 <**artifactId**>hive-exec</**artifactId**>  
 <**version**>${hive.version}</**version**>  
 </**dependency**>  
</**dependencies**>  
  
<**build**>  
 <**plugins**>  
 <**plugin**>  
 <**artifactId**>maven-compiler-plugin</**artifactId**>  
 <**version**>2.3.2</**version**>  
 <**configuration**>  
 <**source**>1.8</**source**>  
 <**target**>1.8</**target**>  
 </**configuration**>  
 </**plugin**>  
 <**plugin**>  
 <**artifactId**>maven-assembly-plugin</**artifactId**>  
 <**configuration**>  
 <**descriptorRefs**>  
 <**descriptorRef**>jar-with-dependencies</**descriptorRef**>  
 </**descriptorRefs**>  
 </**configuration**>  
 <**executions**>  
 <**execution**>  
 <**id**>make-assembly</**id**>  
 <**phase**>package</**phase**>  
 <**goals**>  
 <**goal**>single</**goal**>  
 </**goals**>  
 </**execution**>  
 </**executions**>  
 </**plugin**>  
 </**plugins**>  
</**build**>

4）UDF用于解析公共字段

**package** com.byf.udf;  
  
**import** org.apache.commons.lang.StringUtils;  
**import** org.apache.hadoop.hive.ql.exec.UDF;  
**import** org.json.JSONException;  
**import** org.json.JSONObject;  
  
**public class** BaseFieldUDF **extends** UDF {  
  
 **public** String evaluate(String line, String jsonkeysString) {  
   
 *// 0 准备一个sb* StringBuilder sb = **new** StringBuilder();  
  
 *// 1 切割jsonkeys mid uid vc vn l sr os ar md* String[] jsonkeys = jsonkeysString.split(**","**);  
  
 *// 2 处理line 服务器时间 | json* String[] logContents = line.split(**"\\|"**);  
  
 *// 3 合法性校验* **if** (logContents.**length** != 2 || StringUtils.*isBlank*(logContents[1])) {  
 **return ""**;  
 }  
  
 *// 4 开始处理json* **try** {  
 JSONObject jsonObject = **new** JSONObject(logContents[1]);  
  
 *// 获取cm里面的对象* JSONObject base = jsonObject.getJSONObject(**"cm"**);  
  
 *// 循环遍历取值* **for** (**int** i = 0; i < jsonkeys.**length**; i++) {  
 String filedName = jsonkeys[i].trim();  
  
 **if** (base.has(filedName)) {  
 sb.append(base.getString(filedName)).append(**"\t"**);  
 } **else** {  
 sb.append(**"\t"**);  
 }  
 }  
  
 sb.append(jsonObject.getString(**"et"**)).append(**"\t"**);  
 sb.append(logContents[0]).append(**"\t"**);  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
  
 **return** sb.toString();  
 }  
  
 **public static void** main(String[] args) {  
  
 String line = **"1541217850324|{\"cm\":{\"mid\":\"m7856\",\"uid\":\"u8739\",\"ln\":\"-74.8\",\"sv\":\"V2.2.2\",\"os\":\"8.1.3\",\"g\":\"P7XC9126@gmail.com\",\"nw\":\"3G\",\"l\":\"es\",\"vc\":\"6\",\"hw\":\"640\*960\",\"ar\":\"MX\",\"t\":\"1541204134250\",\"la\":\"-31.7\",\"md\":\"huawei-17\",\"vn\":\"1.1.2\",\"sr\":\"O\",\"ba\":\"Huawei\"},\"ap\":\"weather\",\"et\":[{\"ett\":\"1541146624055\",\"en\":\"display\",\"kv\":{\"newsid\":\"n4195\",\"copyright\":\"ESPN\",\"content\_provider\":\"CNN\",\"extend2\":\"5\",\"action\":\"2\",\"extend1\":\"2\",\"place\":\"3\",\"showtype\":\"2\",\"category\":\"72\",\"newstype\":\"5\"}},{\"ett\":\"1541213331817\",\"en\":\"loading\",\"kv\":{\"extend2\":\"\",\"loading\_time\":\"15\",\"action\":\"3\",\"extend1\":\"\",\"type1\":\"\",\"type\":\"3\",\"loading\_way\":\"1\"}},{\"ett\":\"1541126195645\",\"en\":\"ad\",\"kv\":{\"entry\":\"3\",\"show\_style\":\"0\",\"action\":\"2\",\"detail\":\"325\",\"source\":\"4\",\"behavior\":\"2\",\"content\":\"1\",\"newstype\":\"5\"}},{\"ett\":\"1541202678812\",\"en\":\"notification\",\"kv\":{\"ap\_time\":\"1541184614380\",\"action\":\"3\",\"type\":\"4\",\"content\":\"\"}},{\"ett\":\"1541194686688\",\"en\":\"active\_background\",\"kv\":{\"active\_source\":\"3\"}}]}"**;  
 String x = **new** BaseFieldUDF().evaluate(line, **"mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,nw,ln,la,t"**);  
 System.***out***.println(x);  
 }  
}

注意：使用main函数主要用于模拟数据测试。

### 3.3.3 自定义UDTF函数（解析具体事件字段）



1）创建包名：com.byf.udtf

2）在com.byf.udtf包下创建类名：EventJsonUDTF

3）用于展开业务字段

**package** com.byf.udtf;  
  
**import** org.apache.commons.lang.StringUtils;  
**import** org.apache.hadoop.hive.ql.exec.UDFArgumentException;  
**import** org.apache.hadoop.hive.ql.metadata.HiveException;  
**import** org.apache.hadoop.hive.ql.udf.generic.GenericUDTF;  
**import** org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;  
**import** org.apache.hadoop.hive.serde2.objectinspector.ObjectInspectorFactory;  
**import** org.apache.hadoop.hive.serde2.objectinspector.StructObjectInspector;  
**import** org.apache.hadoop.hive.serde2.objectinspector.primitive.PrimitiveObjectInspectorFactory;  
**import** org.json.JSONArray;  
**import** org.json.JSONException;  
  
**import** java.util.ArrayList;  
  
**public class** EventJsonUDTF **extends** GenericUDTF {  
  
 *//该方法中，我们将指定输出参数的名称和参数类型：* @Override  
 **public** StructObjectInspector initialize(ObjectInspector[] argOIs) **throws** UDFArgumentException {  
  
 ArrayList<String> fieldNames = **new** ArrayList<String>();  
 ArrayList<ObjectInspector> fieldOIs = **new** ArrayList<ObjectInspector>();  
  
 fieldNames.add(**"event\_name"**);  
 fieldOIs.add(PrimitiveObjectInspectorFactory.***javaStringObjectInspector***);  
 fieldNames.add(**"event\_json"**);  
 fieldOIs.add(PrimitiveObjectInspectorFactory.***javaStringObjectInspector***);  
  
 **return** ObjectInspectorFactory.*getStandardStructObjectInspector*(fieldNames, fieldOIs);  
 }  
  
 *//输入1条记录，输出若干条结果* @Override  
 **public void** process(Object[] objects) **throws** HiveException {  
  
 *// 获取传入的et* String input = objects[0].toString();  
  
 *// 如果传进来的数据为空，直接返回过滤掉该数据* **if** (StringUtils.*isBlank*(input)) {  
 **return**;  
 } **else** {  
  
 **try** {  
 *// 获取一共有几个事件（ad/facoriters）* JSONArray ja = **new** JSONArray(input);  
  
 **if** (ja == **null**)  
 **return**;  
  
 *// 循环遍历每一个事件* **for** (**int** i = 0; i < ja.length(); i++) {  
 String[] result = **new** String[2];  
  
 **try** {  
 *// 取出每个的事件名称（ad/facoriters）* result[0] = ja.getJSONObject(i).getString(**"en"**);  
  
 *// 取出每一个事件整体* result[1] = ja.getString(i);  
 } **catch** (JSONException e) {  
 **continue**;  
 }  
  
 *// 将结果返回* forward(result);  
 }  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 *//当没有记录处理的时候该方法会被调用，用来清理代码或者产生额外的输出* @Override  
 **public void** close() **throws** HiveException {  
  
 }  
}

2）打包



3）将hivefunction-1.0-SNAPSHOT上传到hadoop102的/opt/module/hive/

4）将jar包添加到Hive的classpath

hive (gmall)> add jar /opt/module/hive/hivefunction-1.0-SNAPSHOT.jar;

5）创建临时函数与开发好的java class关联

hive (gmall)>

create temporary function base\_analizer as 'com.byf.udf.BaseFieldUDF';

create temporary function flat\_analizer as 'com.byf.udtf.EventJsonUDTF';

### 3.3.4 解析启动日志基础明细表

1）解析启动日志基础明细表

hive (gmall)>

use gmall;

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_base\_start\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source ,

os ,

area ,

model ,

brand ,

sdk\_version ,

gmail ,

height\_width ,

app\_time ,

network ,

lng ,

lat ,

event\_name ,

event\_json ,

server\_time ,

dt

from

(

select

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[0] as mid\_id,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[1] as user\_id,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[2] as version\_code,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[3] as version\_name,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[4] as lang,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[5] as source,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[6] as os,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[7] as area,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[8] as model,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[9] as brand,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[10] as sdk\_version,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[11] as gmail,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[12] as height\_width,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[13] as app\_time,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[14] as network,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[15] as lng,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[16] as lat,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[17] as ops,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[18] as server\_time,

dt

from ods\_start\_log where dt='2019-02-10' and base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la')<>''

) sdk\_log lateral view flat\_analizer(ops) tmp\_k as event\_name, event\_json;

2）测试

hive (gmall)> select \* from dwd\_base\_start\_log limit 2;

### 3.3.5 解析事件日志基础明细表

1）解析事件日志基础明细表

hive (gmall)>

use gmall;

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_base\_event\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source ,

os ,

area ,

model ,

brand ,

sdk\_version ,

gmail ,

height\_width ,

app\_time ,

network ,

lng ,

lat ,

event\_name ,

event\_json ,

server\_time ,

dt

from

(

select

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[0] as mid\_id,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[1] as user\_id,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[2] as version\_code,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[3] as version\_name,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[4] as lang,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[5] as source,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[6] as os,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[7] as area,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[8] as model,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[9] as brand,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[10] as sdk\_version,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[11] as gmail,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[12] as height\_width,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[13] as app\_time,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[14] as network,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[15] as lng,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[16] as lat,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[17] as ops,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[18] as server\_time,

dt

from ods\_event\_log where dt='2019-02-10' and base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la')<>''

) sdk\_log lateral view flat\_analizer(ops) tmp\_k as event\_name, event\_json;

2）测试

hive (gmall)> select \* from dwd\_base\_event\_log limit 2;

### 3.3.6 DWD层数据解析脚本

1）在hadoop102的/home/byf/bin目录下创建脚本

[byf@hadoop102 bin]$ vim dwd\_base.sh

在脚本中编写如下内容

#!/bin/bash

# 定义变量方便修改

APP=gmall

hive=/opt/module/hive/bin/hive

# 如果是输入的日期按照取输入日期；如果没输入日期取当前时间的前一天

if [ -n "$1" ] ;then

do\_date=$1

else

do\_date=`date -d "-1 day" +%F`

fi

sql="

add jar /opt/module/hive/hivefunction-1.0-SNAPSHOT.jar;

create temporary function base\_analizer as 'com.byf.udf.BaseFieldUDF';

create temporary function flat\_analizer as 'com.byf.udtf.EventJsonUDTF';

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table "$APP".dwd\_base\_start\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source ,

os ,

area ,

model ,

brand ,

sdk\_version ,

gmail ,

height\_width ,

network ,

lng ,

lat ,

app\_time ,

event\_name ,

event\_json ,

server\_time ,

dt

from

(

select

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[0] as mid\_id,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[1] as user\_id,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[2] as version\_code,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[3] as version\_name,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[4] as lang,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[5] as source,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[6] as os,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[7] as area,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[8] as model,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[9] as brand,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[10] as sdk\_version,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[11] as gmail,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[12] as height\_width,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[13] as app\_time,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[14] as network,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[15] as lng,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[16] as lat,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[17] as ops,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[18] as server\_time,

dt

from "$APP".ods\_start\_log where dt='$do\_date' and base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la')<>''

) sdk\_log lateral view flat\_analizer(ops) tmp\_k as event\_name, event\_json;

insert overwrite table "$APP".dwd\_base\_event\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source ,

os ,

area ,

model ,

brand ,

sdk\_version ,

gmail ,

height\_width ,

network ,

lng ,

lat ,

app\_time ,

event\_name ,

event\_json ,

server\_time ,

dt

from

(

select

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[0] as mid\_id,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[1] as user\_id,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[2] as version\_code,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[3] as version\_name,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[4] as lang,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[5] as source,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[6] as os,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[7] as area,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[8] as model,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[9] as brand,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[10] as sdk\_version,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[11] as gmail,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[12] as height\_width,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[13] as app\_time,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[14] as network,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[15] as lng,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[16] as lat,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[17] as ops,

split(base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la'),'\t')[18] as server\_time,

dt

from "$APP".ods\_event\_log where dt='$do\_date' and base\_analizer(line,'mid,uid,vc,vn,l,sr,os,ar,md,ba,sv,g,hw,t,nw,ln,la')<>''

) sdk\_log lateral view flat\_analizer(ops) tmp\_k as event\_name, event\_json;

"

$hive -e "$sql"

2）增加脚本执行权限

[byf@hadoop102 bin]$ chmod 777 dwd\_base.sh

3）脚本使用

[byf@hadoop102 module]$ dwd\_base.sh 2019-02-11

4）查询导入结果

hive (gmall)>

select \* from dwd\_base\_start\_log where dt='2019-02-11' limit 2;

select \* from dwd\_base\_event\_log where dt='2019-02-11' limit 2;

5）脚本执行时间

企业开发中一般在每日凌晨30分~1点

## 3.4 DWD层



### 3.4.1 商品点击表



1）建表语句

hive (gmall)>

drop table if exists dwd\_display\_log;

CREATE EXTERNAL TABLE dwd\_display\_log(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`action` string,

`newsid` string,

`place` string,

`extend1` string,

`category` string,

`server\_time` string

)

PARTITIONED BY (dt string)

location '/warehouse/gmall/dwd/dwd\_display\_log/';

2）导入数据

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_display\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.action') action,

get\_json\_object(event\_json,'$.kv.newsid') newsid,

get\_json\_object(event\_json,'$.kv.place') place,

get\_json\_object(event\_json,'$.kv.extend1') extend1,

get\_json\_object(event\_json,'$.kv.category') category,

server\_time,

dt

from dwd\_base\_event\_log

where dt='2019-02-10' and event\_name='display';

3）测试

hive (gmall)> select \* from dwd\_display\_log limit 2;

### 3.4.2 商品详情页表

1）建表语句

hive (gmall)>

drop table if exists dwd\_newsdetail\_log;

CREATE EXTERNAL TABLE `dwd\_newsdetail\_log`(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`entry` string,

`action` string,

`newsid` string,

`showtype` string,

`news\_staytime` string,

`loading\_time` string,

`type1` string,

`category` string,

`server\_time` string)

PARTITIONED BY (dt string)

location '/warehouse/gmall/dwd/dwd\_newsdetail\_log/';

2）导入数据

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_newsdetail\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.entry') entry,

get\_json\_object(event\_json,'$.kv.action') action,

get\_json\_object(event\_json,'$.kv.newsid') newsid,

get\_json\_object(event\_json,'$.kv.showtype') showtype,

get\_json\_object(event\_json,'$.kv.news\_staytime') news\_staytime,

get\_json\_object(event\_json,'$.kv.loading\_time') loading\_time,

get\_json\_object(event\_json,'$.kv.type1') type1,

get\_json\_object(event\_json,'$.kv.category') category,

server\_time,

dt

from dwd\_base\_event\_log

where dt='2019-02-10' and event\_name='newsdetail';

3）测试

hive (gmall)> select \* from dwd\_newsdetail\_log limit 2;

### 3.4.3 商品列表页表

1）建表语句

hive (gmall)>

drop table if exists dwd\_loading\_log;

CREATE EXTERNAL TABLE `dwd\_loading\_log`(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`action` string,

`loading\_time` string,

`loading\_way` string,

`extend1` string,

`extend2` string,

`type` string,

`type1` string,

`server\_time` string)

PARTITIONED BY (dt string)

location '/warehouse/gmall/dwd/dwd\_loading\_log/';

2）导入数据

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_loading\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.action') action,

get\_json\_object(event\_json,'$.kv.loading\_time') loading\_time,

get\_json\_object(event\_json,'$.kv.loading\_way') loading\_way,

get\_json\_object(event\_json,'$.kv.extend1') extend1,

get\_json\_object(event\_json,'$.kv.extend2') extend2,

get\_json\_object(event\_json,'$.kv.type') type,

get\_json\_object(event\_json,'$.kv.type1') type1,

server\_time,

dt

from dwd\_base\_event\_log

where dt='2019-02-10' and event\_name='loading';

3）测试

hive (gmall)> select \* from dwd\_loading\_log limit 2;

### 3.4.4 广告表

1）建表语句

hive (gmall)>

drop table if exists dwd\_ad\_log;

CREATE EXTERNAL TABLE `dwd\_ad\_log`(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`entry` string,

`action` string,

`content` string,

`detail` string,

`ad\_source` string,

`behavior` string,

`newstype` string,

`show\_style` string,

`server\_time` string)

PARTITIONED BY (dt string)

location '/warehouse/gmall/dwd/dwd\_ad\_log/';

2）导入数据

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_ad\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.entry') entry,

get\_json\_object(event\_json,'$.kv.action') action,

get\_json\_object(event\_json,'$.kv.content') content,

get\_json\_object(event\_json,'$.kv.detail') detail,

get\_json\_object(event\_json,'$.kv.source') ad\_source,

get\_json\_object(event\_json,'$.kv.behavior') behavior,

get\_json\_object(event\_json,'$.kv.newstype') newstype,

get\_json\_object(event\_json,'$.kv.show\_style') show\_style,

server\_time,

dt

from dwd\_base\_event\_log

where dt='2019-02-10' and event\_name='ad';

3）测试

hive (gmall)> select \* from dwd\_ad\_log limit 2;

### 3.4.5 消息通知表

1）建表语句

hive (gmall)>

drop table if exists dwd\_notification\_log;

CREATE EXTERNAL TABLE `dwd\_notification\_log`(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`action` string,

`noti\_type` string,

`ap\_time` string,

`content` string,

`server\_time` string

)

PARTITIONED BY (dt string)

location '/warehouse/gmall/dwd/dwd\_notification\_log/';

2）导入数据

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_notification\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.action') action,

get\_json\_object(event\_json,'$.kv.noti\_type') noti\_type,

get\_json\_object(event\_json,'$.kv.ap\_time') ap\_time,

get\_json\_object(event\_json,'$.kv.content') content,

server\_time,

dt

from dwd\_base\_event\_log

where dt='2019-02-10' and event\_name='notification';

3）测试

hive (gmall)> select \* from dwd\_notification\_log limit 2;

### 3.4.6 用户前台活跃表

1）建表语句

hive (gmall)>

drop table if exists dwd\_active\_foreground\_log;

CREATE EXTERNAL TABLE `dwd\_active\_foreground\_log`(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`active\_source` string,

`server\_time` string)

PARTITIONED BY (dt string)

location '/warehouse/gmall/dwd/dwd\_foreground\_log/';

2）导入数据

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_active\_foreground\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.active\_source') active\_source,

server\_time,

dt

from dwd\_base\_event\_log

where dt='2019-02-10' and event\_name='active\_foreground';

3）测试

hive (gmall)> select \* from dwd\_active\_foreground\_log limit 2;

### 3.4.7 用户后台活跃表

1）建表语句

hive (gmall)>

drop table if exists dwd\_active\_background\_log;

CREATE EXTERNAL TABLE `dwd\_active\_background\_log`(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`active\_source` string,

`server\_time` string

)

PARTITIONED BY (dt string)

location '/warehouse/gmall/dwd/dwd\_background\_log/';

2）导入数据

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_active\_background\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.active\_source') active\_source,

server\_time,

dt

from dwd\_base\_event\_log

where dt='2019-02-10' and event\_name='active\_background';

3）测试

hive (gmall)> select \* from dwd\_active\_background\_log limit 2;

### 3.4.8 评论表

1）建表语句

hive (gmall)>

drop table if exists dwd\_comment\_log;

CREATE EXTERNAL TABLE `dwd\_comment\_log`(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`comment\_id` int,

`userid` int,

`p\_comment\_id` int,

`content` string,

`addtime` string,

`other\_id` int,

`praise\_count` int,

`reply\_count` int,

`server\_time` string

)

PARTITIONED BY (dt string)

location '/warehouse/gmall/dwd/dwd\_comment\_log/';

2）导入数据

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_comment\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.comment\_id') comment\_id,

get\_json\_object(event\_json,'$.kv.userid') userid,

get\_json\_object(event\_json,'$.kv.p\_comment\_id') p\_comment\_id,

get\_json\_object(event\_json,'$.kv.content') content,

get\_json\_object(event\_json,'$.kv.addtime') addtime,

get\_json\_object(event\_json,'$.kv.other\_id') other\_id,

get\_json\_object(event\_json,'$.kv.praise\_count') praise\_count,

get\_json\_object(event\_json,'$.kv.reply\_count') reply\_count,

server\_time,

dt

from dwd\_base\_event\_log

where dt='2019-02-10' and event\_name='comment';

3）测试

hive (gmall)> select \* from dwd\_comment\_log limit 2;

### 3.4.9 收藏表

1）建表语句

hive (gmall)>

drop table if exists dwd\_favorites\_log;

CREATE EXTERNAL TABLE `dwd\_favorites\_log`(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`id` int,

`course\_id` int,

`userid` int,

`add\_time` string,

`server\_time` string

)

PARTITIONED BY (dt string)

location '/warehouse/gmall/dwd/dwd\_favorites\_log/';

2）导入数据

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_favorites\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.id') id,

get\_json\_object(event\_json,'$.kv.course\_id') course\_id,

get\_json\_object(event\_json,'$.kv.userid') userid,

get\_json\_object(event\_json,'$.kv.add\_time') add\_time,

server\_time,

dt

from dwd\_base\_event\_log

where dt='2019-02-10' and event\_name='favorites';

3）测试

hive (gmall)> select \* from dwd\_favorites\_log limit 2;

### 3.4.10 点赞表

1）建表语句

hive (gmall)>

drop table if exists dwd\_praise\_log;

CREATE EXTERNAL TABLE `dwd\_praise\_log`(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`id` string,

`userid` string,

`target\_id` string,

`type` string,

`add\_time` string,

`server\_time` string

)

PARTITIONED BY (dt string)

location '/warehouse/gmall/dwd/dwd\_praise\_log/';

2）导入数据

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_praise\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.id') id,

get\_json\_object(event\_json,'$.kv.userid') userid,

get\_json\_object(event\_json,'$.kv.target\_id') target\_id,

get\_json\_object(event\_json,'$.kv.type') type,

get\_json\_object(event\_json,'$.kv.add\_time') add\_time,

server\_time,

dt

from dwd\_base\_event\_log

where dt='2019-02-10' and event\_name='praise';

3）测试

hive (gmall)> select \* from dwd\_praise\_log limit 2;

### 3.4.11 启动日志表

1）建表语句

hive (gmall)>

drop table if exists dwd\_start\_log;

CREATE EXTERNAL TABLE `dwd\_start\_log`(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`entry` string,

`open\_ad\_type` string,

`action` string,

`loading\_time` string,

`detail` string,

`extend1` string,

`server\_time` string

)

PARTITIONED BY (dt string)

location '/warehouse/gmall/dwd/dwd\_start\_log/';

2）导入数据

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_start\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.entry') entry,

get\_json\_object(event\_json,'$.kv.open\_ad\_type') open\_ad\_type,

get\_json\_object(event\_json,'$.kv.action') action,

get\_json\_object(event\_json,'$.kv.loading\_time') loading\_time,

get\_json\_object(event\_json,'$.kv.detail') detail,

get\_json\_object(event\_json,'$.kv.extend1') extend1,

server\_time,

dt

from dwd\_base\_start\_log

where dt='2019-02-10' and event\_name='start';

3）测试

hive (gmall)> select \* from dwd\_start\_log limit 2;

### 3.4.12 错误日志表

1）建表语句

hive (gmall)>

drop table if exists dwd\_error\_log;

CREATE EXTERNAL TABLE `dwd\_error\_log`(

`mid\_id` string,

`user\_id` string,

`version\_code` string,

`version\_name` string,

`lang` string,

`source` string,

`os` string,

`area` string,

`model` string,

`brand` string,

`sdk\_version` string,

`gmail` string,

`height\_width` string,

`app\_time` string,

`network` string,

`lng` string,

`lat` string,

`errorBrief` string,

`errorDetail` string,

`server\_time` string)

PARTITIONED BY (dt string)

location '/warehouse/gmall/dwd/dwd\_error\_log/';

2）导入数据

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dwd\_error\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.errorBrief') errorBrief,

get\_json\_object(event\_json,'$.kv.errorDetail') errorDetail,

server\_time,

dt

from dwd\_base\_event\_log

where dt='2019-02-10' and event\_name='error';

3）测试

hive (gmall)> select \* from dwd\_error\_log limit 2;

### 3.4.13 DWD层加载数据脚本

1）在hadoop102的/home/byf/bin目录下创建脚本

[byf@hadoop102 bin]$ vim dwd.sh

在脚本中编写如下内容

#!/bin/bash

# 定义变量方便修改

APP=gmall

hive=/opt/module/hive/bin/hive

# 如果是输入的日期按照取输入日期；如果没输入日期取当前时间的前一天

if [ -n "$1" ] ;then

do\_date=$1

else

do\_date=`date -d "-1 day" +%F`

fi

sql="

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table "$APP".dwd\_display\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.action') action,

get\_json\_object(event\_json,'$.kv.newsid') newsid,

get\_json\_object(event\_json,'$.kv.place') place,

get\_json\_object(event\_json,'$.kv.extend1') extend1,

get\_json\_object(event\_json,'$.kv.category') category,

server\_time,

dt

from "$APP".dwd\_base\_event\_log

where dt='$do\_date' and event\_name='display';

insert overwrite table "$APP".dwd\_newsdetail\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.entry') entry,

get\_json\_object(event\_json,'$.kv.action') action,

get\_json\_object(event\_json,'$.kv.newsid') newsid,

get\_json\_object(event\_json,'$.kv.showtype') showtype,

get\_json\_object(event\_json,'$.kv.news\_staytime') news\_staytime,

get\_json\_object(event\_json,'$.kv.loading\_time') loading\_time,

get\_json\_object(event\_json,'$.kv.type1') type1,

get\_json\_object(event\_json,'$.kv.category') category,

server\_time,

dt

from "$APP".dwd\_base\_event\_log

where dt='$do\_date' and event\_name='newsdetail';

insert overwrite table "$APP".dwd\_loading\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.action') action,

get\_json\_object(event\_json,'$.kv.loading\_time') loading\_time,

get\_json\_object(event\_json,'$.kv.loading\_way') loading\_way,

get\_json\_object(event\_json,'$.kv.extend1') extend1,

get\_json\_object(event\_json,'$.kv.extend2') extend2,

get\_json\_object(event\_json,'$.kv.type') type,

get\_json\_object(event\_json,'$.kv.type1') type1,

server\_time,

dt

from "$APP".dwd\_base\_event\_log

where dt='$do\_date' and event\_name='loading';

insert overwrite table "$APP".dwd\_ad\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.entry') entry,

get\_json\_object(event\_json,'$.kv.action') action,

get\_json\_object(event\_json,'$.kv.content') content,

get\_json\_object(event\_json,'$.kv.detail') detail,

get\_json\_object(event\_json,'$.kv.source') ad\_source,

get\_json\_object(event\_json,'$.kv.behavior') behavior,

get\_json\_object(event\_json,'$.kv.newstype') newstype,

get\_json\_object(event\_json,'$.kv.show\_style') show\_style,

server\_time,

dt

from "$APP".dwd\_base\_event\_log

where dt='$do\_date' and event\_name='ad';

insert overwrite table "$APP".dwd\_notification\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.action') action,

get\_json\_object(event\_json,'$.kv.noti\_type') noti\_type,

get\_json\_object(event\_json,'$.kv.ap\_time') ap\_time,

get\_json\_object(event\_json,'$.kv.content') content,

server\_time,

dt

from "$APP".dwd\_base\_event\_log

where dt='$do\_date' and event\_name='notification';

insert overwrite table "$APP".dwd\_active\_foreground\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.active\_source') active\_source,

server\_time,

dt

from "$APP".dwd\_base\_event\_log

where dt='$do\_date' and event\_name='active\_background';

insert overwrite table "$APP".dwd\_active\_background\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.active\_source') active\_source,

server\_time,

dt

from "$APP".dwd\_base\_event\_log

where dt='$do\_date' and event\_name='active\_background';

insert overwrite table "$APP".dwd\_comment\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.comment\_id') comment\_id,

get\_json\_object(event\_json,'$.kv.userid') userid,

get\_json\_object(event\_json,'$.kv.p\_comment\_id') p\_comment\_id,

get\_json\_object(event\_json,'$.kv.content') content,

get\_json\_object(event\_json,'$.kv.addtime') addtime,

get\_json\_object(event\_json,'$.kv.other\_id') other\_id,

get\_json\_object(event\_json,'$.kv.praise\_count') praise\_count,

get\_json\_object(event\_json,'$.kv.reply\_count') reply\_count,

server\_time,

dt

from "$APP".dwd\_base\_event\_log

where dt='$do\_date' and event\_name='comment';

insert overwrite table "$APP".dwd\_favorites\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.id') id,

get\_json\_object(event\_json,'$.kv.course\_id') course\_id,

get\_json\_object(event\_json,'$.kv.userid') userid,

get\_json\_object(event\_json,'$.kv.add\_time') add\_time,

server\_time,

dt

from "$APP".dwd\_base\_event\_log

where dt='$do\_date' and event\_name='favorites';

insert overwrite table "$APP".dwd\_praise\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.id') id,

get\_json\_object(event\_json,'$.kv.userid') userid,

get\_json\_object(event\_json,'$.kv.target\_id') target\_id,

get\_json\_object(event\_json,'$.kv.type') type,

get\_json\_object(event\_json,'$.kv.add\_time') add\_time,

server\_time,

dt

from "$APP".dwd\_base\_event\_log

where dt='$do\_date' and event\_name='praise';

insert overwrite table "$APP".dwd\_start\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.entry') entry,

get\_json\_object(event\_json,'$.kv.open\_ad\_type') open\_ad\_type,

get\_json\_object(event\_json,'$.kv.action') action,

get\_json\_object(event\_json,'$.kv.loading\_time') loading\_time,

get\_json\_object(event\_json,'$.kv.detail') detail,

get\_json\_object(event\_json,'$.kv.extend1') extend1,

server\_time,

dt

from "$APP".dwd\_base\_start\_log

where dt='$do\_date' and event\_name='start';

insert overwrite table "$APP".dwd\_error\_log

PARTITION (dt)

select

mid\_id,

user\_id,

version\_code,

version\_name,

lang,

source,

os,

area,

model,

brand,

sdk\_version,

gmail,

height\_width,

app\_time,

network,

lng,

lat,

get\_json\_object(event\_json,'$.kv.errorBrief') errorBrief,

get\_json\_object(event\_json,'$.kv.errorDetail') errorDetail,

server\_time,

dt

from "$APP".dwd\_base\_event\_log

where dt='$do\_date' and event\_name='error';

"

$hive -e "$sql"

2）增加脚本执行权限

[byf@hadoop102 bin]$ chmod 777 dwd.sh

3）脚本使用

[byf@hadoop102 module]$ dwd.sh 2019-02-11

4）查询导入结果

hive (gmall)>

select \* from dwd\_start\_log where dt='2019-02-11' limit 2;

select \* from dwd\_comment\_log where dt='2019-02-11' limit 2;

5）脚本执行时间

企业开发中一般在每日凌晨30分~1点

# 第4章 业务知识准备

## 4.1 业务术语

1. 用户

用户以设备为判断标准，在移动统计中，每个独立设备认为是一个独立用户。Android系统根据IMEI号，IOS系统根据OpenUDID来标识一个独立用户，每部手机一个用户。

1. 新增用户

首次联网使用应用的用户。如果一个用户首次打开某app，那这个用户定义为新增用户；卸载再安装的设备，不会被算作一次新增。新增用户包括日新增用户、周新增用户、月新增用户。

1. 活跃用户

打开应用的用户即为活跃用户，不考虑用户的使用情况。每天一台设备打开多次会被计为一个活跃用户。

1. 周（月）活跃用户

某个自然周（月）内启动过应用的用户，该周（月）内的多次启动只记一个活跃用户。

1. 月活跃率

月活跃用户与截止到该月累计的用户总和之间的比例。

1. 沉默用户

用户仅在安装当天（次日）启动一次，后续时间无再启动行为。该指标可以反映新增用户质量和用户与APP的匹配程度。

1. 版本分布

不同版本的周内各天新增用户数，活跃用户数和启动次数。利于判断App各个版本之间的优劣和用户行为习惯。

1. 本周回流用户

上周未启动过应用，本周启动了应用的用户。

1. 连续n周活跃用户

连续n周，每周至少启动一次。

1. 忠诚用户

连续活跃5周以上的用户

1. 连续活跃用户

连续2周及以上活跃的用户

1. 近期流失用户

连续n(2<= n <= 4)周没有启动应用的用户。（第n+1周没有启动过）

1. 留存用户

某段时间内的新增用户，经过一段时间后，仍然使用应用的被认作是留存用户；这部分用户占当时新增用户的比例即是留存率。例如，5月份新增用户200，这200人在6月份启动过应用的有100人，7月份启动过应用的有80人，8月份启动过应用的有50人；则5月份新增用户一个月后的留存率是50%，二个月后的留存率是40%，三个月后的留存率是25%。

1. 用户新鲜度

每天启动应用的新老用户比例，即新增用户数占活跃用户数的比例。

1. 单次使用时长

每次启动使用的时间长度。

1. 日使用时长

累计一天内的使用时间长度。

1. 启动次数计算标准

IOS平台应用退到后台就算一次独立的启动；Android平台我们规定，两次启动之间的间隔小于30秒，被计算一次启动。用户在使用过程中，若因收发短信或接电话等退出应用30秒又再次返回应用中，那这两次行为应该是延续而非独立的，所以可以被算作一次使用行为，即一次启动。业内大多使用30秒这个标准，但用户还是可以自定义此时间间隔。

## 4.2 系统函数

### 4.2.1 collect\_set函数

1）创建原数据表

hive (gmall)>

drop table if exists stud;

create table stud (name string, area string, course string, score int);

2）向原数据表中插入数据

hive (gmall)>

insert into table stud values('zhang3','bj','math',88);

insert into table stud values('li4','bj','math',99);

insert into table stud values('wang5','sh','chinese',92);

insert into table stud values('zhao6','sh','chinese',54);

insert into table stud values('tian7','bj','chinese',91);

3）查询表中数据

hive (gmall)> select \* from stud;

stud.name stud.area stud.course stud.score

zhang3 bj math 88

li4 bj math 99

wang5 sh chinese 92

zhao6 sh chinese 54

tian7 bj chinese 91

4）把同一分组的不同行的数据聚合成一个集合

hive (gmall)> select course, collect\_set(area), avg(score) from stud group by course;

chinese ["sh","bj"] 79.0

math ["bj"] 93.5

5） 用下标可以取某一个

hive (gmall)> select course, collect\_set(area)[0], avg(score) from stud group by course;

chinese sh 79.0

math bj 93.5

### 4.2.2 日期处理函数

1）date\_format函数（根据格式整理日期）

hive (gmall)> select date\_format('2019-02-10','yyyy-MM');

2019-02

2）date\_add函数（加减日期）

hive (gmall)> select date\_add('2019-02-10',-1);

2019-02-09

hive (gmall)> select date\_add('2019-02-10',1);

2019-02-11

3）next\_day函数

（1）取当前天的下一个周一

hive (gmall)> select next\_day('2019-02-12','MO')

2019-02-18

（2）取当前周的周一

hive (gmall)> select date\_add(next\_day('2019-02-12','MO'),-7);

2019-02-11

4）last\_day函数（求当月最后一天日期）

hive (gmall)> select last\_day('2019-02-10');

2019-02-28

# 第5章 需求一：用户活跃主题

## 5.1 DWS层

目标：统计当日、当周、当月活动的每个设备明细

### 5.1.1 每日活跃设备明细



1）建表语句

hive (gmall)>

drop table if exists dws\_uv\_detail\_day;

create table dws\_uv\_detail\_day(

  `mid\_id` string COMMENT '设备唯一标识',

    `user\_id` string COMMENT '用户标识',

    `version\_code` string COMMENT '程序版本号',

    `version\_name` string COMMENT '程序版本名',

`lang` string COMMENT '系统语言',

`source` string COMMENT '渠道号',

`os` string COMMENT '安卓系统版本',

`area` string COMMENT '区域',

`model` string COMMENT '手机型号',

`brand` string COMMENT '手机品牌',

`sdk\_version` string COMMENT 'sdkVersion',

`gmail` string COMMENT 'gmail',

`height\_width` string COMMENT '屏幕宽高',

`app\_time` string COMMENT '客户端日志产生时的时间',

`network` string COMMENT '网络模式',

`lng` string COMMENT '经度',

`lat` string COMMENT '纬度'

) COMMENT '活跃用户按天明细'

PARTITIONED BY ( `dt` string)

stored as parquet

location '/warehouse/gmall/dws/dws\_uv\_detail\_day/'

;

2）数据导入

以用户单日访问为key进行聚合，如果某个用户在一天中使用了两种操作系统、两个系统版本、多个地区，登录不同账号，只取其中之一

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dws\_uv\_detail\_day partition(dt='2019-02-10')

select

mid\_id,

concat\_ws('|', collect\_set(user\_id)) user\_id,

concat\_ws('|', collect\_set(version\_code)) version\_code,

concat\_ws('|', collect\_set(version\_name)) version\_name,

concat\_ws('|', collect\_set(lang))lang,

concat\_ws('|', collect\_set(source)) source,

concat\_ws('|', collect\_set(os)) os,

concat\_ws('|', collect\_set(area)) area,

concat\_ws('|', collect\_set(model)) model,

concat\_ws('|', collect\_set(brand)) brand,

concat\_ws('|', collect\_set(sdk\_version)) sdk\_version,

concat\_ws('|', collect\_set(gmail)) gmail,

concat\_ws('|', collect\_set(height\_width)) height\_width,

concat\_ws('|', collect\_set(app\_time)) app\_time,

concat\_ws('|', collect\_set(network)) network,

concat\_ws('|', collect\_set(lng)) lng,

concat\_ws('|', collect\_set(lat)) lat

from dwd\_start\_log

where dt='2019-02-10'

group by mid\_id;

3）查询导入结果

hive (gmall)> select \* from dws\_uv\_detail\_day limit 1;

hive (gmall)> select count(\*) from dws\_uv\_detail\_day;

4）思考：不同渠道来源的每日活跃数统计怎么计算？

### 5.1.2 每周活跃设备明细



根据日用户访问明细 ，获得周用户访问明细

1）建表语句

hive (gmall)>

drop table if exists dws\_uv\_detail\_wk;

create table dws\_uv\_detail\_wk(

  `mid\_id` string COMMENT '设备唯一标识',

    `user\_id` string COMMENT '用户标识',

    `version\_code` string COMMENT '程序版本号',

    `version\_name` string COMMENT '程序版本名',

`lang` string COMMENT '系统语言',

`source` string COMMENT '渠道号',

`os` string COMMENT '安卓系统版本',

`area` string COMMENT '区域',

`model` string COMMENT '手机型号',

`brand` string COMMENT '手机品牌',

`sdk\_version` string COMMENT 'sdkVersion',

`gmail` string COMMENT 'gmail',

`height\_width` string COMMENT '屏幕宽高',

`app\_time` string COMMENT '客户端日志产生时的时间',

`network` string COMMENT '网络模式',

`lng` string COMMENT '经度',

`lat` string COMMENT '纬度',

    `monday\_date` string COMMENT '周一日期',

    `sunday\_date` string COMMENT '周日日期'

) COMMENT '活跃用户按周明细'

PARTITIONED BY (`wk\_dt` string)

stored as parquet

location '/warehouse/gmall/dws/dws\_uv\_detail\_wk/'

;

2）数据导入

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dws\_uv\_detail\_wk partition(wk\_dt)

select

mid\_id,

concat\_ws('|', collect\_set(user\_id)) user\_id,

concat\_ws('|', collect\_set(version\_code)) version\_code,

concat\_ws('|', collect\_set(version\_name)) version\_name,

concat\_ws('|', collect\_set(lang)) lang,

concat\_ws('|', collect\_set(source)) source,

concat\_ws('|', collect\_set(os)) os,

concat\_ws('|', collect\_set(area)) area,

concat\_ws('|', collect\_set(model)) model,

concat\_ws('|', collect\_set(brand)) brand,

concat\_ws('|', collect\_set(sdk\_version)) sdk\_version,

concat\_ws('|', collect\_set(gmail)) gmail,

concat\_ws('|', collect\_set(height\_width)) height\_width,

concat\_ws('|', collect\_set(app\_time)) app\_time,

concat\_ws('|', collect\_set(network)) network,

concat\_ws('|', collect\_set(lng)) lng,

concat\_ws('|', collect\_set(lat)) lat,

date\_add(next\_day('2019-02-10','MO'),-7),

date\_add(next\_day('2019-02-10','MO'),-1),

concat(date\_add( next\_day('2019-02-10','MO'),-7), '\_' , date\_add(next\_day('2019-02-10','MO'),-1)

)

from dws\_uv\_detail\_day

where dt>=date\_add(next\_day('2019-02-10','MO'),-7) and dt<=date\_add(next\_day('2019-02-10','MO'),-1)

group by mid\_id;

3）查询导入结果

hive (gmall)> select \* from dws\_uv\_detail\_wk limit 1;

hive (gmall)> select count(\*) from dws\_uv\_detail\_wk;

### 5.1.3 每月活跃设备明细



1）建表语句

hive (gmall)>

drop table if exists dws\_uv\_detail\_mn;

create external table dws\_uv\_detail\_mn(

  `mid\_id` string COMMENT '设备唯一标识',

    `user\_id` string COMMENT '用户标识',

    `version\_code` string COMMENT '程序版本号',

    `version\_name` string COMMENT '程序版本名',

`lang` string COMMENT '系统语言',

`source` string COMMENT '渠道号',

`os` string COMMENT '安卓系统版本',

`area` string COMMENT '区域',

`model` string COMMENT '手机型号',

`brand` string COMMENT '手机品牌',

`sdk\_version` string COMMENT 'sdkVersion',

`gmail` string COMMENT 'gmail',

`height\_width` string COMMENT '屏幕宽高',

`app\_time` string COMMENT '客户端日志产生时的时间',

`network` string COMMENT '网络模式',

`lng` string COMMENT '经度',

`lat` string COMMENT '纬度'

) COMMENT '活跃用户按月明细'

PARTITIONED BY (`mn` string)

stored as parquet

location '/warehouse/gmall/dws/dws\_uv\_detail\_mn/'

;

2）数据导入

hive (gmall)>

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table dws\_uv\_detail\_mn partition(mn)

select

mid\_id,

concat\_ws('|', collect\_set(user\_id)) user\_id,

concat\_ws('|', collect\_set(version\_code)) version\_code,

concat\_ws('|', collect\_set(version\_name)) version\_name,

concat\_ws('|', collect\_set(lang)) lang,

concat\_ws('|', collect\_set(source)) source,

concat\_ws('|', collect\_set(os)) os,

concat\_ws('|', collect\_set(area)) area,

concat\_ws('|', collect\_set(model)) model,

concat\_ws('|', collect\_set(brand)) brand,

concat\_ws('|', collect\_set(sdk\_version)) sdk\_version,

concat\_ws('|', collect\_set(gmail)) gmail,

concat\_ws('|', collect\_set(height\_width)) height\_width,

concat\_ws('|', collect\_set(app\_time)) app\_time,

concat\_ws('|', collect\_set(network)) network,

concat\_ws('|', collect\_set(lng)) lng,

concat\_ws('|', collect\_set(lat)) lat,

date\_format('2019-02-10','yyyy-MM')

from dws\_uv\_detail\_day

where date\_format(dt,'yyyy-MM') = date\_format('2019-02-10','yyyy-MM')

group by mid\_id;

3）查询导入结果

hive (gmall)> select \* from dws\_uv\_detail\_mn limit 1;

hive (gmall)> select count(\*) from dws\_uv\_detail\_mn ;

### 5.1.4 DWS层加载数据脚本

1）在hadoop102的/home/byf/bin目录下创建脚本

[byf@hadoop102 bin]$ vim dws.sh

在脚本中编写如下内容

#!/bin/bash

# 定义变量方便修改

APP=gmall

hive=/opt/module/hive/bin/hive

# 如果是输入的日期按照取输入日期；如果没输入日期取当前时间的前一天

if [ -n "$1" ] ;then

do\_date=$1

else

do\_date=`date -d "-1 day" +%F`

fi

sql="

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table "$APP".dws\_uv\_detail\_day partition(dt='$do\_date')

select

mid\_id,

concat\_ws('|', collect\_set(user\_id)) user\_id,

concat\_ws('|', collect\_set(version\_code)) version\_code,

concat\_ws('|', collect\_set(version\_name)) version\_name,

concat\_ws('|', collect\_set(lang)) lang,

concat\_ws('|', collect\_set(source)) source,

concat\_ws('|', collect\_set(os)) os,

concat\_ws('|', collect\_set(area)) area,

concat\_ws('|', collect\_set(model)) model,

concat\_ws('|', collect\_set(brand)) brand,

concat\_ws('|', collect\_set(sdk\_version)) sdk\_version,

concat\_ws('|', collect\_set(gmail)) gmail,

concat\_ws('|', collect\_set(height\_width)) height\_width,

concat\_ws('|', collect\_set(app\_time)) app\_time,

concat\_ws('|', collect\_set(network)) network,

concat\_ws('|', collect\_set(lng)) lng,

concat\_ws('|', collect\_set(lat)) lat

from "$APP".dwd\_start\_log

where dt='$do\_date'

group by mid\_id;

insert overwrite table "$APP".dws\_uv\_detail\_wk partition(wk\_dt)

select

mid\_id,

concat\_ws('|', collect\_set(user\_id)) user\_id,

concat\_ws('|', collect\_set(version\_code)) version\_code,

concat\_ws('|', collect\_set(version\_name)) version\_name,

concat\_ws('|', collect\_set(lang)) lang,

concat\_ws('|', collect\_set(source)) source,

concat\_ws('|', collect\_set(os)) os,

concat\_ws('|', collect\_set(area)) area,

concat\_ws('|', collect\_set(model)) model,

concat\_ws('|', collect\_set(brand)) brand,

concat\_ws('|', collect\_set(sdk\_version)) sdk\_version,

concat\_ws('|', collect\_set(gmail)) gmail,

concat\_ws('|', collect\_set(height\_width)) height\_width,

concat\_ws('|', collect\_set(app\_time)) app\_time,

concat\_ws('|', collect\_set(network)) network,

concat\_ws('|', collect\_set(lng)) lng,

concat\_ws('|', collect\_set(lat)) lat,

date\_add(next\_day('$do\_date','MO'),-7),

date\_add(next\_day('$do\_date','SU'),-7),

concat(date\_add( next\_day('$do\_date','MO'),-7), '\_' , date\_add(next\_day('$do\_date','MO'),-1)

)

from "$APP".dws\_uv\_detail\_day

where dt>=date\_add(next\_day('$do\_date','MO'),-7) and dt<=date\_add(next\_day('$do\_date','MO'),-1)

group by mid\_id;

insert overwrite table "$APP".dws\_uv\_detail\_mn partition(mn)

select

mid\_id,

concat\_ws('|', collect\_set(user\_id)) user\_id,

concat\_ws('|', collect\_set(version\_code)) version\_code,

concat\_ws('|', collect\_set(version\_name)) version\_name,

concat\_ws('|', collect\_set(lang))lang,

concat\_ws('|', collect\_set(source)) source,

concat\_ws('|', collect\_set(os)) os,

concat\_ws('|', collect\_set(area)) area,

concat\_ws('|', collect\_set(model)) model,

concat\_ws('|', collect\_set(brand)) brand,

concat\_ws('|', collect\_set(sdk\_version)) sdk\_version,

concat\_ws('|', collect\_set(gmail)) gmail,

concat\_ws('|', collect\_set(height\_width)) height\_width,

concat\_ws('|', collect\_set(app\_time)) app\_time,

concat\_ws('|', collect\_set(network)) network,

concat\_ws('|', collect\_set(lng)) lng,

concat\_ws('|', collect\_set(lat)) lat,

date\_format('$do\_date','yyyy-MM')

from "$APP".dws\_uv\_detail\_day

where date\_format(dt,'yyyy-MM') = date\_format('$do\_date','yyyy-MM')

group by mid\_id;

"

$hive -e "$sql"

2）增加脚本执行权限

[byf@hadoop102 bin]$ chmod 777 dws.sh

3）脚本使用

[byf@hadoop102 module]$ dws.sh 2019-02-11

4）查询结果

hive (gmall)> select count(\*) from dws\_uv\_detail\_day;

hive (gmall)> select count(\*) from dws\_uv\_detail\_wk;

hive (gmall)> select count(\*) from dws\_uv\_detail\_mn ;

5）脚本执行时间

企业开发中一般在每日凌晨30分~1点

## 5.2 ADS层

目标：当日、当周、当月活跃设备数

### 5.2.1 活跃设备数



1）建表语句

hive (gmall)>

drop table if exists ads\_uv\_count;

create external table ads\_uv\_count(

    `dt` string COMMENT '统计日期',

    `day\_count` bigint COMMENT '当日用户数量',

    `wk\_count` bigint COMMENT '当周用户数量',

    `mn\_count` bigint COMMENT '当月用户数量',

    `is\_weekend` string COMMENT 'Y,N是否是周末,用于得到本周最终结果',

    `is\_monthend` string COMMENT 'Y,N是否是月末,用于得到本月最终结果'

) COMMENT '每日活跃用户数量'

stored as parquet

location '/warehouse/gmall/ads/ads\_uv\_count\_day/'

;

2）导入数据

hive (gmall)>

insert overwrite table ads\_uv\_count

select

'2019-02-10' dt,

daycount.ct,

wkcount.ct,

mncount.ct,

if(date\_add(next\_day('2019-02-10','MO'),-1)='2019-02-10','Y','N') ,

if(last\_day('2019-02-10')='2019-02-10','Y','N')

from

(

select

'2019-02-10' dt,

count(\*) ct

from dws\_uv\_detail\_day

where dt='2019-02-10'

)daycount join

(

select

'2019-02-10' dt,

count (\*) ct

from dws\_uv\_detail\_wk

where wk\_dt=concat(date\_add(next\_day('2019-02-10','MO'),-7),'\_' ,date\_add(next\_day('2019-02-10','MO'),-1) )

) wkcount on daycount.dt=wkcount.dt

join

(

select

'2019-02-10' dt,

count (\*) ct

from dws\_uv\_detail\_mn

where mn=date\_format('2019-02-10','yyyy-MM')

)mncount on daycount.dt=mncount.dt

;

3）查询导入结果

hive (gmall)> select \* from ads\_uv\_count ;

### 5.2.2 ADS层加载数据脚本

1）在hadoop102的/home/byf/bin目录下创建脚本

[byf@hadoop102 bin]$ vim ads.sh

在脚本中编写如下内容

#!/bin/bash

# 定义变量方便修改

APP=gmall

hive=/opt/module/hive/bin/hive

# 如果是输入的日期按照取输入日期；如果没输入日期取当前时间的前一天

if [ -n "$1" ] ;then

do\_date=$1

else

do\_date=`date -d "-1 day" +%F`

fi

sql="

set hive.exec.dynamic.partition.mode=nonstrict;

insert into table "$APP".ads\_uv\_count

select

'$do\_date' dt,

daycount.ct,

wkcount.ct,

mncount.ct,

if(date\_add(next\_day('$do\_date','MO'),-1)='$do\_date','Y','N') ,

if(last\_day('$do\_date')='$do\_date','Y','N')

from

(

select

'$do\_date' dt,

count(\*) ct

from "$APP".dws\_uv\_detail\_day

where dt='$do\_date'

)daycount join

(

select

'$do\_date' dt,

count (\*) ct

from "$APP".dws\_uv\_detail\_wk

where wk\_dt=concat(date\_add(next\_day('$do\_date','MO'),-7),'\_' ,date\_add(next\_day('$do\_date','MO'),-1) )

) wkcount on daycount.dt=wkcount.dt

join

(

select

'$do\_date' dt,

count (\*) ct

from "$APP".dws\_uv\_detail\_mn

where mn=date\_format('$do\_date','yyyy-MM')

)mncount on daycount.dt=mncount.dt;

"

$hive -e "$sql"

2）增加脚本执行权限

[byf@hadoop102 bin]$ chmod 777 ads.sh

3）脚本使用

[byf@hadoop102 module]$ ads.sh 2019-02-11

4）脚本执行时间

企业开发中一般在每日凌晨30分~1点

5）查询导入结果

hive (gmall)> select \* from ads\_uv\_count ;

# 第6章 需求二：用户新增主题

首次联网使用应用的用户。如果一个用户首次打开某APP，那这个用户定义为新增用户；卸载再安装的设备，不会被算作一次新增。新增用户包括日新增用户、周新增用户、月新增用户。

## 6.1 DWS层（每日新增设备明细表）





1）建表语句

hive (gmall)>

drop table if exists dws\_new\_mid\_day;

create table dws\_new\_mid\_day

(

`mid\_id` string COMMENT '设备唯一标识',

`user\_id` string COMMENT '用户标识',

`version\_code` string COMMENT '程序版本号',

`version\_name` string COMMENT '程序版本名',

`lang` string COMMENT '系统语言',

`source` string COMMENT '渠道号',

`os` string COMMENT '安卓系统版本',

`area` string COMMENT '区域',

`model` string COMMENT '手机型号',

`brand` string COMMENT '手机品牌',

`sdk\_version` string COMMENT 'sdkVersion',

`gmail` string COMMENT 'gmail',

`height\_width` string COMMENT '屏幕宽高',

`app\_time` string COMMENT '客户端日志产生时的时间',

`network` string COMMENT '网络模式',

`lng` string COMMENT '经度',

`lat` string COMMENT '纬度',

    `create\_date`  string comment '创建时间'

) COMMENT '每日新增设备信息'

stored as parquet

location '/warehouse/gmall/dws/dws\_new\_mid\_day/';

2）导入数据

用每日活跃用户表 left join 每日新增设备表，关联的条件是mid\_id相等。如果是每日新增的设备，则在每日新增设备表中为null。

hive (gmall)>

insert into table dws\_new\_mid\_day

select

ud.mid\_id,

ud.user\_id ,

ud.version\_code ,

ud.version\_name ,

ud.lang ,

ud.source,

ud.os,

ud.area,

ud.model,

ud.brand,

ud.sdk\_version,

ud.gmail,

ud.height\_width,

ud.app\_time,

ud.network,

ud.lng,

ud.lat,

'2019-02-10'

from dws\_uv\_detail\_day ud left join dws\_new\_mid\_day nm on ud.mid\_id=nm.mid\_id

where ud.dt='2019-02-10' and nm.mid\_id is null;

3）查询导入数据

hive (gmall)> select count(\*) from dws\_new\_mid\_day ;

## 6.2 ADS层（每日新增设备表）



1）建表语句

hive (gmall)>

drop table if exists `ads\_new\_mid\_count`;

create table `ads\_new\_mid\_count`

(

    `create\_date`     string comment '创建时间' ,

    `new\_mid\_count`   BIGINT comment '新增设备数量'

) COMMENT '每日新增设备信息数量'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_new\_mid\_count/';

2）导入数据

hive (gmall)>

insert into table ads\_new\_mid\_count

select create\_date , count(\*) from dws\_new\_mid\_day

where create\_date='2019-02-10'

group by create\_date ;

3）查询导入数据

hive (gmall)> select \* from ads\_new\_mid\_count;

# 第7章 需求三：用户留存主题

## 7.1 需求目标

### 7.1.1 用户留存概念



### 7.1.2 需求描述



## 7.2 DWS层

### 7.3.1 DWS层（每日留存用户明细表）



1）建表语句

hive (gmall)>

drop table if exists `dws\_user\_retention\_day`;

create table `dws\_user\_retention\_day`

(

`mid\_id` string COMMENT '设备唯一标识',

`user\_id` string COMMENT '用户标识',

`version\_code` string COMMENT '程序版本号',

`version\_name` string COMMENT '程序版本名',

`lang` string COMMENT '系统语言',

`source` string COMMENT '渠道号',

`os` string COMMENT '安卓系统版本',

`area` string COMMENT '区域',

`model` string COMMENT '手机型号',

`brand` string COMMENT '手机品牌',

`sdk\_version` string COMMENT 'sdkVersion',

`gmail` string COMMENT 'gmail',

`height\_width` string COMMENT '屏幕宽高',

`app\_time` string COMMENT '客户端日志产生时的时间',

`network` string COMMENT '网络模式',

`lng` string COMMENT '经度',

`lat` string COMMENT '纬度',

   `create\_date`       string comment '设备新增时间',

   `retention\_day`     int comment '截止当前日期留存天数'

) COMMENT '每日用户留存情况'

PARTITIONED BY ( `dt` string)

stored as parquet

location '/warehouse/gmall/dws/dws\_user\_retention\_day/'

;

2）导入数据（每天计算前1天的新用户访问留存明细）

hive (gmall)>

insert overwrite table dws\_user\_retention\_day partition(dt="2019-02-11")

select

nm.mid\_id,

nm.user\_id ,

nm.version\_code ,

nm.version\_name ,

nm.lang ,

nm.source,

nm.os,

nm.area,

nm.model,

nm.brand,

nm.sdk\_version,

nm.gmail,

nm.height\_width,

nm.app\_time,

nm.network,

nm.lng,

nm.lat,

nm.create\_date,

1 retention\_day

from dws\_uv\_detail\_day ud join dws\_new\_mid\_day nm   on ud.mid\_id =nm.mid\_id

where ud.dt='2019-02-11' and nm.create\_date=date\_add('2019-02-11',-1);

3）查询导入数据（每天计算前1天的新用户访问留存明细）

hive (gmall)> select count(\*) from dws\_user\_retention\_day;

### 7.3.2 DWS层（1,2,3,n天留存用户明细表）

1）导入数据（每天计算前1,2,3，n天的新用户访问留存明细）

hive (gmall)>

insert overwrite table dws\_user\_retention\_day partition(dt="2019-02-11")

select

nm.mid\_id,

nm.user\_id ,

nm.version\_code ,

nm.version\_name ,

nm.lang ,

nm.source,

nm.os,

nm.area,

nm.model,

nm.brand,

nm.sdk\_version,

nm.gmail,

nm.height\_width,

nm.app\_time,

nm.network,

nm.lng,

nm.lat,

nm.create\_date,

1 retention\_day

from dws\_uv\_detail\_day ud join dws\_new\_mid\_day nm on ud.mid\_id =nm.mid\_id

where ud.dt='2019-02-11' and nm.create\_date=date\_add('2019-02-11',-1)

union all

select

nm.mid\_id,

nm.user\_id ,

nm.version\_code ,

nm.version\_name ,

nm.lang ,

nm.source,

nm.os,

nm.area,

nm.model,

nm.brand,

nm.sdk\_version,

nm.gmail,

nm.height\_width,

nm.app\_time,

nm.network,

nm.lng,

nm.lat,

nm.create\_date,

2 retention\_day

from dws\_uv\_detail\_day ud join dws\_new\_mid\_day nm on ud.mid\_id =nm.mid\_id

where ud.dt='2019-02-11' and nm.create\_date=date\_add('2019-02-11',-2)

union all

select

nm.mid\_id,

nm.user\_id ,

nm.version\_code ,

nm.version\_name ,

nm.lang ,

nm.source,

nm.os,

nm.area,

nm.model,

nm.brand,

nm.sdk\_version,

nm.gmail,

nm.height\_width,

nm.app\_time,

nm.network,

nm.lng,

nm.lat,

nm.create\_date,

3 retention\_day

from dws\_uv\_detail\_day ud join dws\_new\_mid\_day nm on ud.mid\_id =nm.mid\_id

where ud.dt='2019-02-11' and nm.create\_date=date\_add('2019-02-11',-3);

2）查询导入数据（每天计算前1,2,3天的新用户访问留存明细）

hive (gmall)> select retention\_day , count(\*) from dws\_user\_retention\_day group by retention\_day;

## 7.3 ADS层

### 7.3.1留存用户数



1）建表语句

hive (gmall)>

drop table if exists `ads\_user\_retention\_day\_count`;

create table `ads\_user\_retention\_day\_count`

(

   `create\_date`       string comment '设备新增日期',

   `retention\_day`     int comment '截止当前日期留存天数',

   `retention\_count` bigint comment '留存数量'

) COMMENT '每日用户留存情况'

stored as parquet

location '/warehouse/gmall/ads/ads\_user\_retention\_day\_count/';

2）导入数据

hive (gmall)>

insert into table ads\_user\_retention\_day\_count

select

create\_date,

retention\_day,

count(\*) retention\_count

from dws\_user\_retention\_day

where dt='2019-02-11'

group by create\_date,retention\_day;

3）查询导入数据

hive (gmall)> select \* from ads\_user\_retention\_day\_count;

### 7.3.2留存用户比率



1）建表语句

hive (gmall)>

drop table if exists `ads\_user\_retention\_day\_rate`;

create table `ads\_user\_retention\_day\_rate`

(

     `stat\_date` string comment '统计日期',

     `create\_date`       string comment '设备新增日期',

     `retention\_day`     int comment '截止当前日期留存天数',

     `retention\_count` bigint comment '留存数量',

     `new\_mid\_count`     string comment '当日设备新增数量',

     `retention\_ratio`   decimal(10,2) comment '留存率'

) COMMENT '每日用户留存情况'

stored as parquet

location '/warehouse/gmall/ads/ads\_user\_retention\_day\_rate/';

2）导入数据

hive (gmall)>

insert into table ads\_user\_retention\_day\_rate

select

'2019-02-11' ,

ur.create\_date,

ur.retention\_day,

ur.retention\_count ,

nc.new\_mid\_count,

ur.retention\_count/nc.new\_mid\_count\*100

from

(

select

create\_date,

retention\_day,

count(\*) retention\_count

from `dws\_user\_retention\_day`

where dt='2019-02-11'

group by create\_date,retention\_day

) ur join ads\_new\_mid\_count nc on nc.create\_date=ur.create\_date;

3）查询导入数据

hive (gmall)>select \* from ads\_user\_retention\_day\_rate;

# 第8章 总结

## 8.1 数仓分几层？每层做什么的？

1）ODS层（原始数据层）

存储原始数据，直接加载原始日志、数据，数据保持原貌不做处理。

2）DWD层（明细层）

对ODS层数据进行清洗（去除空值、脏数据，超过极限范围的数据）

3）DWS层（服务数据层）

以DWD层为基础，进行轻度汇总。比如：用户当日、设备当日、商品当日。

4）ADS层（数据应用层）

## 8.2 Tez引擎优点？

Tez可以将多个有依赖的作业转换为一个作业，这样只需写一次HDFS，且中间节点较少，从而大大提升作业的计算性能。

## 8.3 在项目中是否自定义过UDF、UDTF函数，以及用他们处理了什么问题？

自定义过。

用UDF函数解析公共字段；用UDTF函数解析事件字段。

## 8.4 如何分析用户活跃？

在启动日志中统计不同设备id 出现次数。

## 8.5 如何分析用户新增？

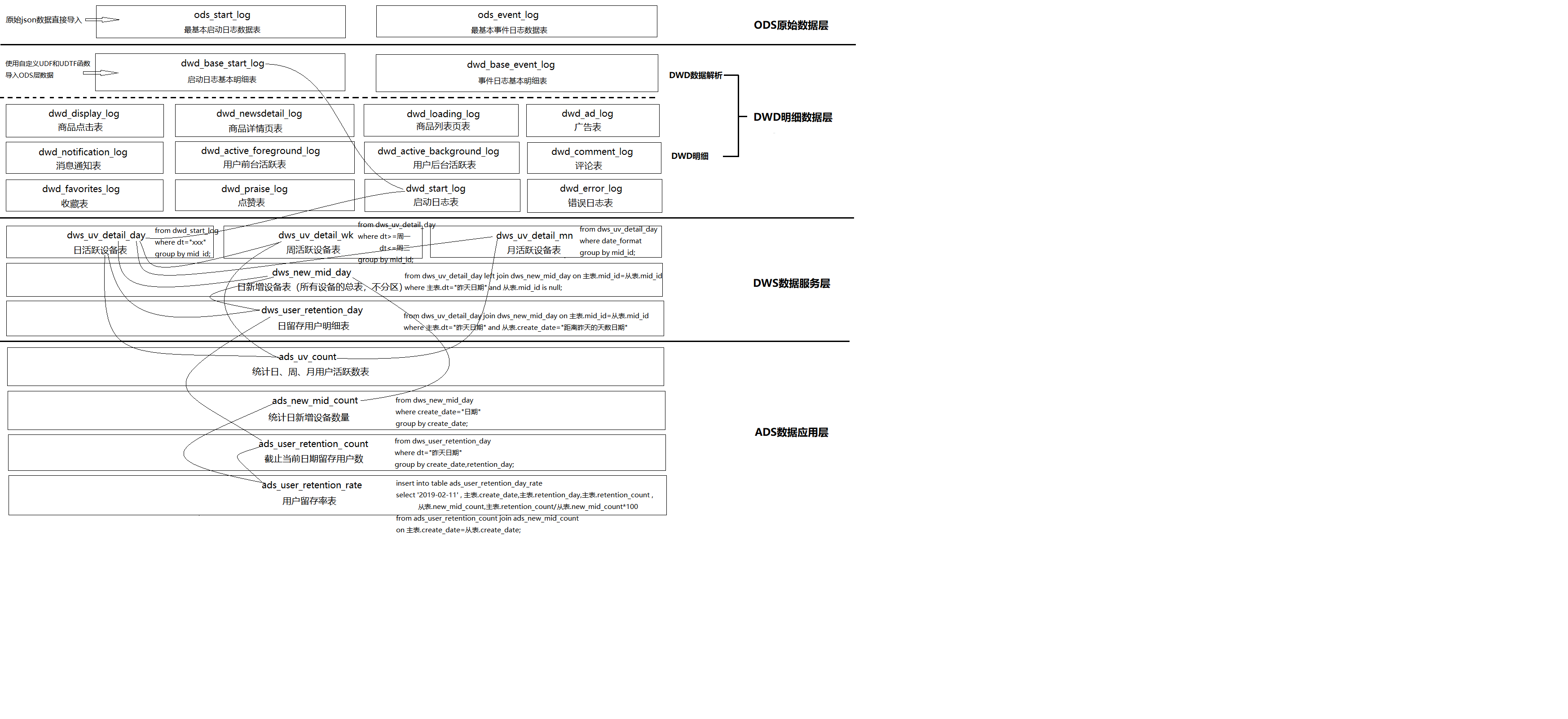
用活跃用户表 left join 用户新增表，用户新增表中mid为空的即为用户新增。

## 8.6 如何分析用户1天留存？

留存用户=前一天新增 join 今天活跃

用户留存率=留存用户/前一天新增

## 8.7 整个文档中涉及的所有层级及表



<http://app.liuchengtu.com/#Rb734cafbf09d4651caadd74e5a3d304d>

