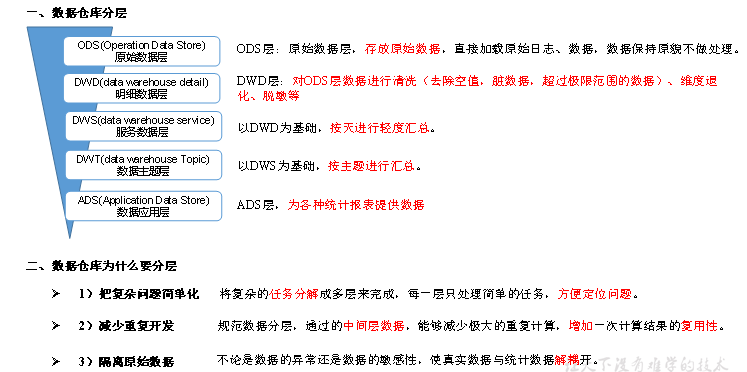
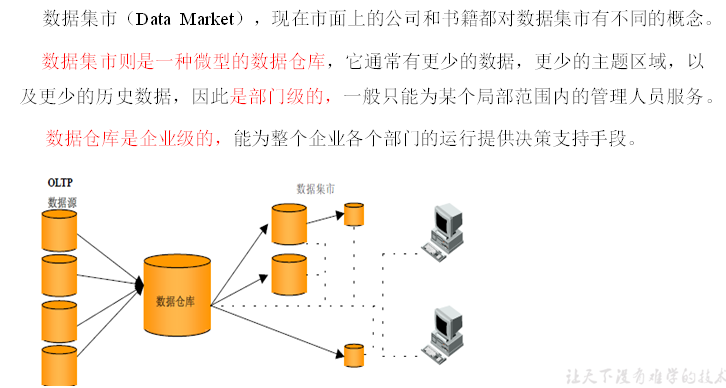
数据仓库系统

# 第1章 数仓分层

## 1.1 为什么要分层



## 1.2 数据集市与数据仓库概念



## 1.3 数仓命名规范

### 1.3.1 表命名

* ODS层命名为ods\_表名
* DWD层命名为dwd\_dim/fact\_表名
* DWS层命名为dws\_表名
* DWT层命名为dwt\_购物车
* ADS层命名为ads\_表名
* 临时表命名为xxx\_tmp
* 用户行为表，以log为后缀。

### 1.3.2 脚本命名

* 数据源\_to\_目标\_db/log.sh
* 用户行为脚本以log为后缀；业务数据脚本以db为后缀。

# 第2章 数仓理论

## 2.1 范式理论

### 2.1.1 范式概念

1）定义

范式可以理解为设计一张数据表的表结构，符合的标准级别。 规范和要求

2）优点

关系型数据库设计时，遵照一定的规范要求，目的在于降低数据的冗余性。

为什么要降低数据冗余性？

（1）十几年前，磁盘很贵，为了减少磁盘存储。

（2）以前没有分布式系统，都是单机，只能增加磁盘，磁盘个数也是有限的

（3）一次修改，需要修改多个表，很难保证数据一致性

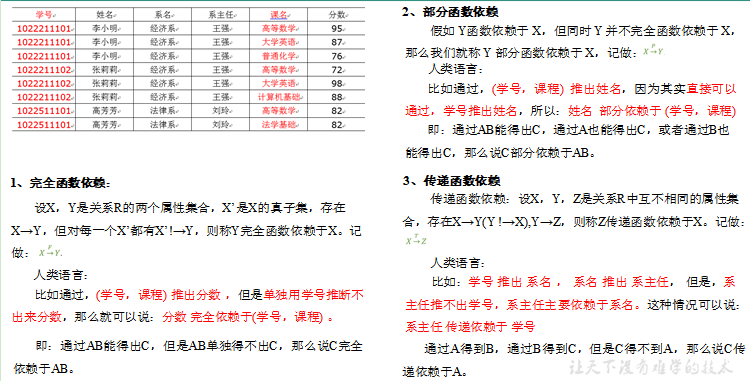
3）缺点

范式的缺点是获取数据时，需要通过Join拼接出最后的数据。

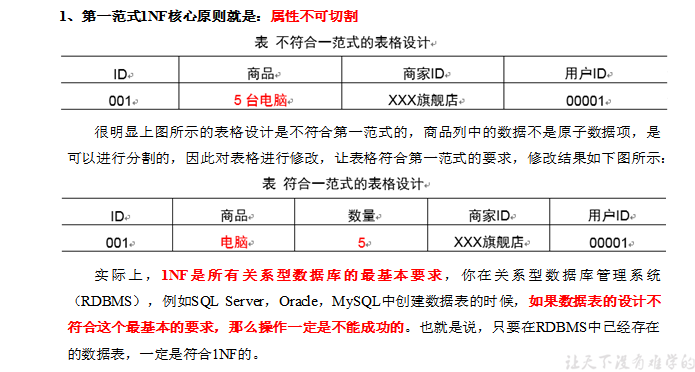
4）分类

目前业界范式有：第一范式(1NF)、第二范式(2NF)、第三范式(3NF)、巴斯-科德范式(BCNF)、第四范式(4NF)、第五范式(5NF)。

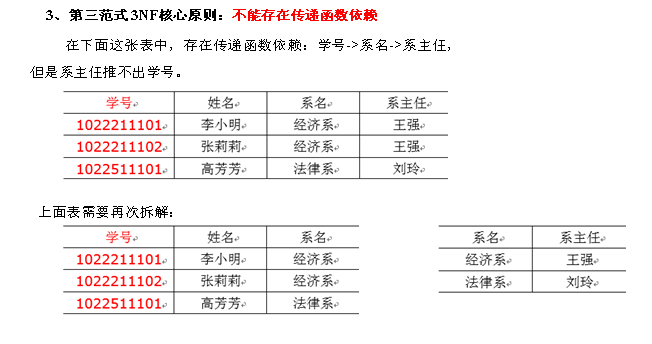
### 2.1.2 函数依赖



### 2.1.3 三范式区分





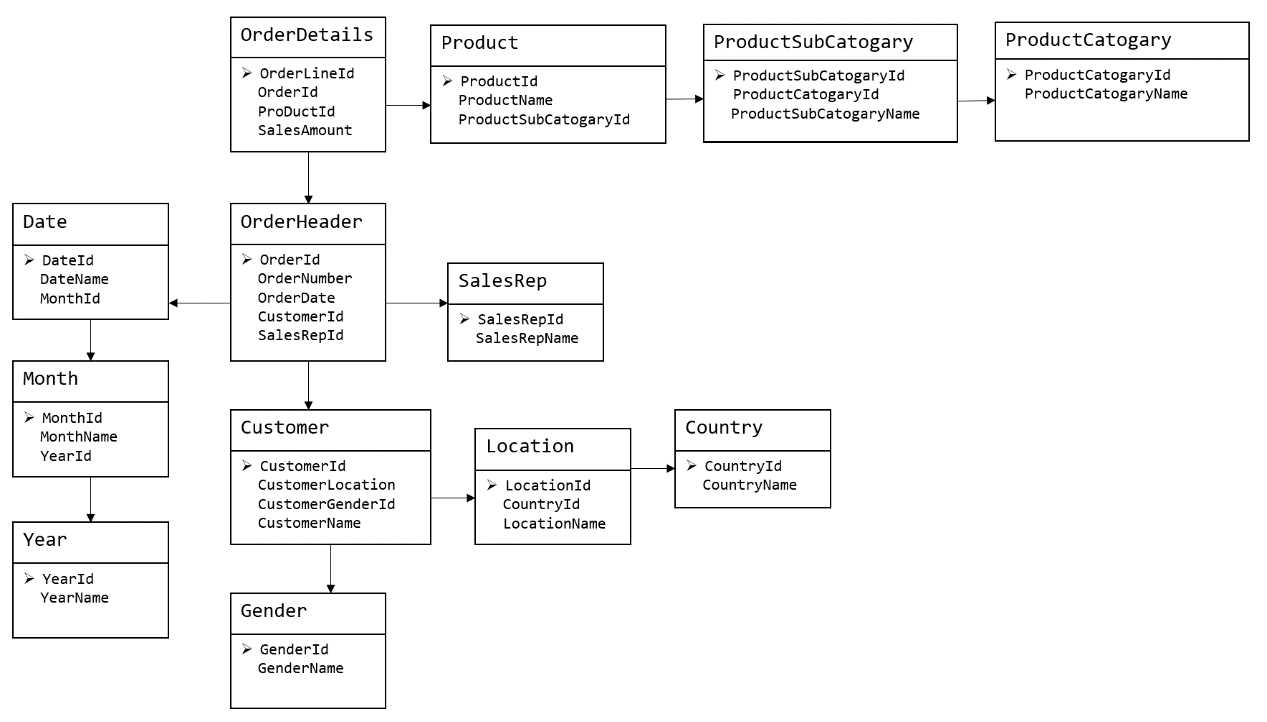


## 2.2 关系建模与维度建模

当今的数据处理大致可以分成两大类：联机事务处理OLTP（on-line transaction processing）、联机分析处理OLAP（On-Line Analytical Processing）。OLTP是传统的关系型数据库的主要应用，主要是基本的、日常的事务处理，例如银行交易。OLAP是数据仓库系统的主要应用，支持复杂的分析操作，侧重决策支持，并且提供直观易懂的查询结果。二者的主要区别对比如下表所示。

|  |  |  |
| --- | --- | --- |
| **对比属性** | **OLTP** | **OLAP** |
| **读特性** | 每次查询只返回少量记录 | 对大量记录进行汇总 |
| **写特性** | 随机、低延时写入用户的输入 | 批量导入 |
| **使用场景** | 用户，Java EE项目 | 内部分析师，为决策提供支持 |
| **数据表征** | 最新数据状态 | 随时间变化的历史状态 |
| **数据规模** | GB | TB到PB |

### 2.2.1 关系建模



关系模型如图所示，严格遵循第三范式（3NF），从图中可以看出，较为松散、零碎，物理表数量多，而数据冗余程度低。由于数据分布于众多的表中，这些数据可以更为灵活地被应用，功能性较强。关系模型主要应用与OLTP系统中，为了保证数据的一致性以及避免冗余，所以大部分业务系统的表都是遵循第三范式的。

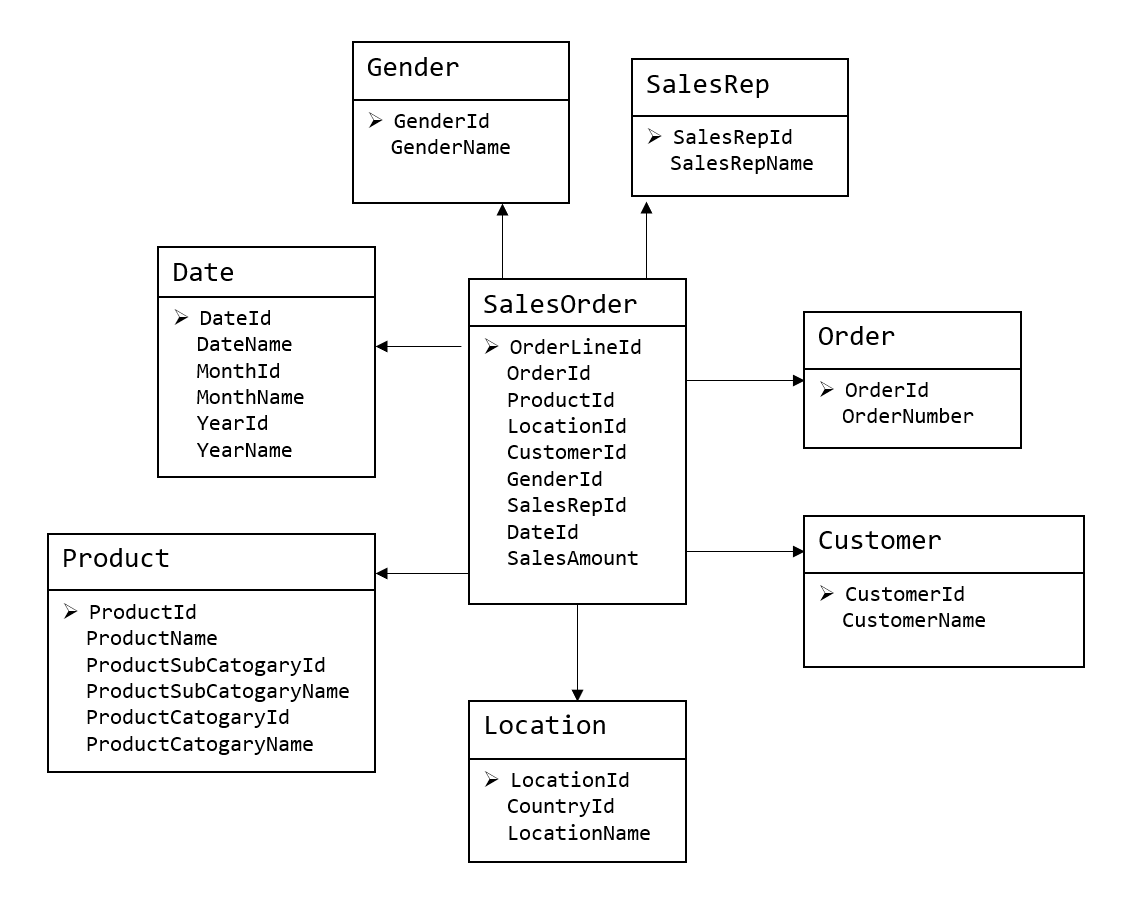


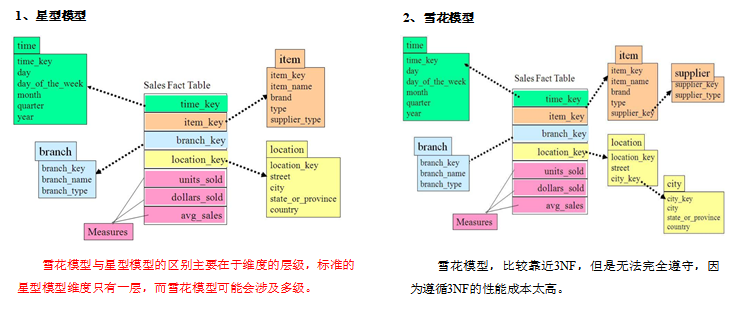
图 维度模型示意图

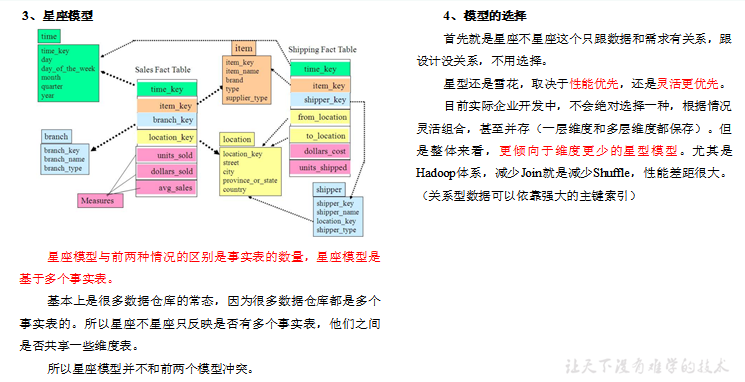
维度模型如图所示，主要应用于OLAP系统中，通常以某一个事实表为中心进行表的组织，主要面向业务，特征是可能存在数据的冗余，但是能方便的得到数据。

关系模型虽然冗余少，但是在大规模数据，跨表分析统计查询过程中，会造成多表关联，这会大大降低执行效率。所以通常我们采用维度模型建模，把相关各种表整理成两种：事实表和维度表两种。

### 2.2.2 维度建模

在维度建模的基础上又分为三种模型：星型模型、雪花模型、星座模型。





## 2.3 维度表和事实表（重点）

### 2.3.1 维度表

**维度表**：一般是对事实的**描述信息**。每一张维表对应现实世界中的一个对象或者概念。 例如：用户、商品、日期、地区等。

**维表的特征：**

* 维表的范围很宽（具有多个属性、列比较多）
* 跟事实表相比，行数相对较小：通常< 10万条
* 内容相对固定：编码表

时间维度表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 日期ID | day of week | day of year | 季度 | 节假日 |
| 2020-01-01 | 2 | 1 | 1 | 元旦 |
| 2020-01-02 | 3 | 2 | 1 | 无 |
| 2020-01-03 | 4 | 3 | 1 | 无 |
| 2020-01-04 | 5 | 4 | 1 | 无 |
| 2020-01-05 | 6 | 5 | 1 | 无 |

### 2.3.2 事实表

**事实表中的每行数据代表一个业务事件（下单、支付、退款、评价等）**。“事实”这个术语表示的是业务事件的**度量值（可统计次数、个数、金额等）**，例如，订单事件中的下单金额。

每一个事实表的行包括：具有可加性的数值型的度量值、与维表相连接的外键、通常具有两个和两个以上的外键、外键之间表示维表之间多对多的关系。

事实表的特征：

* 非常的大
* 内容相对的窄：列数较少
* 经常发生变化，每天会新增加很多。

**1）事务型事实表**

以**每个事务或事件为单位**，例如一个销售订单记录，一笔支付记录等，作为事实表里的一行数据。一旦事务被提交，事实表数据被插入，数据就不再进行更改，其更新方式为增量更新。

**2）周期型快照事实表**

周期型快照事实表中**不会保留所有数据**，**只保留固定时间间隔的数据**，例如每天或者每月的销售额，或每月的账户余额等。

**3）累积型快照事实表**

**累计快照事实表用于跟踪业务事实的变化。**例如，数据仓库中可能需要累积或者存储订单从下订单开始，到订单商品被打包、运输、和签收的各个业务阶段的时间点数据来跟踪订单声明周期的进展情况。当这个业务过程进行时，事实表的记录也要不断更新。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **订单id** | **用户id** | **下单时间** | **打包时间** | **发货时间** | **签收时间** | **订单金额** |
|  |  | 3-8 | 3-8 | 3-9 | 3-10 |  |

## 2.4 数据仓库建模（绝对重点）

### 2.4.1 ODS层

（1）保持数据原貌不做任何修改，起到备份数据的作用。

（2）数据采用压缩，减少磁盘存储空间（例如：原始数据100G，可以压缩到10G左右）

（3）创建分区表，防止后续的全表扫描

### 2.4.2 DWD层

DWD层需构建维度模型，一般采用星型模型，呈现的状态一般为星座模型。

维度建模一般按照以下四个步骤：

**选择业务过程→声明粒度→确认维度→确认事实**

**（1）选择业务过程**

在业务系统中，挑选我们感兴趣的业务线，比如下单业务，支付业务，退款业务，物流业务，一条业务线对应一张事实表。

**（2）声明粒度**

数据粒度指数据仓库的数据中保存数据的细化程度或综合程度的级别。

声明粒度意味着精确定义事实表中的一行数据表示什么，应该尽可能选择**最小粒度**，以此来应各种各样的需求。

**典型的粒度声明如下：**

订单中，每个商品项作为下单事实表中的一行，粒度为每次下单

每周的订单次数作为一行，粒度就是每周下单。

每月的订单次数作为一行，粒度就是每月下单

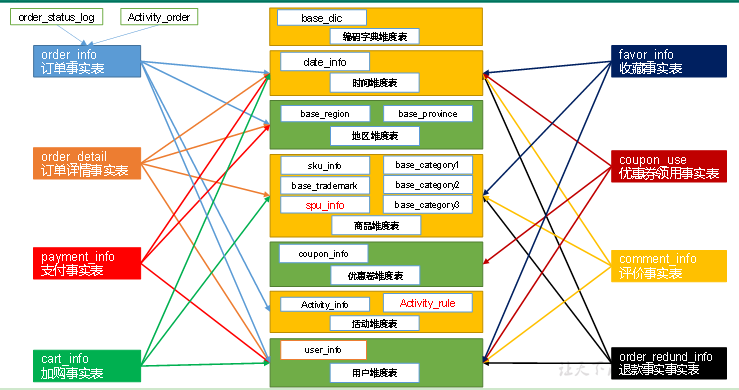
**（3）确定维度**

维度的主要作用是描述业务是事实，主要表示的是“谁，何处，何时”等信息。

**（4）确定事实**

此处的“事实”一词，指的是业务中的度量值，例如订单金额、下单次数等。

在DWD层，以**业务过程**为建模驱动，基于每个具体业务过程的特点，构建**最细粒度**的明细层事实表。事实表可做适当的宽表化处理。



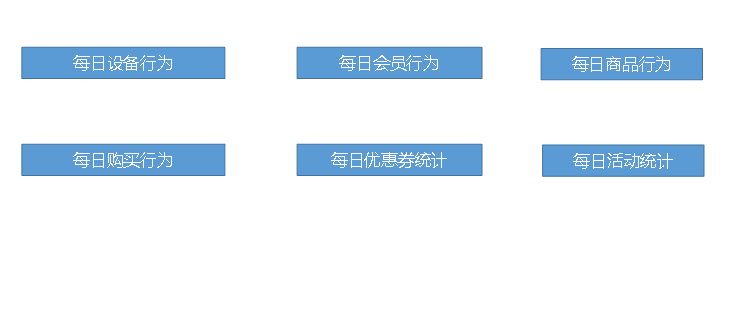
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **时间** | **用户** | **地区** | **商品** | **优惠券** | **活动** | **编码** | **度量值** |
| **订单** | √ | √ | √ |  |  | √ |  | 件数/金额 |
| **订单详情** | √ |  | √ | √ |  |  |  | 件数/金额 |
| **支付** | √ |  | √ |  |  |  |  | 金额 |
| **加购** | √ | √ |  | √ |  |  |  | 件数/金额 |
| **收藏** | √ | √ |  | √ |  |  |  | 个数 |
| **评价** | √ | √ |  | √ |  |  |  | 个数 |
| **退款** | √ | √ |  | √ |  |  |  | 件数/金额 |
| **优惠券领用** | √ | √ |  |  | √ |  |  | 个数 |

至此，数仓的维度建模已经完毕，DWS、DWT和ADS和维度建模已经没有关系了。

DWS和DWT都是建宽表，宽表都是按照主题去建。主题相当于观察问题的角度。对应着维度表。

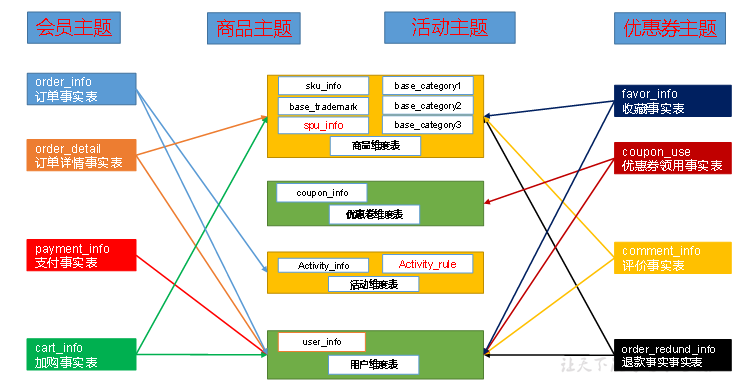
### 2.4.3 DWS层

统计各个主题对象的当天行为，服务于DWT层的主题宽表，以及一些业务明细数据，应对特殊需求（例如，购买行为，统计商品复购率）。



### 2.4.4 DWT层

以分析的**主题对象**为建模驱动，基于上层的应用和产品的指标需求，构建主题对象的全量宽表。



### 2.4.5 ADS层

对电商系统各大主题指标分别进行分析。

# 第3章 数仓搭建-ODS层

1）保持数据原貌不做任何修改，起到备份数据的作用。

2）数据采用LZO压缩，减少磁盘存储空间。100G数据可以压缩到10G以内。

3）创建分区表，防止后续的全表扫描，在企业开发中大量使用分区表。

4）创建外部表。在企业开发中，除了自己用的临时表，创建内部表外，绝大多数场景都是创建外部表。

## 3.1 创建数据库

1）启动hive

[woaini@hadoop102 hive]$ nohup bin/hive --service metastore &

[woaini@hadoop102 hive]$ nohup bin/hive --service hiveserver2 &

[woaini@hadoop102 hive]$ bin/hive

2）显示数据库

hive (default)> show databases;

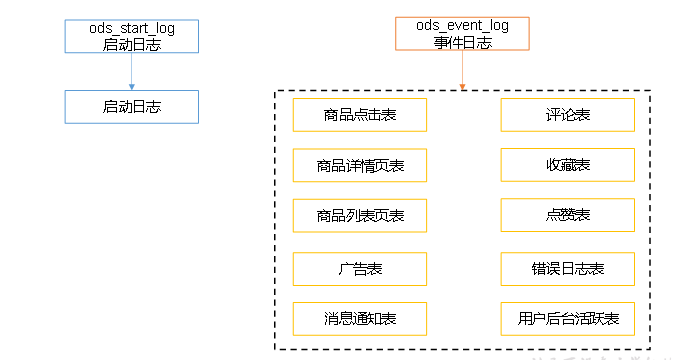
3）创建数据库

hive (default)> create database gmall;

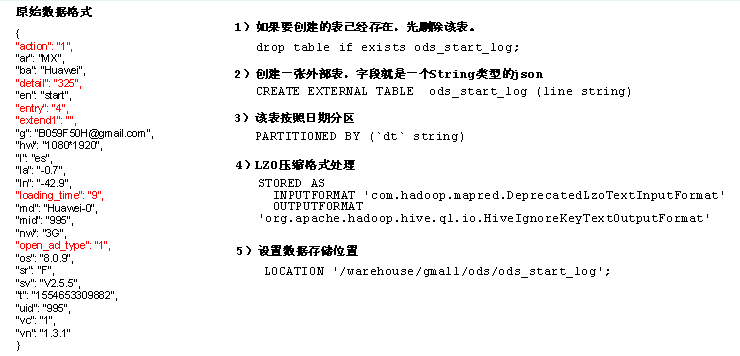
4）使用数据库

hive (default)> use gmall;

## 3.2 ODS层（用户行为数据）



### 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/2020-03-10' into table gmall.ods\_start\_log partition(dt='2020-03-10');

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)>

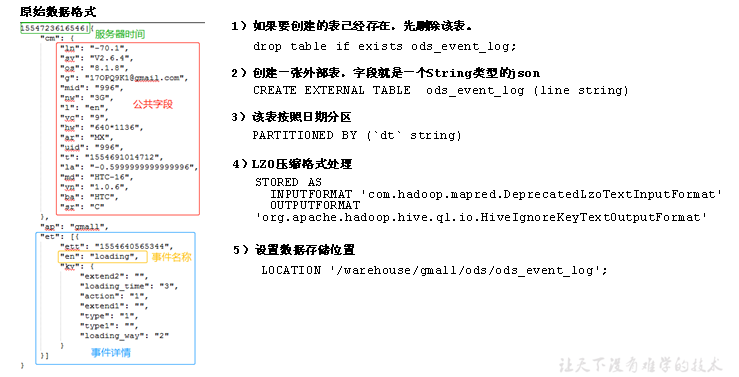
select \* from ods\_start\_log where dt='2020-03-10' limit 2;

4）为lzo压缩文件创建索引

[woaini@hadoop102 hadoop-2.7.2]$

hadoop jar /opt/module/hadoop-2.7.2/share/hadoop/common/hadoop-lzo-0.4.20.jar com.hadoop.compression.lzo.DistributedLzoIndexer /warehouse/gmall/ods/ods\_start\_log/dt=2020-03-10

### 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/2020-03-10' into table gmall.ods\_event\_log partition(dt='2020-03-10');

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 where dt="2020-03-10" limit 2;

4）为lzo压缩文件创建索引

[woaini@hadoop102 hadoop-2.7.2]$

hadoop jar /opt/module/hadoop-2.7.2/share/hadoop/common/hadoop-lzo-0.4.20.jar com.hadoop.compression.lzo.DistributedLzoIndexer /warehouse/gmall/ods/ods\_event\_log/dt=2020-03-10

### 3.2.3 Shell中单引号和双引号区别

1）在/home/woaini/bin创建一个test.sh文件

[woaini@hadoop102 bin]$ vim test.sh

在文件中添加如下内容

#!/bin/bash

do\_date=$1

echo '$do\_date'

echo "$do\_date"

echo "'$do\_date'"

echo '"$do\_date"'

echo `date`

2）查看执行结果

[woaini@hadoop102 bin]$ test.sh 2020-03-10

$do\_date

2020-03-10

'2020-03-10'

"$do\_date"

2020年 05月 02日 星期四 21:02:08 CST

3）总结：

（1）单引号不取变量值

（2）双引号取变量值

（3）反引号`，执行引号中命令

（4）双引号内部嵌套单引号，取出变量值

（5）单引号内部嵌套双引号，不取出变量值

### 3.2.4 ODS层加载数据脚本

1）在hadoop102的/home/woaini/bin目录下创建脚本

[woaini@hadoop102 bin]$ vim hdfs\_to\_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' overwrite into table ${APP}.ods\_start\_log partition(dt='$do\_date');

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

"

$hive -e "$sql"

hadoop jar /opt/module/hadoop-2.7.2/share/hadoop/common/hadoop-lzo-0.4.20.jar com.hadoop.compression.lzo.DistributedLzoIndexer /warehouse/gmall/ods/ods\_start\_log/dt=$do\_date

hadoop jar /opt/module/hadoop-2.7.2/share/hadoop/common/hadoop-lzo-0.4.20.jar com.hadoop.compression.lzo.DistributedLzoIndexer /warehouse/gmall/ods/ods\_event\_log/dt=$do\_date

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，[woaini@hadoop102 ~]$ date --help

2）增加脚本执行权限

[woaini@hadoop102 bin]$ chmod 777 hdfs\_to\_ods\_log.sh

3）脚本使用

[woaini@hadoop102 module]$ hdfs\_to\_ods\_log.sh 2020-03-11

4）查看导入数据

hive (gmall)>

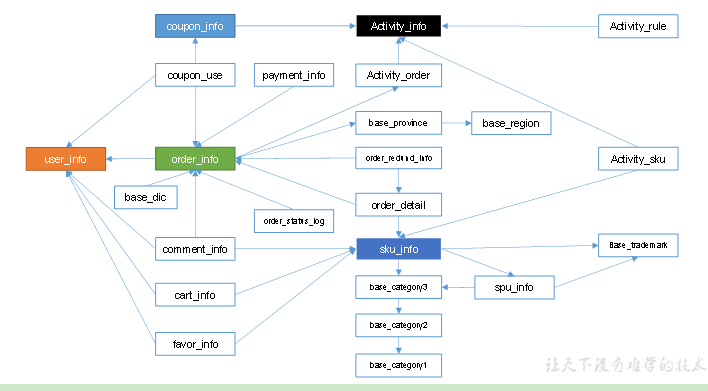
select \* from ods\_start\_log where dt='2020-03-11' limit 2;

select \* from ods\_event\_log where dt='2020-03-11' 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 ODS层（业务数据）



### 3.3.1 订单表（增量及更新）

hive (gmall)>

drop table if exists ods\_order\_info;

create external table ods\_order\_info (

`id` string COMMENT '订单号',

`final\_total\_amount` decimal(10,2) COMMENT '订单金额',

`order\_status` string COMMENT '订单状态',

`user\_id` string COMMENT '用户id',

`out\_trade\_no` string COMMENT '支付流水号',

`create\_time` string COMMENT '创建时间',

`operate\_time` string COMMENT '操作时间',

`province\_id` string COMMENT '省份ID',

`benefit\_reduce\_amount` decimal(10,2) COMMENT '优惠金额',

`original\_total\_amount` decimal(10,2) COMMENT '原价金额',

`feight\_fee` decimal(10,2) COMMENT '运费'

) COMMENT '订单表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_order\_info/';

### 3.3.2 订单详情表（增量）

hive (gmall)>

drop table if exists ods\_order\_detail;

create external table ods\_order\_detail(

`id` string COMMENT '订单编号',

`order\_id` string COMMENT '订单号',

`user\_id` string COMMENT '用户id',

`sku\_id` string COMMENT '商品id',

`sku\_name` string COMMENT '商品名称',

`order\_price` decimal(10,2) COMMENT '商品价格',

`sku\_num` bigint COMMENT '商品数量',

`create\_time` string COMMENT '创建时间'

) COMMENT '订单详情表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_order\_detail/';

### 3.3.3 SKU商品表（全量）

hive (gmall)>

drop table if exists ods\_sku\_info;

create external table ods\_sku\_info(

`id` string COMMENT 'skuId',

`spu\_id` string COMMENT 'spuid',

`price` decimal(10,2) COMMENT '价格',

`sku\_name` string COMMENT '商品名称',

`sku\_desc` string COMMENT '商品描述',

`weight` string COMMENT '重量',

`tm\_id` string COMMENT '品牌id',

`category3\_id` string COMMENT '品类id',

`create\_time` string COMMENT '创建时间'

) COMMENT 'SKU商品表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_sku\_info/';

### 3.3.4 用户表（增量及更新）

hive (gmall)>

drop table if exists ods\_user\_info;

create external table ods\_user\_info(

`id` string COMMENT '用户id',

`name` string COMMENT '姓名',

`birthday` string COMMENT '生日',

`gender` string COMMENT '性别',

`email` string COMMENT '邮箱',

`user\_level` string COMMENT '用户等级',

`create\_time` string COMMENT '创建时间',

`operate\_time` string COMMENT '操作时间'

) COMMENT '用户表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_user\_info/';

### 3.3.5 商品一级分类表（全量）

hive (gmall)>

drop table if exists ods\_base\_category1;

create external table ods\_base\_category1(

`id` string COMMENT 'id',

`name` string COMMENT '名称'

) COMMENT '商品一级分类表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_base\_category1/';

### 3.3.6 商品二级分类表（全量）

hive (gmall)>

drop table if exists ods\_base\_category2;

create external table ods\_base\_category2(

`id` string COMMENT ' id',

`name` string COMMENT '名称',

category1\_id string COMMENT '一级品类id'

) COMMENT '商品二级分类表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_base\_category2/';

### 3.3.7 商品三级分类表（全量）

hive (gmall)>

drop table if exists ods\_base\_category3;

create external table ods\_base\_category3(

`id` string COMMENT ' id',

`name` string COMMENT '名称',

category2\_id string COMMENT '二级品类id'

) COMMENT '商品三级分类表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_base\_category3/';

### 3.3.8 支付流水表（增量）

hive (gmall)>

drop table if exists ods\_payment\_info;

create external table ods\_payment\_info(

`id` bigint COMMENT '编号',

`out\_trade\_no` string COMMENT '对外业务编号',

`order\_id` string COMMENT '订单编号',

`user\_id` string COMMENT '用户编号',

`alipay\_trade\_no` string COMMENT '支付宝交易流水编号',

`total\_amount` decimal(16,2) COMMENT '支付金额',

`subject` string COMMENT '交易内容',

`payment\_type` string COMMENT '支付类型',

`payment\_time` string COMMENT '支付时间'

) COMMENT '支付流水表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_payment\_info/';

### 3.3.9 省份表（特殊）

hive (gmall)>

drop table if exists ods\_base\_province;

create external table ods\_base\_province (

`id` bigint COMMENT '编号',

`name` string COMMENT '省份名称',

`region\_id` string COMMENT '地区ID',

`area\_code` string COMMENT '地区编码',

`iso\_code` string COMMENT 'iso编码'

) COMMENT '省份表'

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_base\_province/';

### 3.3.10 地区表（特殊）

hive (gmall)>

drop table if exists ods\_base\_region;

create external table ods\_base\_region (

`id` bigint COMMENT '编号',

`region\_name` string COMMENT '地区名称'

) COMMENT '地区表'

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_base\_region/';

### 3.3.11 品牌表（全量）

hive (gmall)>

drop table if exists ods\_base\_trademark;

create external table ods\_base\_trademark (

`tm\_id` bigint COMMENT '编号',

`tm\_name` string COMMENT '品牌名称'

) COMMENT '品牌表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_base\_trademark/';

### 3.3.12 订单状态表（增量）

hive (gmall)>

drop table if exists ods\_order\_status\_log;

create external table ods\_order\_status\_log (

`id` bigint COMMENT '编号',

`order\_id` string COMMENT '订单ID',

`order\_status` string COMMENT '订单状态',

`operate\_time` string COMMENT '修改时间'

) COMMENT '订单状态表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_order\_status\_log/';

### 3.3.13 SPU商品表（全量）

hive (gmall)>

drop table if exists ods\_spu\_info;

create external table ods\_spu\_info(

`id` string COMMENT 'spuid',

`spu\_name` string COMMENT 'spu名称',

`category3\_id` string COMMENT '品类id',

`tm\_id` string COMMENT '品牌id'

) COMMENT 'SPU商品表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_spu\_info/';

### 3.3.14 商品评论表（增量）

hive (gmall)>

drop table if exists ods\_comment\_info;

create external table ods\_comment\_info(

`id` string COMMENT '编号',

`user\_id` string COMMENT '用户ID',

`sku\_id` string COMMENT '商品sku',

`spu\_id` string COMMENT '商品spu',

`order\_id` string COMMENT '订单ID',

`appraise` string COMMENT '评价',

`create\_time` string COMMENT '评价时间'

) COMMENT '商品评论表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_comment\_info/';

### 3.3.15 退单表（增量）

hive (gmall)>

drop table if exists ods\_order\_refund\_info;

create external table ods\_order\_refund\_info(

`id` string COMMENT '编号',

`user\_id` string COMMENT '用户ID',

`order\_id` string COMMENT '订单ID',

`sku\_id` string COMMENT '商品ID',

`refund\_type` string COMMENT '退款类型',

`refund\_num` bigint COMMENT '退款件数',

`refund\_amount` decimal(16,2) COMMENT '退款金额',

`refund\_reason\_type` string COMMENT '退款原因类型',

`create\_time` string COMMENT '退款时间'

) COMMENT '退单表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_order\_refund\_info/';

### 3.3.16 加购表（全量）

hive (gmall)>

drop table if exists ods\_cart\_info;

create external table ods\_cart\_info(

`id` string COMMENT '编号',

`user\_id` string COMMENT '用户id',

`sku\_id` string COMMENT 'skuid',

`cart\_price` string COMMENT '放入购物车时价格',

`sku\_num` string COMMENT '数量',

`sku\_name` string COMMENT 'sku名称 (冗余)',

`create\_time` string COMMENT '创建时间',

`operate\_time` string COMMENT '修改时间',

`is\_ordered` string COMMENT '是否已经下单',

`order\_time` string COMMENT '下单时间'

) COMMENT '加购表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_cart\_info/';

### 3.3.17 商品收藏表（全量）

hive (gmall)>

drop table if exists ods\_favor\_info;

create external table ods\_favor\_info(

`id` string COMMENT '编号',

`user\_id` string COMMENT '用户id',

`sku\_id` string COMMENT 'skuid',

`spu\_id` string COMMENT 'spuid',

`is\_cancel` string COMMENT '是否取消',

`create\_time` string COMMENT '收藏时间',

`cancel\_time` string COMMENT '取消时间'

) COMMENT '商品收藏表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_favor\_info/';

### 3.3.18 优惠券领用表（新增及变化）

hive (gmall)>

drop table if exists ods\_coupon\_use;

create external table ods\_coupon\_use(

`id` string COMMENT '编号',

`coupon\_id` string COMMENT '优惠券ID',

`user\_id` string COMMENT 'skuid',

`order\_id` string COMMENT 'spuid',

`coupon\_status` string COMMENT '优惠券状态',

`get\_time` string COMMENT '领取时间',

`using\_time` string COMMENT '使用时间(下单)',

`used\_time` string COMMENT '使用时间(支付)'

) COMMENT '优惠券领用表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_coupon\_use/';

### 3.3.19 优惠券表（全量）

hive (gmall)>

drop table if exists ods\_coupon\_info;

create external table ods\_coupon\_info(

`id` string COMMENT '购物券编号',

`coupon\_name` string COMMENT '购物券名称',

`coupon\_type` string COMMENT '购物券类型 1 现金券 2 折扣券 3 满减券 4 满件打折券',

`condition\_amount` string COMMENT '满额数',

`condition\_num` string COMMENT '满件数',

`activity\_id` string COMMENT '活动编号',

`benefit\_amount` string COMMENT '减金额',

`benefit\_discount` string COMMENT '折扣',

`create\_time` string COMMENT '创建时间',

`range\_type` string COMMENT '范围类型 1、商品 2、品类 3、品牌',

`spu\_id` string COMMENT '商品id',

`tm\_id` string COMMENT '品牌id',

`category3\_id` string COMMENT '品类id',

`limit\_num` string COMMENT '最多领用次数',

`operate\_time` string COMMENT '修改时间',

`expire\_time` string COMMENT '过期时间'

) COMMENT '优惠券表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_coupon\_info/';

### 3.3.20 活动表（全量）

hive (gmall)>

drop table if exists ods\_activity\_info;

create external table ods\_activity\_info(

`id` string COMMENT '编号',

`activity\_name` string COMMENT '活动名称',

`activity\_type` string COMMENT '活动类型',

`start\_time` string COMMENT '开始时间',

`end\_time` string COMMENT '结束时间',

`create\_time` string COMMENT '创建时间'

) COMMENT '活动表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_activity\_info/';

### 3.3.21 活动订单关联表（增量）

hive (gmall)>

drop table if exists ods\_activity\_order;

create external table ods\_activity\_order(

`id` string COMMENT '编号',

`activity\_id` string COMMENT '优惠券ID',

`order\_id` string COMMENT 'skuid',

`create\_time` string COMMENT '领取时间'

) COMMENT '活动订单关联表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_activity\_order/';

### 3.3.22 优惠规则表（全量）

hive (gmall)>

drop table if exists ods\_activity\_rule;

create external table ods\_activity\_rule(

`id` string COMMENT '编号',

`activity\_id` string COMMENT '活动ID',

`condition\_amount` string COMMENT '满减金额',

`condition\_num` string COMMENT '满减件数',

`benefit\_amount` string COMMENT '优惠金额',

`benefit\_discount` string COMMENT '优惠折扣',

`benefit\_level` string COMMENT '优惠级别'

) COMMENT '优惠规则表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_activity\_rule/';

### 3.3.23 编码字典表（全量）

hive (gmall)>

drop table if exists ods\_base\_dic;

create external table ods\_base\_dic(

`dic\_code` string COMMENT '编号',

`dic\_name` string COMMENT '编码名称',

`parent\_code` string COMMENT '父编码',

`create\_time` string COMMENT '创建日期',

`operate\_time` string COMMENT '操作日期'

) COMMENT '编码字典表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

STORED AS

INPUTFORMAT 'com.hadoop.mapred.DeprecatedLzoTextInputFormat'

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

location '/warehouse/gmall/ods/ods\_base\_dic/';

### 3.3.24 ODS层加载数据脚本

1）在/home/woaini/bin目录下创建脚本hdfs\_to\_ods\_db.sh

[woaini@hadoop102 bin]$ vim hdfs\_to\_ods\_db.sh

在脚本中填写如下内容

#!/bin/bash

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 "$2" ] ;then

do\_date=$2

else

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

fi

sql1="

load data inpath '/origin\_data/$APP/db/order\_info/$do\_date' OVERWRITE into table ${APP}.ods\_order\_info partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/order\_detail/$do\_date' OVERWRITE into table ${APP}.ods\_order\_detail partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/sku\_info/$do\_date' OVERWRITE into table ${APP}.ods\_sku\_info partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/user\_info/$do\_date' OVERWRITE into table ${APP}.ods\_user\_info partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/payment\_info/$do\_date' OVERWRITE into table ${APP}.ods\_payment\_info partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/base\_category1/$do\_date' OVERWRITE into table ${APP}.ods\_base\_category1 partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/base\_category2/$do\_date' OVERWRITE into table ${APP}.ods\_base\_category2 partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/base\_category3/$do\_date' OVERWRITE into table ${APP}.ods\_base\_category3 partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/base\_trademark/$do\_date' OVERWRITE into table ${APP}.ods\_base\_trademark partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/activity\_info/$do\_date' OVERWRITE into table ${APP}.ods\_activity\_info partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/activity\_order/$do\_date' OVERWRITE into table ${APP}.ods\_activity\_order partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/cart\_info/$do\_date' OVERWRITE into table ${APP}.ods\_cart\_info partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/comment\_info/$do\_date' OVERWRITE into table ${APP}.ods\_comment\_info partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/coupon\_info/$do\_date' OVERWRITE into table ${APP}.ods\_coupon\_info partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/coupon\_use/$do\_date' OVERWRITE into table ${APP}.ods\_coupon\_use partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/favor\_info/$do\_date' OVERWRITE into table ${APP}.ods\_favor\_info partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/order\_refund\_info/$do\_date' OVERWRITE into table ${APP}.ods\_order\_refund\_info partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/order\_status\_log/$do\_date' OVERWRITE into table ${APP}.ods\_order\_status\_log partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/spu\_info/$do\_date' OVERWRITE into table ${APP}.ods\_spu\_info partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/activity\_rule/$do\_date' OVERWRITE into table ${APP}.ods\_activity\_rule partition(dt='$do\_date');

load data inpath '/origin\_data/$APP/db/base\_dic/$do\_date' OVERWRITE into table ${APP}.ods\_base\_dic partition(dt='$do\_date');

"

sql2="

load data inpath '/origin\_data/$APP/db/base\_province/$do\_date' OVERWRITE into table ${APP}.ods\_base\_province;

load data inpath '/origin\_data/$APP/db/base\_region/$do\_date' OVERWRITE into table ${APP}.ods\_base\_region;

"

case $1 in

"first"){

$hive -e "$sql1"

$hive -e "$sql2"

};;

"all"){

$hive -e "$sql1"

};;

esac

2）修改权限

[woaini@hadoop102 bin]$ chmod 777 hdfs\_to\_ods\_db.sh

3）初次导入

[woaini@hadoop102 bin]$ hdfs\_to\_ods\_db.sh first 2020-03-10

4）每日导入

[woaini@hadoop102 bin]$ hdfs\_to\_ods\_db.sh all 2020-03-11

5）测试数据是否导入成功

hive (gmall)> select \* from ods\_order\_detail where dt='2020-03-11';

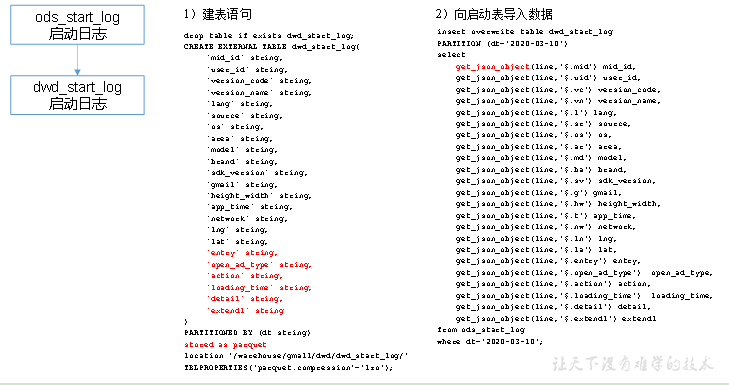
# 第4章 数仓搭建-DWD层

1）对用户行为数据解析

2）对核心数据进行判空过滤。

3）对业务数据采用**维度模型**重新建模，即**维度退化**。

## 4.1 DWD层（用户行为启动表数据解析）



### 4.1.1 创建启动表

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

)

PARTITIONED BY (dt string)

stored as parquet

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

TBLPROPERTIES('parquet.compression'='lzo');

说明：数据采用parquet存储方式，是可以支持切片的，不需要再对数据创建索引。

### 4.1.2 get\_json\_object函数使用

1）输入数据xjson

Xjson=[{"name":"大郎","sex":"男","age":"25"},{"name":"西门庆","sex":"男","age":"47"}]

2）取出第一个json对象

SELECT get\_json\_object(xjson,"$.[0]") FROM person;

结果是：{"name":"大郎","sex":"男","age":"25"}

3）取出第一个json的age字段的值

SELECT get\_json\_object(xjson,"$.[0].age") FROM person;

结果是：25

### 4.1.3 向启动表导入数据

hive (gmall)>

insert overwrite table dwd\_start\_log

PARTITION (dt='2020-03-10')

select

get\_json\_object(line,'$.mid') mid\_id,

get\_json\_object(line,'$.uid') user\_id,

get\_json\_object(line,'$.vc') version\_code,

get\_json\_object(line,'$.vn') version\_name,

get\_json\_object(line,'$.l') lang,

get\_json\_object(line,'$.sr') source,

get\_json\_object(line,'$.os') os,

get\_json\_object(line,'$.ar') area,

get\_json\_object(line,'$.md') model,

get\_json\_object(line,'$.ba') brand,

get\_json\_object(line,'$.sv') sdk\_version,

get\_json\_object(line,'$.g') gmail,

get\_json\_object(line,'$.hw') height\_width,

get\_json\_object(line,'$.t') app\_time,

get\_json\_object(line,'$.nw') network,

get\_json\_object(line,'$.ln') lng,

get\_json\_object(line,'$.la') lat,

get\_json\_object(line,'$.entry') entry,

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

get\_json\_object(line,'$.action') action,

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

get\_json\_object(line,'$.detail') detail,

get\_json\_object(line,'$.extend1') extend1

from ods\_start\_log

where dt='2020-03-10';

3）测试

hive (gmall)>

select \* from dwd\_start\_log where dt='2020-03-10' limit 2;

### 4.1.4 DWD层启动表加载数据脚本

1）在hadoop102的/home/woaini/bin目录下创建脚本

[woaini@hadoop102 bin]$ vim ods\_to\_dwd\_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

sql="

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

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

PARTITION (dt='$do\_date')

select

get\_json\_object(line,'$.mid') mid\_id,

get\_json\_object(line,'$.uid') user\_id,

get\_json\_object(line,'$.vc') version\_code,

get\_json\_object(line,'$.vn') version\_name,

get\_json\_object(line,'$.l') lang,

get\_json\_object(line,'$.sr') source,

get\_json\_object(line,'$.os') os,

get\_json\_object(line,'$.ar') area,

get\_json\_object(line,'$.md') model,

get\_json\_object(line,'$.ba') brand,

get\_json\_object(line,'$.sv') sdk\_version,

get\_json\_object(line,'$.g') gmail,

get\_json\_object(line,'$.hw') height\_width,

get\_json\_object(line,'$.t') app\_time,

get\_json\_object(line,'$.nw') network,

get\_json\_object(line,'$.ln') lng,

get\_json\_object(line,'$.la') lat,

get\_json\_object(line,'$.entry') entry,

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

get\_json\_object(line,'$.action') action,

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

get\_json\_object(line,'$.detail') detail,

get\_json\_object(line,'$.extend1') extend1

from "$APP".ods\_start\_log

where dt='$do\_date';

"

$hive -e "$sql"

2）增加脚本执行权限

[woaini@hadoop102 bin]$ chmod 777 ods\_to\_dwd\_log.sh

3）脚本使用

[woaini@hadoop102 module]$ ods\_to\_dwd\_log.sh 2020-03-11

4）查询导入结果

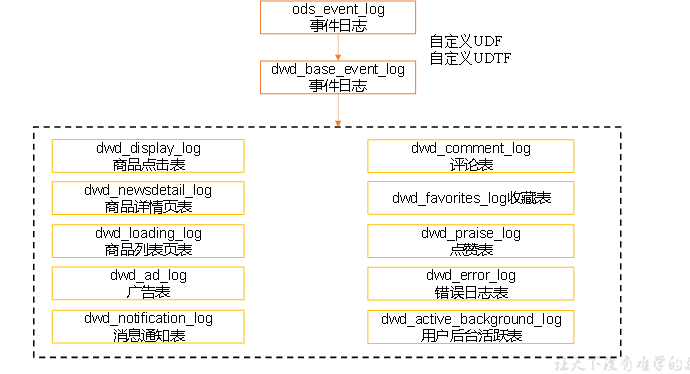
hive (gmall)>

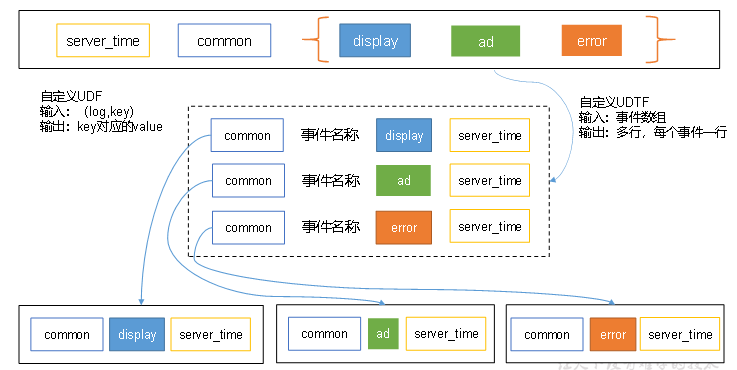
select \* from dwd\_start\_log where dt='2020-03-11' limit 2;

5）脚本执行时间

企业开发中一般在每日凌晨30分~1点

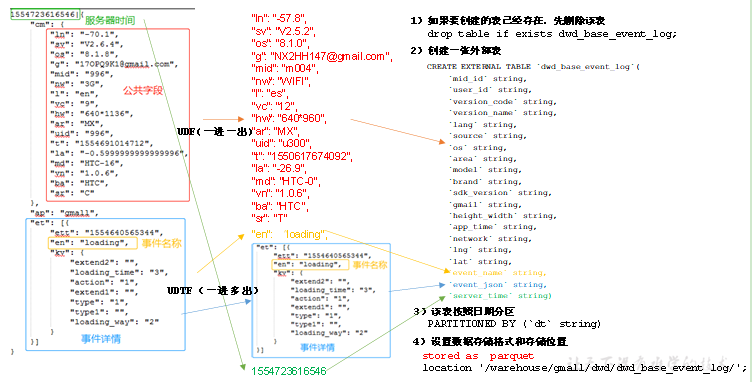
## 4.2 DWD层（用户行为事件表数据解析）





### 4.2.1 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

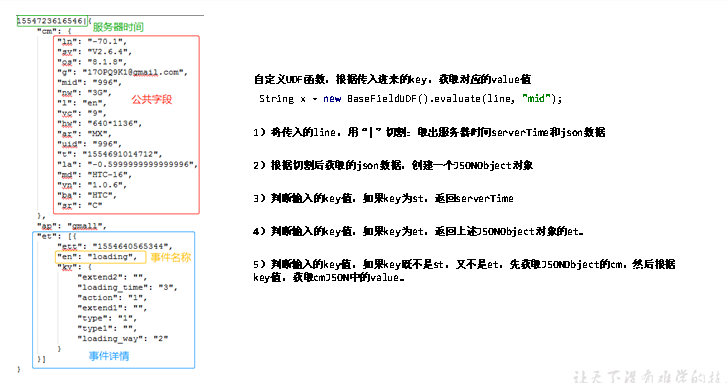
location '/warehouse/gmall/dwd/dwd\_base\_event\_log/'

TBLPROPERTIES('parquet.compression'='lzo');

2）说明：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。

### 4.2.2 自定义UDF函数（解析公共字段）

UDF函数特点：一行进一行出。简称，一进一出。



1）创建一个maven工程：hivefunction

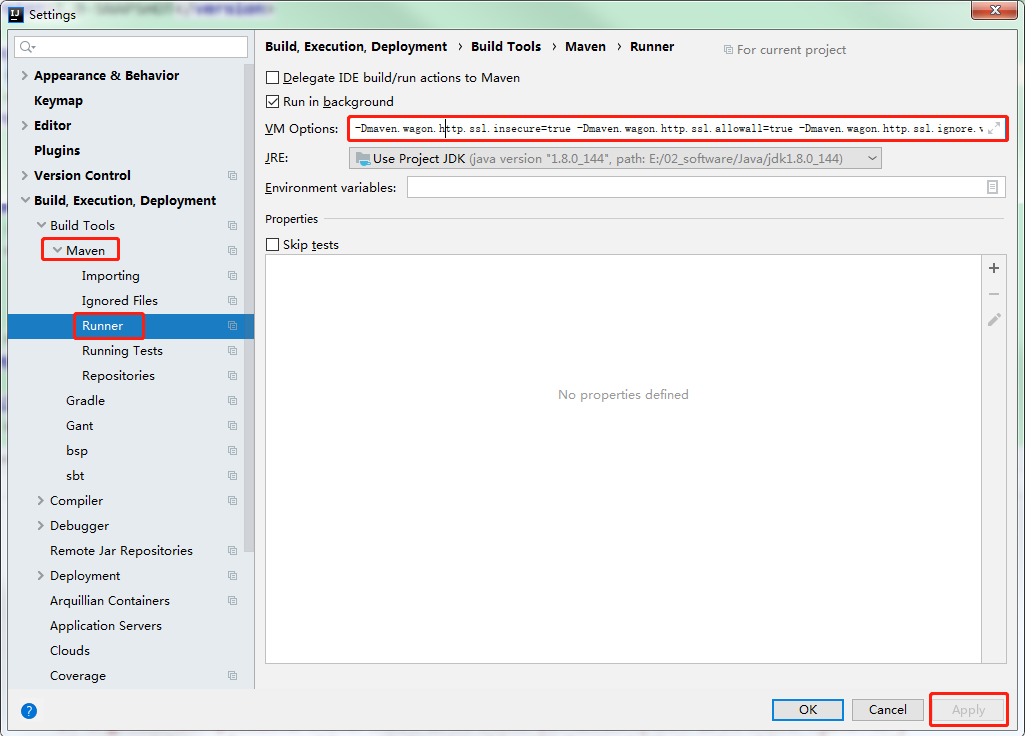
2）创建包名：com.woaini.udf

3）在pom.xml文件中添加如下内容

<**properties**>  
 <**hive.version**>2.3.0</**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**>

**注意1：**如果hive的jar包下载失败，可以将如下参数配置添加到idea中

-Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true -Dmaven.wagon.http.ssl.ignore.validity.dates=true



详见：<https://blog.csdn.net/qq_22041375/article/details/103491941>

**注意2：**如果提示pentaho-aggdesigner-algorithm.jar包下载失败，需要在maven的pom中增加如下仓库

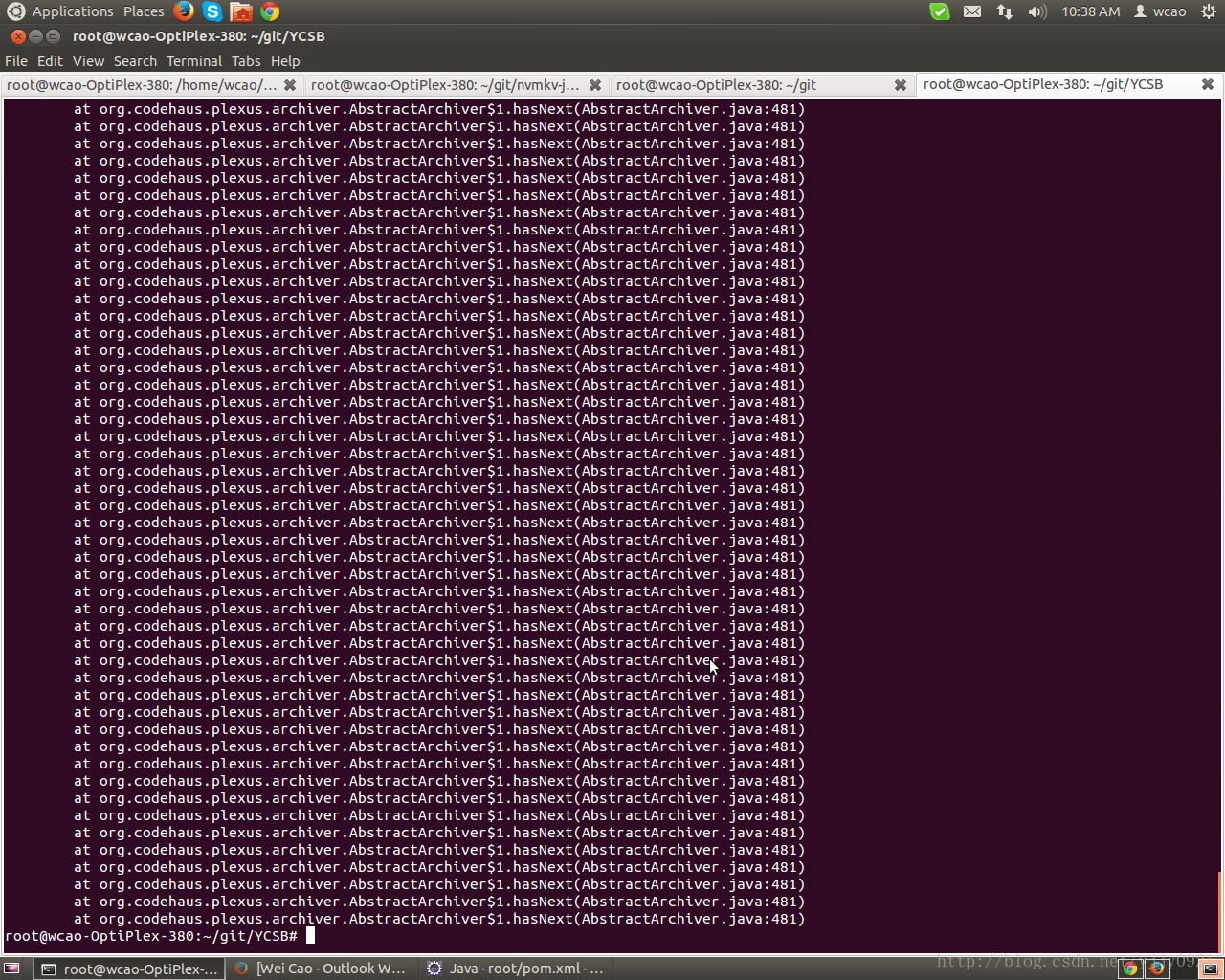
<**repositories**>  
 <**repository**>  
 <**id**>spring-plugin</**id**>  
 <**url**>https://repo.spring.io/plugins-release/</**url**>  
 </**repository**>  
</**repositories**>

**注意3：**如果出现如下图片中情况，说明idea内存溢出

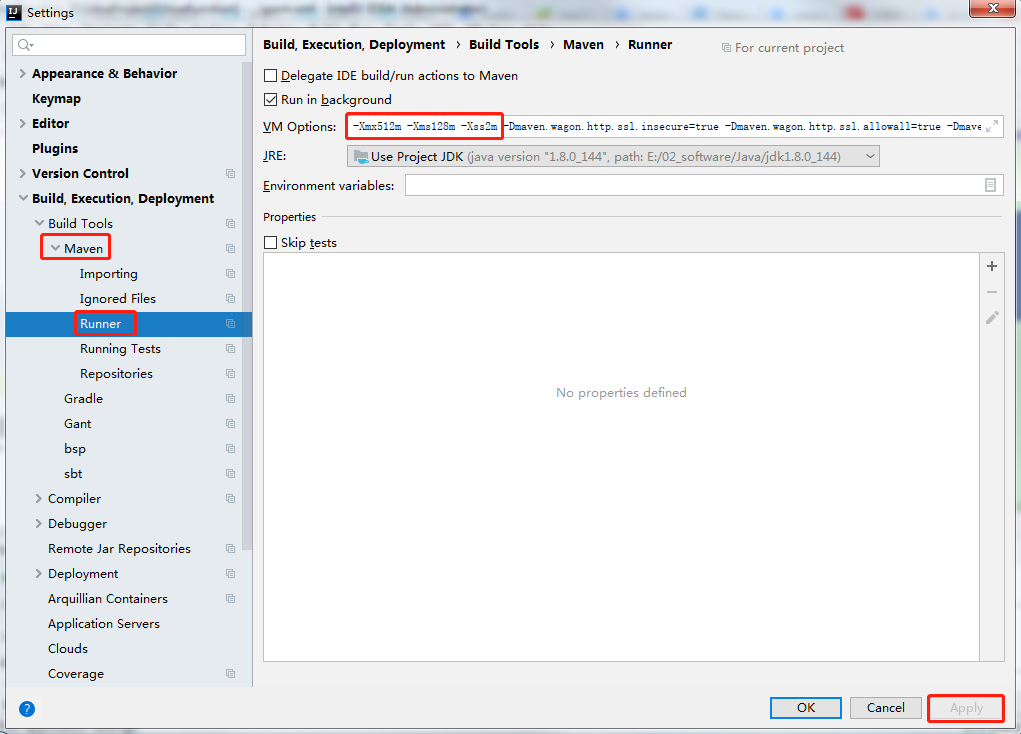
Exception in thread "main" java.lang.StackOverflowError

at sun.nio.cs.UTF\_8$Encoder.encodeLoop(UTF\_8.java:691)

at java.nio.charset.CharsetEncoder.encode(CharsetEncoder.java:579)



修改办法：把-Xmx512 -Xms128m -Xss2m添加到下图位置。



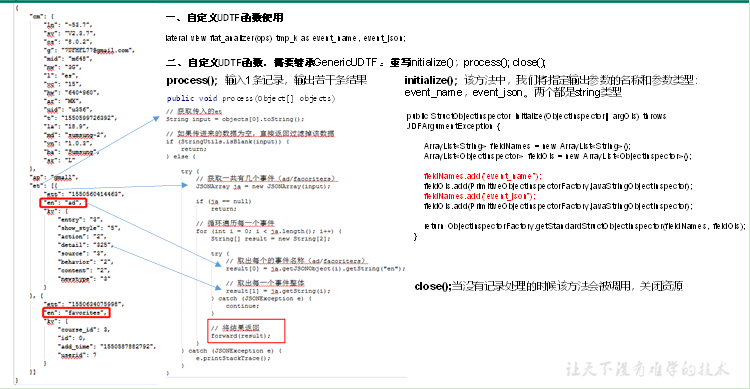
4）UDF用于解析公共字段

**package** com.woaini.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 key) **throws** JSONException {  
  
 String[] log = line.split(**"\\|"**);  
  
 **if** (log.**length** != 2 || StringUtils.*isBlank*(log[1])) {  
 **return ""**;  
 }  
  
 JSONObject baseJson = **new** JSONObject(log[1].trim());  
  
 String result = **""**;  
  
 *// 获取服务器时间* **if** (**"st"**.equals(key)) {  
 result = log[0].trim();  
 } **else if** (**"et"**.equals(key)) {  
 *// 获取事件数组* **if** (baseJson.has(**"et"**)) {  
 result = baseJson.getString(**"et"**);  
 }  
 } **else** {  
 JSONObject cm = baseJson.getJSONObject(**"cm"**);  
  
 *// 获取key对应公共字段的value* **if** (cm.has(key)) {  
 result = cm.getString(key);  
 }  
 }  
  
 **return** result;  
 }  
  
 **public static void** main (String[] args) **throws** JSONException {  
  
 String line = **"1583776223469|{\"cm\":{\"ln\":\"-48.5\",\"sv\":\"V2.5.7\",\"os\":\"8.0.9\",\"g\":\"6F76AVD5@gmail.com\",\"mid\":\"0\",\"nw\":\"4G\",\"l\":\"pt\",\"vc\":\"3\",\"hw\":\"750\*1134\",\"ar\":\"MX\",\"uid\":\"0\",\"t\":\"1583707297317\",\"la\":\"-52.9\",\"md\":\"sumsung-18\",\"vn\":\"1.2.4\",\"ba\":\"Sumsung\",\"sr\":\"V\"},\"ap\":\"app\",\"et\":[{\"ett\":\"1583705574227\",\"en\":\"display\",\"kv\":{\"goodsid\":\"0\",\"action\":\"1\",\"extend1\":\"1\",\"place\":\"0\",\"category\":\"63\"}},{\"ett\":\"1583760986259\",\"en\":\"loading\",\"kv\":{\"extend2\":\"\",\"loading\_time\":\"4\",\"action\":\"3\",\"extend1\":\"\",\"type\":\"3\",\"type1\":\"\",\"loading\_way\":\"1\"}},{\"ett\":\"1583746639124\",\"en\":\"ad\",\"kv\":{\"activityId\":\"1\",\"displayMills\":\"111839\",\"entry\":\"1\",\"action\":\"5\",\"contentType\":\"0\"}},{\"ett\":\"1583758016208\",\"en\":\"notification\",\"kv\":{\"ap\_time\":\"1583694079866\",\"action\":\"1\",\"type\":\"3\",\"content\":\"\"}},{\"ett\":\"1583699890760\",\"en\":\"favorites\",\"kv\":{\"course\_id\":4,\"id\":0,\"add\_time\":\"1583730648134\",\"userid\":7}}]}"**;  
  
 String mid = **new** BaseFieldUDF().evaluate(line, **"mid"**);  
  
 System.***out***.println(mid);  
 }  
}

注意：使用main函数主要用于模拟数据测试。

### 4.2.3 自定义UDTF函数（解析事件字段）

UDTF函数特点：多行进多行出。 简称，多进多出。



1）创建包名：com.woaini.udtf

2）在com.woaini.udtf包下创建类名：EventJsonUDTF

3）用于展开业务字段

**package** com.woaini.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：* **public** StructObjectInspector initialize(StructObjectInspector 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.jar上传到hadoop102的/opt/module，然后再将该jar包上传到HDFS的/user/hive/jars路径下

[woaini@hadoop102 module]$ hadoop fs -mkdir -p /user/hive/jars

[woaini@hadoop102 module]$ hadoop fs -put hivefunction-1.0-SNAPSHOT.jar /user/hive/jars

4）创建永久函数与开发好的java class关联

hive (gmall)>

create function base\_analizer as 'com.woaini.udf.BaseFieldUDF' using jar 'hdfs://hadoop102:9000/user/hive/jars/hivefunction-1.0-SNAPSHOT.jar';

create function flat\_analizer as 'com.woaini.udtf.EventJsonUDTF' using jar 'hdfs://hadoop102:9000/user/hive/jars/hivefunction-1.0-SNAPSHOT.jar';

**5）注意：**如果修改了自定义函数重新生成jar包怎么处理？只需要替换HDFS路径上的旧jar包，然后重启Hive客户端即可。

### 4.2.4 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)>

insert overwrite table dwd\_base\_event\_log partition(dt='2020-03-10')

select

base\_analizer(line,'mid') as mid\_id,

base\_analizer(line,'uid') as user\_id,

base\_analizer(line,'vc') as version\_code,

base\_analizer(line,'vn') as version\_name,

base\_analizer(line,'l') as lang,

base\_analizer(line,'sr') as source,

base\_analizer(line,'os') as os,

base\_analizer(line,'ar') as area,

base\_analizer(line,'md') as model,

base\_analizer(line,'ba') as brand,

base\_analizer(line,'sv') as sdk\_version,

base\_analizer(line,'g') as gmail,

base\_analizer(line,'hw') as height\_width,

base\_analizer(line,'t') as app\_time,

base\_analizer(line,'nw') as network,

base\_analizer(line,'ln') as lng,

base\_analizer(line,'la') as lat,

event\_name,

event\_json,

base\_analizer(line,'st') as server\_time

from ods\_event\_log lateral view flat\_analizer(base\_analizer(line,'et')) tmp\_flat as event\_name,event\_json

where dt='2020-03-10' and base\_analizer(line,'et')<>'';

2）测试

hive (gmall)>

select \* from dwd\_base\_event\_log where dt='2020-03-10' limit 2;

### 4.2.5 DWD层数据解析脚本

1）在hadoop102的/home/woaini/bin目录下创建脚本

[woaini@hadoop102 bin]$ vim ods\_to\_dwd\_base\_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

sql="

use gmall;

insert overwrite table "$APP".dwd\_base\_event\_log partition(dt='$do\_date')

select

base\_analizer(line,'mid') as mid\_id,

base\_analizer(line,'uid') as user\_id,

base\_analizer(line,'vc') as version\_code,

base\_analizer(line,'vn') as version\_name,

base\_analizer(line,'l') as lang,

base\_analizer(line,'sr') as source,

base\_analizer(line,'os') as os,

base\_analizer(line,'ar') as area,

base\_analizer(line,'md') as model,

base\_analizer(line,'ba') as brand,

base\_analizer(line,'sv') as sdk\_version,

base\_analizer(line,'g') as gmail,

base\_analizer(line,'hw') as height\_width,

base\_analizer(line,'t') as app\_time,

base\_analizer(line,'nw') as network,

base\_analizer(line,'ln') as lng,

base\_analizer(line,'la') as lat,

event\_name,

event\_json,

base\_analizer(line,'st') as server\_time

from "$APP".ods\_event\_log lateral view flat\_analizer(base\_analizer(line,'et')) tem\_flat as event\_name,event\_json

where dt='$do\_date' and base\_analizer(line,'et')<>'';

"

$hive -e "$sql"

**注意：**使用自定义函数时，需要在执行脚本前，增加上要使用的库。例如：use gmall;

2）增加脚本执行权限

[woaini@hadoop102 bin]$ chmod 777 ods\_to\_dwd\_base\_log.sh

3）脚本使用

[woaini@hadoop102 module]$ ods\_to\_dwd\_base\_log.sh 2020-03-11

4）查询导入结果

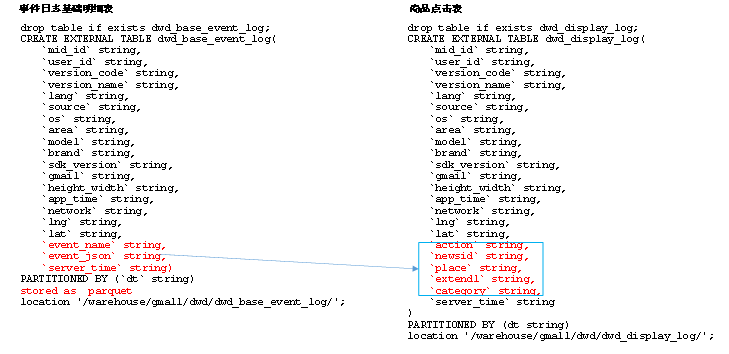
hive (gmall)>

select \* from dwd\_base\_event\_log where dt='2020-03-11' limit 2;

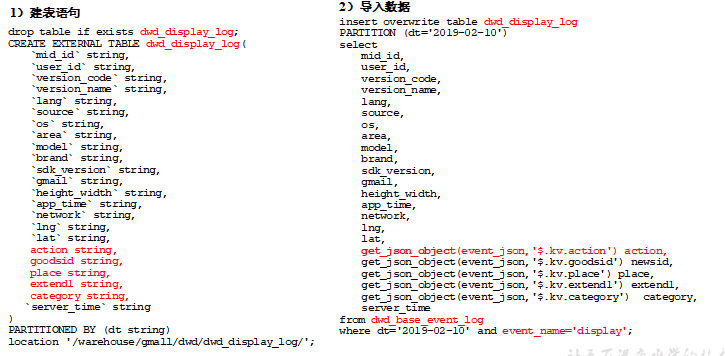
5）脚本执行时间

企业开发中一般在每日凌晨30分~1点

## 4.3 DWD层（用户行为事件表获取）



### 4.3.1 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,

`goodsid` string,

`place` string,

`extend1` string,

`category` string,

`server\_time` string

)

PARTITIONED BY (dt string)

stored as parquet

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

TBLPROPERTIES('parquet.compression'='lzo');

2）导入数据

hive (gmall)>

insert overwrite table dwd\_display\_log PARTITION (dt='2020-03-10')

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.goodsid') goodsid,

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

from dwd\_base\_event\_log

where dt='2020-03-10' and event\_name='display';

3）测试

hive (gmall)> select \* from dwd\_display\_log where dt='2020-03-10' limit 2;

### 4.3.2 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,

`goodsid` string,

`showtype` string,

`news\_staytime` string,

`loading\_time` string,

`type1` string,

`category` string,

`server\_time` string)

PARTITIONED BY (dt string)

stored as parquet

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

TBLPROPERTIES('parquet.compression'='lzo');

2）导入数据

hive (gmall)>

insert overwrite table dwd\_newsdetail\_log PARTITION (dt='2020-03-10')

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.goodsid') goodsid,

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

from dwd\_base\_event\_log

where dt='2020-03-10' and event\_name='newsdetail';

3）测试

hive (gmall)>

select \* from dwd\_newsdetail\_log where dt='2020-03-10' limit 2;

### 4.3.3 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)

stored as parquet

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

TBLPROPERTIES('parquet.compression'='lzo');

2）导入数据

hive (gmall)>

insert overwrite table dwd\_loading\_log PARTITION (dt='2020-03-10')

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

from dwd\_base\_event\_log

where dt='2020-03-10' and event\_name='loading';

3）测试

hive (gmall)> select \* from dwd\_loading\_log where dt='2020-03-10' limit 2;

### 4.3.4 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,

`contentType` string,

`displayMills` string,

`itemId` string,

`activityId` string,

`server\_time` string)

PARTITIONED BY (dt string)

stored as parquet

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

TBLPROPERTIES('parquet.compression'='lzo');

2）导入数据

hive (gmall)>

insert overwrite table dwd\_ad\_log PARTITION (dt='2020-03-10')

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.contentType') contentType,

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

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

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

server\_time

from dwd\_base\_event\_log

where dt='2020-03-10' and event\_name='ad';

3）测试

hive (gmall)> select \* from dwd\_ad\_log where dt='2020-03-10' limit 2;

### 4.3.5 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)

stored as parquet

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

TBLPROPERTIES('parquet.compression'='lzo');

2）导入数据

hive (gmall)>

insert overwrite table dwd\_notification\_log PARTITION (dt='2020-03-10')

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

from dwd\_base\_event\_log

where dt='2020-03-10' and event\_name='notification';

3）测试

hive (gmall)>

select \* from dwd\_notification\_log where dt='2020-03-10' limit 2;

### 4.3.6 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)

stored as parquet

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

TBLPROPERTIES('parquet.compression'='lzo');

2）导入数据

hive (gmall)>

insert overwrite table dwd\_active\_background\_log PARTITION (dt='2020-03-10')

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

from dwd\_base\_event\_log

where dt='2020-03-10' and event\_name='active\_background';

3）测试

hive (gmall)>

select \* from dwd\_active\_background\_log where dt='2020-03-10' limit 2;

### 4.3.7 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)

stored as parquet

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

TBLPROPERTIES('parquet.compression'='lzo');

2）导入数据

hive (gmall)>

insert overwrite table dwd\_comment\_log PARTITION (dt='2020-03-10')

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

from dwd\_base\_event\_log

where dt='2020-03-10' and event\_name='comment';

3）测试

hive (gmall)> select \* from dwd\_comment\_log where dt='2020-03-10' limit 2;

### 4.3.8 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)

stored as parquet

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

TBLPROPERTIES('parquet.compression'='lzo');

2）导入数据

hive (gmall)>

insert overwrite table dwd\_favorites\_log PARTITION (dt='2020-03-10')

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

from dwd\_base\_event\_log

where dt='2020-03-10' and event\_name='favorites';

3）测试

hive (gmall)>

select \* from dwd\_favorites\_log where dt='2020-03-10' limit 2;

### 4.3.9 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)

stored as parquet

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

TBLPROPERTIES('parquet.compression'='lzo');

2）导入数据

hive (gmall)>

insert overwrite table dwd\_praise\_log PARTITION (dt='2020-03-10')

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

from dwd\_base\_event\_log

where dt='2020-03-10' and event\_name='praise';

3）测试

hive (gmall)> select \* from dwd\_praise\_log where dt='2020-03-10' limit 2;

### 4.3.10 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)

stored as parquet

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

TBLPROPERTIES('parquet.compression'='lzo');

2）导入数据

hive (gmall)>

insert overwrite table dwd\_error\_log PARTITION (dt='2020-03-10')

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

from dwd\_base\_event\_log

where dt='2020-03-10' and event\_name='error';

3）测试

hive (gmall)> select \* from dwd\_error\_log where dt='2020-03-10' limit 2;

### 4.3.11 DWD层事件表加载数据脚本

1）在hadoop102的/home/woaini/bin目录下创建脚本

[woaini@hadoop102 bin]$ vim ods\_to\_dwd\_event\_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

sql="

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

PARTITION (dt='$do\_date')

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.goodsid') goodsid,

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

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

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

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

PARTITION (dt='$do\_date')

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.goodsid') goodsid,

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

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

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

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

PARTITION (dt='$do\_date')

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

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

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

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

PARTITION (dt='$do\_date')

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.contentType') contentType,

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

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

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

server\_time

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

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

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

PARTITION (dt='$do\_date')

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

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

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

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

PARTITION (dt='$do\_date')

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

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

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

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

PARTITION (dt='$do\_date')

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

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

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

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

PARTITION (dt='$do\_date')

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

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

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

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

PARTITION (dt='$do\_date')

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

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

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

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

PARTITION (dt='$do\_date')

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

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

[woaini@hadoop102 bin]$ chmod 777 ods\_to\_dwd\_event\_log.sh

3）脚本使用

[woaini@hadoop102 module]$ ods\_to\_dwd\_event\_log.sh 2020-03-11

4）查询导入结果

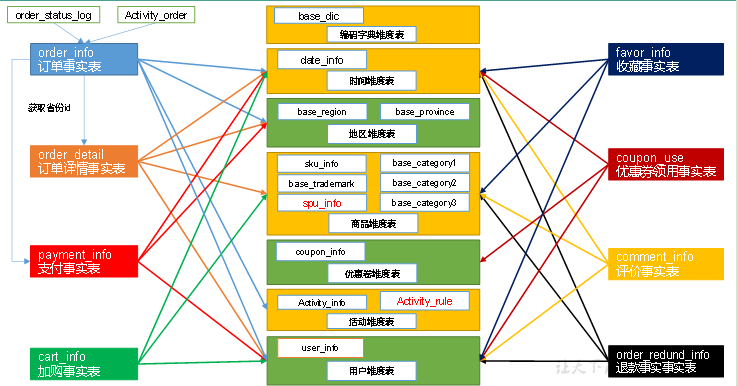
hive (gmall)>

select \* from dwd\_comment\_log where dt='2020-03-11' limit 2;

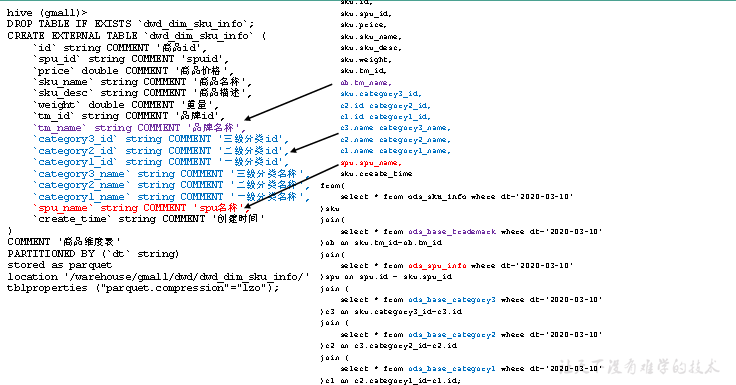
5）脚本执行时间

企业开发中一般在每日凌晨30分~1点

## 4.4 DWD层（业务数据）



### 4.4.1 商品维度表（全量表）



1）建表语句

hive (gmall)>

DROP TABLE IF EXISTS `dwd\_dim\_sku\_info`;

CREATE EXTERNAL TABLE `dwd\_dim\_sku\_info` (

`id` string COMMENT '商品id',

`spu\_id` string COMMENT 'spuid',

`price` double COMMENT '商品价格',

`sku\_name` string COMMENT '商品名称',

`sku\_desc` string COMMENT '商品描述',

`weight` double COMMENT '重量',

`tm\_id` string COMMENT '品牌id',

`tm\_name` string COMMENT '品牌名称',

`category3\_id` string COMMENT '三级分类id',

`category2\_id` string COMMENT '二级分类id',

`category1\_id` string COMMENT '一级分类id',

`category3\_name` string COMMENT '三级分类名称',

`category2\_name` string COMMENT '二级分类名称',

`category1\_name` string COMMENT '一级分类名称',

`spu\_name` string COMMENT 'spu名称',

`create\_time` string COMMENT '创建时间'

)

COMMENT '商品维度表'

PARTITIONED BY (`dt` string)

stored as parquet

location '/warehouse/gmall/dwd/dwd\_dim\_sku\_info/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwd\_dim\_sku\_info partition(dt='2020-03-10')

select

sku.id,

sku.spu\_id,

sku.price,

sku.sku\_name,

sku.sku\_desc,

sku.weight,

sku.tm\_id,

ob.tm\_name,

sku.category3\_id,

c2.id category2\_id,

c1.id category1\_id,

c3.name category3\_name,

c2.name category2\_name,

c1.name category1\_name,

spu.spu\_name,

sku.create\_time

from

(

select \* from ods\_sku\_info where dt='2020-03-10'

)sku

join

(

select \* from ods\_base\_trademark where dt='2020-03-10'

)ob on sku.tm\_id=ob.tm\_id

join

(

select \* from ods\_spu\_info where dt='2020-03-10'

)spu on spu.id = sku.spu\_id

join

(

select \* from ods\_base\_category3 where dt='2020-03-10'

)c3 on sku.category3\_id=c3.id

join

(

select \* from ods\_base\_category2 where dt='2020-03-10'

)c2 on c3.category2\_id=c2.id

join

(

select \* from ods\_base\_category1 where dt='2020-03-10'

)c1 on c2.category1\_id=c1.id;

3）查询加载结果

hive (gmall)> select \* from dwd\_dim\_sku\_info where dt='2020-03-10';

### 4.4.2 优惠券信息表（全量）

把ODS层ods\_coupon\_info表数据导入到DWD层优惠卷信息表，在导入过程中可以做适当的清洗。

1）建表语句

hive (gmall)>

drop table if exists dwd\_dim\_coupon\_info;

create external table dwd\_dim\_coupon\_info(

`id` string COMMENT '购物券编号',

`coupon\_name` string COMMENT '购物券名称',

`coupon\_type` string COMMENT '购物券类型 1 现金券 2 折扣券 3 满减券 4 满件打折券',

`condition\_amount` string COMMENT '满额数',

`condition\_num` string COMMENT '满件数',

`activity\_id` string COMMENT '活动编号',

`benefit\_amount` string COMMENT '减金额',

`benefit\_discount` string COMMENT '折扣',

`create\_time` string COMMENT '创建时间',

`range\_type` string COMMENT '范围类型 1、商品 2、品类 3、品牌',

`spu\_id` string COMMENT '商品id',

`tm\_id` string COMMENT '品牌id',

`category3\_id` string COMMENT '品类id',

`limit\_num` string COMMENT '最多领用次数',

`operate\_time` string COMMENT '修改时间',

`expire\_time` string COMMENT '过期时间'

) COMMENT '优惠券信息表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

stored as parquet

location '/warehouse/gmall/dwd/dwd\_dim\_coupon\_info/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwd\_dim\_coupon\_info partition(dt='2020-03-10')

select

id,

coupon\_name,

coupon\_type,

condition\_amount,

condition\_num,

activity\_id,

benefit\_amount,

benefit\_discount,

create\_time,

range\_type,

spu\_id,

tm\_id,

category3\_id,

limit\_num,

operate\_time,

expire\_time

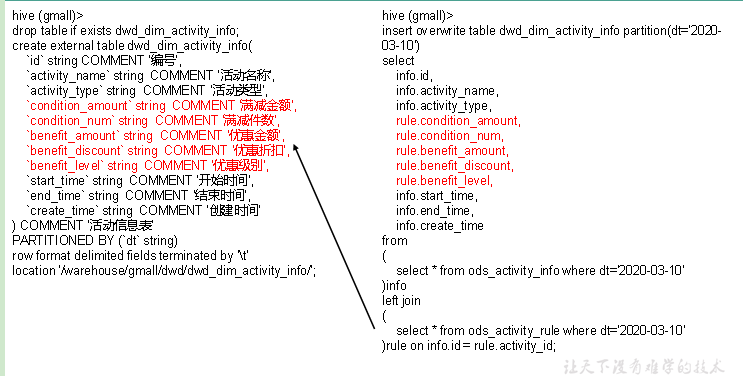
from ods\_coupon\_info

where dt='2020-03-10';

3）查询加载结果

hive (gmall)> select \* from dwd\_dim\_coupon\_info where dt='2020-03-10';

### 4.4.3 活动维度表（全量）



1）建表语句

hive (gmall)>

drop table if exists dwd\_dim\_activity\_info;

create external table dwd\_dim\_activity\_info(

`id` string COMMENT '编号',

`activity\_name` string COMMENT '活动名称',

`activity\_type` string COMMENT '活动类型',

`condition\_amount` string COMMENT '满减金额',

`condition\_num` string COMMENT '满减件数',

`benefit\_amount` string COMMENT '优惠金额',

`benefit\_discount` string COMMENT '优惠折扣',

`benefit\_level` string COMMENT '优惠级别',

`start\_time` string COMMENT '开始时间',

`end\_time` string COMMENT '结束时间',

`create\_time` string COMMENT '创建时间'

) COMMENT '活动信息表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

stored as parquet

location '/warehouse/gmall/dwd/dwd\_dim\_activity\_info/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwd\_dim\_activity\_info partition(dt='2020-03-10')

select

info.id,

info.activity\_name,

info.activity\_type,

rule.condition\_amount,

rule.condition\_num,

rule.benefit\_amount,

rule.benefit\_discount,

rule.benefit\_level,

info.start\_time,

info.end\_time,

info.create\_time

from

(

select \* from ods\_activity\_info where dt='2020-03-10'

)info

left join

(

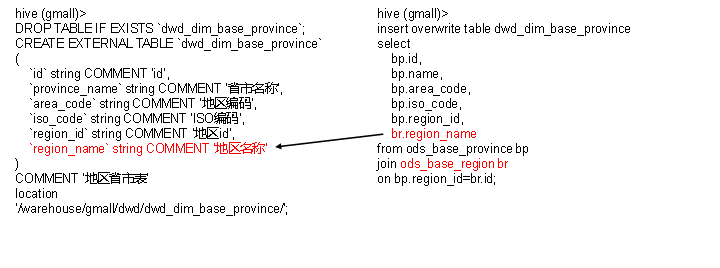
select \* from ods\_activity\_rule where dt='2020-03-10'

)rule on info.id = rule.activity\_id;

3）查询加载结果

hive (gmall)> select \* from dwd\_dim\_activity\_info where dt='2020-03-10';

### 4.4.4 地区维度表（特殊）



1）建表语句

hive (gmall)>

DROP TABLE IF EXISTS `dwd\_dim\_base\_province`;

CREATE EXTERNAL TABLE `dwd\_dim\_base\_province` (

`id` string COMMENT 'id',

`province\_name` string COMMENT '省市名称',

`area\_code` string COMMENT '地区编码',

`iso\_code` string COMMENT 'ISO编码',

`region\_id` string COMMENT '地区id',

`region\_name` string COMMENT '地区名称'

)

COMMENT '地区省市表'

stored as parquet

location '/warehouse/gmall/dwd/dwd\_dim\_base\_province/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwd\_dim\_base\_province

select

bp.id,

bp.name,

bp.area\_code,

bp.iso\_code,

bp.region\_id,

br.region\_name

from ods\_base\_province bp

join ods\_base\_region br

on bp.region\_id=br.id;

3）查询加载结果

hive (gmall)> select \* from dwd\_dim\_base\_province;

### 4.4.5 时间维度表（特殊）（预留）

1）建表语句

hive (gmall)>

DROP TABLE IF EXISTS `dwd\_dim\_date\_info`;

CREATE EXTERNAL TABLE `dwd\_dim\_date\_info`(

`date\_id` string COMMENT '日',

`week\_id` int COMMENT '周',

`week\_day` int COMMENT '周的第几天',

`day` int COMMENT '每月的第几天',

`month` int COMMENT '第几月',

`quarter` int COMMENT '第几季度',

`year` int COMMENT '年',

`is\_workday` int COMMENT '是否是周末',

`holiday\_id` int COMMENT '是否是节假日'

)

row format delimited fields terminated by '\t'

stored as parquet

location '/warehouse/gmall/dwd/dwd\_dim\_date\_info/'

tblproperties ("parquet.compression"="lzo");

2）把date\_info.txt文件上传到hadoop102的/opt/module/db\_log/路径

3）数据装载

hive (gmall)>

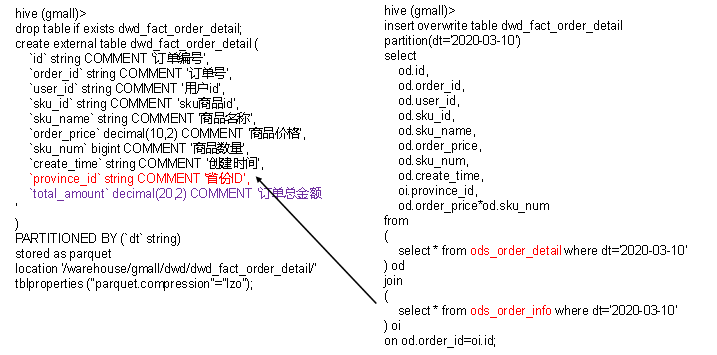
load data local inpath '/opt/module/db\_log/date\_info.txt' into table dwd\_dim\_date\_info;

4）查询加载结果

hive (gmall)> select \* from dwd\_dim\_date\_info;

### 4.4.6 订单明细事实表（事务型快照事实表）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **时间** | **用户** | **地区** | **商品** | **优惠券** | **活动** | **编码** | **度量值** |
| **订单详情** | √ |  | √ | √ |  |  |  | 件数/金额 |



1）建表语句

hive (gmall)>

drop table if exists dwd\_fact\_order\_detail;

create external table dwd\_fact\_order\_detail (

`id` string COMMENT '订单编号',

`order\_id` string COMMENT '订单号',

`user\_id` string COMMENT '用户id',

`sku\_id` string COMMENT 'sku商品id',

`sku\_name` string COMMENT '商品名称',

`order\_price` decimal(10,2) COMMENT '商品价格',

`sku\_num` bigint COMMENT '商品数量',

`create\_time` string COMMENT '创建时间',

`province\_id` string COMMENT '省份ID',

`total\_amount` decimal(20,2) COMMENT '订单总金额'

)

PARTITIONED BY (`dt` string)

stored as parquet

location '/warehouse/gmall/dwd/dwd\_fact\_order\_detail/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwd\_fact\_order\_detail partition(dt='2020-03-10')

select

od.id,

od.order\_id,

od.user\_id,

od.sku\_id,

od.sku\_name,

od.order\_price,

od.sku\_num,

od.create\_time,

oi.province\_id,

od.order\_price\*od.sku\_num

from

(

select \* from ods\_order\_detail where dt='2020-03-10'

) od

join

(

select \* from ods\_order\_info where dt='2020-03-10'

) oi

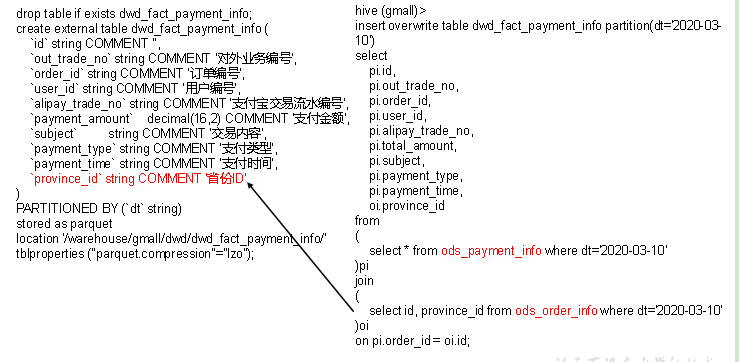
on od.order\_id=oi.id;

3）查询加载结果

hive (gmall)> select \* from dwd\_fact\_order\_detail where dt='2020-03-10';

### 4.4.7 支付事实表（事务型快照事实表）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **时间** | **用户** | **地区** | **商品** | **优惠券** | **活动** | **编码** | **度量值** |
| **支付** | √ |  | √ |  |  |  |  | 金额 |



1）建表语句

hive (gmall)>

drop table if exists dwd\_fact\_payment\_info;

create external table dwd\_fact\_payment\_info (

`id` string COMMENT '',

`out\_trade\_no` string COMMENT '对外业务编号',

`order\_id` string COMMENT '订单编号',

`user\_id` string COMMENT '用户编号',

`alipay\_trade\_no` string COMMENT '支付宝交易流水编号',

`payment\_amount` decimal(16,2) COMMENT '支付金额',

`subject` string COMMENT '交易内容',

`payment\_type` string COMMENT '支付类型',

`payment\_time` string COMMENT '支付时间',

`province\_id` string COMMENT '省份ID'

)

PARTITIONED BY (`dt` string)

stored as parquet

location '/warehouse/gmall/dwd/dwd\_fact\_payment\_info/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwd\_fact\_payment\_info partition(dt='2020-03-10')

select

pi.id,

pi.out\_trade\_no,

pi.order\_id,

pi.user\_id,

pi.alipay\_trade\_no,

pi.total\_amount,

pi.subject,

pi.payment\_type,

pi.payment\_time,

oi.province\_id

from

(

select \* from ods\_payment\_info where dt='2020-03-10'

)pi

join

(

select id, province\_id from ods\_order\_info where dt='2020-03-10'

)oi

on pi.order\_id = oi.id;

3）查询加载结果

hive (gmall)> select \* from dwd\_fact\_payment\_info where dt='2020-03-10';

### 4.4.8 退款事实表（事务型快照事实表）

把ODS层ods\_order\_refund\_info表数据导入到DWD层退款事实表，在导入过程中可以做适当的清洗。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **时间** | **用户** | **地区** | **商品** | **优惠券** | **活动** | **编码** | **度量值** |
| **退款** | √ | √ |  | √ |  |  |  | 件数/金额 |

1）建表语句

hive (gmall)>

drop table if exists dwd\_fact\_order\_refund\_info;

create external table dwd\_fact\_order\_refund\_info(

`id` string COMMENT '编号',

`user\_id` string COMMENT '用户ID',

`order\_id` string COMMENT '订单ID',

`sku\_id` string COMMENT '商品ID',

`refund\_type` string COMMENT '退款类型',

`refund\_num` bigint COMMENT '退款件数',

`refund\_amount` decimal(16,2) COMMENT '退款金额',

`refund\_reason\_type` string COMMENT '退款原因类型',

`create\_time` string COMMENT '退款时间'

) COMMENT '退款事实表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

stored as parquet

location '/warehouse/gmall/dwd/dwd\_fact\_order\_refund\_info/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwd\_fact\_order\_refund\_info partition(dt='2020-03-10')

select

id,

user\_id,

order\_id,

sku\_id,

refund\_type,

refund\_num,

refund\_amount,

refund\_reason\_type,

create\_time

from ods\_order\_refund\_info

where dt='2020-03-10';

3）查询加载结果

hive (gmall)> select \* from dwd\_fact\_order\_refund\_info where dt='2020-03-10';

### 4.4.9 评价事实表（事务型快照事实表）

把ODS层ods\_comment\_info表数据导入到DWD层评价事实表，在导入过程中可以做适当的清洗。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **时间** | **用户** | **地区** | **商品** | **优惠券** | **活动** | **编码** | **度量值** |
| **评价** | √ | √ |  | √ |  |  |  | 评价类型 |

1）建表语句

hive (gmall)>

drop table if exists dwd\_fact\_comment\_info;

create external table dwd\_fact\_comment\_info(

`id` string COMMENT '编号',

`user\_id` string COMMENT '用户ID',

`sku\_id` string COMMENT '商品sku',

`spu\_id` string COMMENT '商品spu',

`order\_id` string COMMENT '订单ID',

`appraise` string COMMENT '评价',

`create\_time` string COMMENT '评价时间'

) COMMENT '评价事实表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

stored as parquet

location '/warehouse/gmall/dwd/dwd\_fact\_comment\_info/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwd\_fact\_comment\_info partition(dt='2020-03-10')

select

id,

user\_id,

sku\_id,

spu\_id,

order\_id,

appraise,

create\_time

from ods\_comment\_info

where dt='2020-03-10';

3）查询加载结果

hive (gmall)> select \* from dwd\_fact\_comment\_info where dt='2020-03-10';

### 4.4.10 加购事实表（周期型快照事实表，每日快照）

由于购物车的数量是会发生变化，所以导增量不合适。

每天做一次快照，导入的数据是全量，区别于事务型事实表是每天导入新增。

周期型快照事实表劣势：存储的数据量会比较大。

解决方案：周期型快照事实表存储的数据比较讲究时效性，时间太久了的意义不大，可以删除以前的数据。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **时间** | **用户** | **地区** | **商品** | **优惠券** | **活动** | **编码** | **度量值** |
| **加购** | √ | √ |  | √ |  |  |  | 件数/金额 |

1）建表语句

hive (gmall)>

drop table if exists dwd\_fact\_cart\_info;

create external table dwd\_fact\_cart\_info(

`id` string COMMENT '编号',

`user\_id` string COMMENT '用户id',

`sku\_id` string COMMENT 'skuid',

`cart\_price` string COMMENT '放入购物车时价格',

`sku\_num` string COMMENT '数量',

`sku\_name` string COMMENT 'sku名称 (冗余)',

`create\_time` string COMMENT '创建时间',

`operate\_time` string COMMENT '修改时间',

`is\_ordered` string COMMENT '是否已经下单。1为已下单;0为未下单',

`order\_time` string COMMENT '下单时间'

) COMMENT '加购事实表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

stored as parquet

location '/warehouse/gmall/dwd/dwd\_fact\_cart\_info/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwd\_fact\_cart\_info partition(dt='2020-03-10')

select

id,

user\_id,

sku\_id,

cart\_price,

sku\_num,

sku\_name,

create\_time,

operate\_time,

is\_ordered,

order\_time

from ods\_cart\_info

where dt='2020-03-10';

3）查询加载结果

hive (gmall)> select \* from dwd\_fact\_cart\_info where dt='2020-03-10';

### 4.4.11 收藏事实表（周期型快照事实表，每日快照）

收藏的标记，是否取消，会发生变化，做增量不合适。

每天做一次快照，导入的数据是全量，区别于事务型事实表是每天导入新增。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **时间** | **用户** | **地区** | **商品** | **优惠券** | **活动** | **编码** | **度量值** |
| **收藏** | √ | √ |  | √ |  |  |  | 个数 |

1）建表语句

hive (gmall)>

drop table if exists dwd\_fact\_favor\_info;

create external table dwd\_fact\_favor\_info(

`id` string COMMENT '编号',

`user\_id` string COMMENT '用户id',

`sku\_id` string COMMENT 'skuid',

`spu\_id` string COMMENT 'spuid',

`is\_cancel` string COMMENT '是否取消',

`create\_time` string COMMENT '收藏时间',

`cancel\_time` string COMMENT '取消时间'

) COMMENT '收藏事实表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

stored as parquet

location '/warehouse/gmall/dwd/dwd\_fact\_favor\_info/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwd\_fact\_favor\_info partition(dt='2020-03-10')

select

id,

user\_id,

sku\_id,

spu\_id,

is\_cancel,

create\_time,

cancel\_time

from ods\_favor\_info

where dt='2020-03-10';

3）查询加载结果

hive (gmall)> select \* from dwd\_fact\_favor\_info where dt='2020-03-10';

### 4.4.12 优惠券领用事实表（累积型快照事实表）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **时间** | **用户** | **地区** | **商品** | **优惠券** | **活动** | **编码** | **度量值** |
| **优惠券领用** | √ | √ |  |  | √ |  |  | 时间 |

优惠卷的生命周期：领取优惠卷-》用优惠卷下单-》优惠卷参与支付

累积型快照事实表使用：统计优惠卷领取次数、优惠卷下单次数、优惠卷参与支付次数

1）建表语句

hive (gmall)>

drop table if exists dwd\_fact\_coupon\_use;

create external table dwd\_fact\_coupon\_use(

`id` string COMMENT '编号',

`coupon\_id` string COMMENT '优惠券ID',

`user\_id` string COMMENT 'userid',

`order\_id` string COMMENT '订单id',

`coupon\_status` string COMMENT '优惠券状态',

`get\_time` string COMMENT '领取时间',

`using\_time` string COMMENT '使用时间(下单)',

`used\_time` string COMMENT '使用时间(支付)'

) COMMENT '优惠券领用事实表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

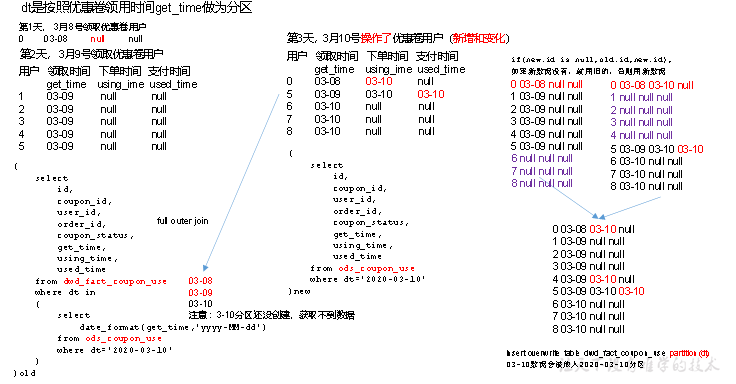
stored as parquet

location '/warehouse/gmall/dwd/dwd\_fact\_coupon\_use/'

tblproperties ("parquet.compression"="lzo");

注意：dt是按照优惠卷领用时间get\_time做为分区。

2）数据装载



hive (gmall)>

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

insert overwrite table dwd\_fact\_coupon\_use partition(dt)

select

if(new.id is null,old.id,new.id),

if(new.coupon\_id is null,old.coupon\_id,new.coupon\_id),

if(new.user\_id is null,old.user\_id,new.user\_id),

if(new.order\_id is null,old.order\_id,new.order\_id),

if(new.coupon\_status is null,old.coupon\_status,new.coupon\_status),

if(new.get\_time is null,old.get\_time,new.get\_time),

if(new.using\_time is null,old.using\_time,new.using\_time),

if(new.used\_time is null,old.used\_time,new.used\_time),

date\_format(if(new.get\_time is null,old.get\_time,new.get\_time),'yyyy-MM-dd')

from

(

select

id,

coupon\_id,

user\_id,

order\_id,

coupon\_status,

get\_time,

using\_time,

used\_time

from dwd\_fact\_coupon\_use

where dt in

(

select

date\_format(get\_time,'yyyy-MM-dd')

from ods\_coupon\_use

where dt='2020-03-10'

)

)old

full outer join

(

select

id,

coupon\_id,

user\_id,

order\_id,

coupon\_status,

get\_time,

using\_time,

used\_time

from ods\_coupon\_use

where dt='2020-03-10'

)new

on old.id=new.id;

3）查询加载结果

hive (gmall)> select \* from dwd\_fact\_coupon\_use where dt='2020-03-10';

### 4.4.13 订单事实表（累积型快照事实表）

1）concat函数

concat函数在连接字符串的时候，只要其中一个是NULL，那么将返回NULL

hive> select concat('a','b');

ab

hive> select concat('a','b',null);

NULL

2）concat\_ws函数

concat\_ws函数在连接字符串的时候，只要有一个字符串不是NULL，就不会返回NULL。concat\_ws函数需要指定分隔符。

hive> select concat\_ws('-','a','b');

a-b

hive> select concat\_ws('-','a','b',null);

a-b

hive> select concat\_ws('','a','b',null);

ab

3）STR\_TO\_MAP函数

（1）语法描述

STR\_TO\_MAP(VARCHAR text, VARCHAR listDelimiter, VARCHAR keyValueDelimiter)

（2）功能描述

使用listDelimiter将text分隔成K-V对，然后使用keyValueDelimiter分隔每个K-V对，组装成MAP返回。默认listDelimiter为（ ，），keyValueDelimiter为（=）。

（3）案例

str\_to\_map('1001=2020-03-10,1002=2020-03-10', ',' , '=')

输出

{"1001":"2020-03-10","1002":"2020-03-10"}

4）建表语句

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **时间** | **用户** | **地区** | **商品** | **优惠券** | **活动** | **编码** | **度量值** |
| **订单** | √ | √ | √ |  |  | √ |  | 件数/金额 |

订单生命周期：创建时间=》支付时间=》取消时间=》完成时间=》退款时间=》退款完成时间。

由于ODS层订单表只有创建时间和操作时间两个状态，不能表达所有时间含义，所以需要关联订单状态表。订单事实表里面增加了活动id，所以需要关联活动订单表。

hive (gmall)>

drop table if exists dwd\_fact\_order\_info;

create external table dwd\_fact\_order\_info (

`id` string COMMENT '订单编号',

`order\_status` string COMMENT '订单状态',

`user\_id` string COMMENT '用户id',

`out\_trade\_no` string COMMENT '支付流水号',

`create\_time` string COMMENT '创建时间(未支付状态)',

`payment\_time` string COMMENT '支付时间(已支付状态)',

`cancel\_time` string COMMENT '取消时间(已取消状态)',

`finish\_time` string COMMENT '完成时间(已完成状态)',

`refund\_time` string COMMENT '退款时间(退款中状态)',

`refund\_finish\_time` string COMMENT '退款完成时间(退款完成状态)',

`province\_id` string COMMENT '省份ID',

`activity\_id` string COMMENT '活动ID',

`original\_total\_amount` string COMMENT '原价金额',

`benefit\_reduce\_amount` string COMMENT '优惠金额',

`feight\_fee` string COMMENT '运费',

`final\_total\_amount` decimal(10,2) COMMENT '订单金额'

)

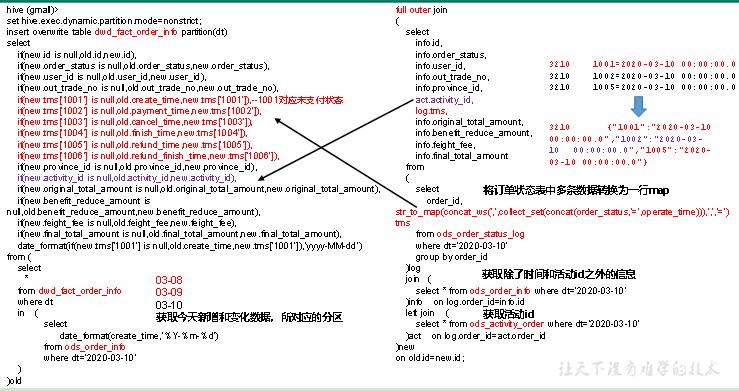
PARTITIONED BY (`dt` string)

stored as parquet

location '/warehouse/gmall/dwd/dwd\_fact\_order\_info/'

tblproperties ("parquet.compression"="lzo");

5）数据装载



5）常用函数

hive (gmall)> select order\_id, concat(order\_status,'=', operate\_time) from ods\_order\_status\_log where dt='2020-03-10';

3210 1001=2020-03-10 00:00:00.0

3211 1001=2020-03-10 00:00:00.0

3212 1001=2020-03-10 00:00:00.0

3210 1002=2020-03-10 00:00:00.0

3211 1002=2020-03-10 00:00:00.0

3212 1002=2020-03-10 00:00:00.0

3210 1005=2020-03-10 00:00:00.0

3211 1004=2020-03-10 00:00:00.0

3212 1004=2020-03-10 00:00:00.0

hive (gmall)> select order\_id, collect\_set(concat(order\_status,'=',operate\_time)) from ods\_order\_status\_log where dt='2020-03-10' group by order\_id;

3210 ["1001=2020-03-10 00:00:00.0","1002=2020-03-10 00:00:00.0","1005=2020-03-10 00:00:00.0"]

3211 ["1001=2020-03-10 00:00:00.0","1002=2020-03-10 00:00:00.0","1004=2020-03-10 00:00:00.0"]

3212 ["1001=2020-03-10 00:00:00.0","1002=2020-03-10 00:00:00.0","1004=2020-03-10 00:00:00.0"]

hive (gmall)>

select order\_id, concat\_ws(',', collect\_set(concat(order\_status,'=',operate\_time))) from ods\_order\_status\_log where dt='2020-03-10' group by order\_id;

3210 1001=2020-03-10 00:00:00.0,1002=2020-03-10 00:00:00.0,1005=2020-03-10 00:00:00.0

3211 1001=2020-03-10 00:00:00.0,1002=2020-03-10 00:00:00.0,1004=2020-03-10 00:00:00.0

3212 1001=2020-03-10 00:00:00.0,1002=2020-03-10 00:00:00.0,1004=2020-03-10 00:00:00.0

hive (gmall)>

select order\_id, str\_to\_map(concat\_ws(',',collect\_set(concat(order\_status,'=',operate\_time))), ',' , '=') from ods\_order\_status\_log where dt='2020-03-10' group by order\_id;

3210 {"1001":"2020-03-10 00:00:00.0","1002":"2020-03-10 00:00:00.0","1005":"2020-03-10 00:00:00.0"}

3211 {"1001":"2020-03-10 00:00:00.0","1002":"2020-03-10 00:00:00.0","1004":"2020-03-10 00:00:00.0"}

3212 {"1001":"2020-03-10 00:00:00.0","1002":"2020-03-10 00:00:00.0","1004":"2020-03-10 00:00:00.0"}

6）数据装载

hive (gmall)>

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

insert overwrite table dwd\_fact\_order\_info partition(dt)

select

if(new.id is null,old.id,new.id),

if(new.order\_status is null,old.order\_status,new.order\_status),

if(new.user\_id is null,old.user\_id,new.user\_id),

if(new.out\_trade\_no is null,old.out\_trade\_no,new.out\_trade\_no),

if(new.tms['1001'] is null,old.create\_time,new.tms['1001']),--1001对应未支付状态

if(new.tms['1002'] is null,old.payment\_time,new.tms['1002']),

if(new.tms['1003'] is null,old.cancel\_time,new.tms['1003']),

if(new.tms['1004'] is null,old.finish\_time,new.tms['1004']),

if(new.tms['1005'] is null,old.refund\_time,new.tms['1005']),

if(new.tms['1006'] is null,old.refund\_finish\_time,new.tms['1006']),

if(new.province\_id is null,old.province\_id,new.province\_id),

if(new.activity\_id is null,old.activity\_id,new.activity\_id),

if(new.original\_total\_amount is null,old.original\_total\_amount,new.original\_total\_amount),

if(new.benefit\_reduce\_amount is null,old.benefit\_reduce\_amount,new.benefit\_reduce\_amount),

if(new.feight\_fee is null,old.feight\_fee,new.feight\_fee),

if(new.final\_total\_amount is null,old.final\_total\_amount,new.final\_total\_amount),

date\_format(if(new.tms['1001'] is null,old.create\_time,new.tms['1001']),'yyyy-MM-dd')

from

(

select

id,

order\_status,

user\_id,

out\_trade\_no,

create\_time,

payment\_time,

cancel\_time,

finish\_time,

refund\_time,

refund\_finish\_time,

province\_id,

activity\_id,

original\_total\_amount,

benefit\_reduce\_amount,

feight\_fee,

final\_total\_amount

from dwd\_fact\_order\_info

where dt

in

(

select

date\_format(create\_time,'yyyy-MM-dd')

from ods\_order\_info

where dt='2020-03-10'

)

)old

full outer join

(

select

info.id,

info.order\_status,

info.user\_id,

info.out\_trade\_no,

info.province\_id,

act.activity\_id,

log.tms,

info.original\_total\_amount,

info.benefit\_reduce\_amount,

info.feight\_fee,

info.final\_total\_amount

from

(

select

order\_id,

str\_to\_map(concat\_ws(',',collect\_set(concat(order\_status,'=',operate\_time))),',','=') tms

from ods\_order\_status\_log

where dt='2020-03-10'

group by order\_id

)log

join

(

select \* from ods\_order\_info where dt='2020-03-10'

)info

on log.order\_id=info.id

left join

(

select \* from ods\_activity\_order where dt='2020-03-10'

)act

on log.order\_id=act.order\_id

)new

on old.id=new.id;

6）查询加载结果

hive (gmall)> select \* from dwd\_fact\_order\_info where dt='2020-03-10';

### 4.4.14 用户维度表（拉链表）

用户表中的数据每日既有可能新增，也有可能修改，但修改频率并不高，属于缓慢变化维度，此处采用拉链表存储用户维度数据。

**1）什么是拉链表**

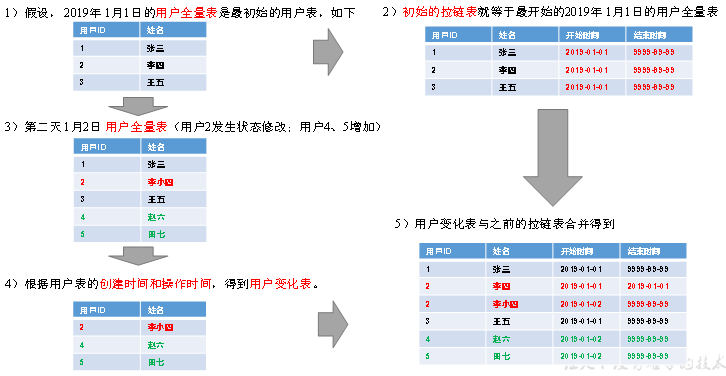


**2）为什么要做拉链表**

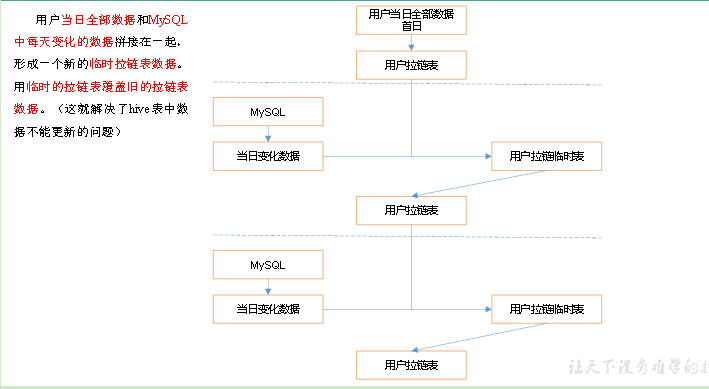




**3）拉链表形成过程**



**4）拉链表制作过程图**



**5）拉链表制作过程**

**步骤0：初始化拉链表（首次独立执行）**

（1）建立拉链表

hive (gmall)>

drop table if exists dwd\_dim\_user\_info\_his;

create external table dwd\_dim\_user\_info\_his(

`id` string COMMENT '用户id',

`name` string COMMENT '姓名',

`birthday` string COMMENT '生日',

`gender` string COMMENT '性别',

`email` string COMMENT '邮箱',

`user\_level` string COMMENT '用户等级',

`create\_time` string COMMENT '创建时间',

`operate\_time` string COMMENT '操作时间',

`start\_date` string COMMENT '有效开始日期',

`end\_date` string COMMENT '有效结束日期'

) COMMENT '订单拉链表'

stored as parquet

location '/warehouse/gmall/dwd/dwd\_dim\_user\_info\_his/'

tblproperties ("parquet.compression"="lzo");

（2）初始化拉链表

hive (gmall)>

insert overwrite table dwd\_dim\_user\_info\_his

select

id,

name,

birthday,

gender,

email,

user\_level,

create\_time,

operate\_time,

'2020-03-10',

'9999-99-99'

from ods\_user\_info oi

where oi.dt='2020-03-10';

**步骤1：制作当日变动数据（包括新增，修改）每日执行**

（1）如何获得每日变动表

a.最好表内有创建时间和变动时间（Lucky!）

b.如果没有，可以利用第三方工具监控比如canal，监控MySQL的实时变化进行记录（麻烦）。

c.逐行对比前后两天的数据，检查md5(concat(全部有可能变化的字段))是否相同(low)

d.要求业务数据库提供变动流水（人品，颜值）

（2）因为ods\_order\_info本身导入过来就是新增变动明细的表，所以不用处理

a）数据库中新增2020-03-11一天的数据

b）通过Sqoop把2020-03-11日所有数据导入

mysqlTohdfs.sh all 2020-03-11

c）ods层数据导入

hdfs\_to\_ods\_db.sh all 2020-03-11

**步骤2：先合并变动信息，再追加新增信息，插入到临时表中**

1）建立临时表

hive (gmall)>

drop table if exists dwd\_dim\_user\_info\_his\_tmp;

create external table dwd\_dim\_user\_info\_his\_tmp(

`id` string COMMENT '用户id',

`name` string COMMENT '姓名',

`birthday` string COMMENT '生日',

`gender` string COMMENT '性别',

`email` string COMMENT '邮箱',

`user\_level` string COMMENT '用户等级',

`create\_time` string COMMENT '创建时间',

`operate\_time` string COMMENT '操作时间',

`start\_date` string COMMENT '有效开始日期',

`end\_date` string COMMENT '有效结束日期'

) COMMENT '订单拉链临时表'

stored as parquet

location '/warehouse/gmall/dwd/dwd\_dim\_user\_info\_his\_tmp/'

tblproperties ("parquet.compression"="lzo");

2）导入脚本

hive (gmall)>

insert overwrite table dwd\_dim\_user\_info\_his\_tmp

select \* from

(

select

id,

name,

birthday,

gender,

email,

user\_level,

create\_time,

operate\_time,

'2020-03-11' start\_date,

'9999-99-99' end\_date

from ods\_user\_info where dt='2020-03-11'

union all

select

uh.id,

uh.name,

uh.birthday,

uh.gender,

uh.email,

uh.user\_level,

uh.create\_time,

uh.operate\_time,

uh.start\_date,

if(ui.id is not null and uh.end\_date='9999-99-99', date\_add(ui.dt,-1), uh.end\_date) end\_date

from dwd\_dim\_user\_info\_his uh left join

(

select

\*

from ods\_user\_info

where dt='2020-03-11'

) ui on uh.id=ui.id

)his

order by his.id, start\_date;

**步骤3：把临时表覆盖给拉链表**

1）导入数据

hive (gmall)>

insert overwrite table dwd\_dim\_user\_info\_his

select \* from dwd\_dim\_user\_info\_his\_tmp;

2）查询导入数据

hive (gmall)> select id, start\_date, end\_date from dwd\_dim\_user\_info\_his;

### 4.4.15 DWD层数据导入脚本

1）在/home/woaini/bin目录下创建脚本ods\_to\_dwd\_db.sh

[woaini@hadoop102 bin]$ vim ods\_to\_dwd\_db.sh

在脚本中填写如下内容

#!/bin/bash

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 "$2" ] ;then

do\_date=$2

else

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

fi

sql1="

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

insert overwrite table ${APP}.dwd\_dim\_sku\_info partition(dt='$do\_date')

select

sku.id,

sku.spu\_id,

sku.price,

sku.sku\_name,

sku.sku\_desc,

sku.weight,

sku.tm\_id,

ob.tm\_name,

sku.category3\_id,

c2.id category2\_id,

c1.id category1\_id,

c3.name category3\_name,

c2.name category2\_name,

c1.name category1\_name,

spu.spu\_name,

sku.create\_time

from

(

select \* from ${APP}.ods\_sku\_info where dt='$do\_date'

)sku

join

(

select \* from ${APP}.ods\_base\_trademark where dt='$do\_date'

)ob on sku.tm\_id=ob.tm\_id

join

(

select \* from ${APP}.ods\_spu\_info where dt='$do\_date'

)spu on spu.id = sku.spu\_id

join

(

select \* from ${APP}.ods\_base\_category3 where dt='$do\_date'

)c3 on sku.category3\_id=c3.id

join

(

select \* from ${APP}.ods\_base\_category2 where dt='$do\_date'

)c2 on c3.category2\_id=c2.id

join

(

select \* from ${APP}.ods\_base\_category1 where dt='$do\_date'

)c1 on c2.category1\_id=c1.id;

insert overwrite table ${APP}.dwd\_dim\_coupon\_info partition(dt='$do\_date')

select

id,

coupon\_name,

coupon\_type,

condition\_amount,

condition\_num,

activity\_id,

benefit\_amount,

benefit\_discount,

create\_time,

range\_type,

spu\_id,

tm\_id,

category3\_id,

limit\_num,

operate\_time,

expire\_time

from ${APP}.ods\_coupon\_info

where dt='$do\_date';

insert overwrite table ${APP}.dwd\_dim\_activity\_info partition(dt='$do\_date')

select

info.id,

info.activity\_name,

info.activity\_type,

rule.condition\_amount,

rule.condition\_num,

rule.benefit\_amount,

rule.benefit\_discount,

rule.benefit\_level,

info.start\_time,

info.end\_time,

info.create\_time

from

(

select \* from ${APP}.ods\_activity\_info where dt='$do\_date'

)info

left join

(

select \* from ${APP}.ods\_activity\_rule where dt='$do\_date'

)rule on info.id = rule.activity\_id;

insert overwrite table ${APP}.dwd\_fact\_order\_detail partition(dt='$do\_date')

select

od.id,

od.order\_id,

od.user\_id,

od.sku\_id,

od.sku\_name,

od.order\_price,

od.sku\_num,

od.create\_time,

oi.province\_id,

od.order\_price\*od.sku\_num

from

(

select \* from ${APP}.ods\_order\_detail where dt='$do\_date'

) od

join

(

select \* from ${APP}.ods\_order\_info where dt='$do\_date'

) oi

on od.order\_id=oi.id;

insert overwrite table ${APP}.dwd\_fact\_payment\_info partition(dt='$do\_date')

select

pi.id,

pi.out\_trade\_no,

pi.order\_id,

pi.user\_id,

pi.alipay\_trade\_no,

pi.total\_amount,

pi.subject,

pi.payment\_type,

pi.payment\_time,

oi.province\_id

from

(

select \* from ${APP}.ods\_payment\_info where dt='$do\_date'

)pi

join

(

select id, province\_id from ${APP}.ods\_order\_info where dt='$do\_date'

)oi

on pi.order\_id = oi.id;

insert overwrite table ${APP}.dwd\_fact\_order\_refund\_info partition(dt='$do\_date')

select

id,

user\_id,

order\_id,

sku\_id,

refund\_type,

refund\_num,

refund\_amount,

refund\_reason\_type,

create\_time

from ${APP}.ods\_order\_refund\_info

where dt='$do\_date';

insert overwrite table ${APP}.dwd\_fact\_comment\_info partition(dt='$do\_date')

select

id,

user\_id,

sku\_id,

spu\_id,

order\_id,

appraise,

create\_time

from ${APP}.ods\_comment\_info

where dt='$do\_date';

insert overwrite table ${APP}.dwd\_fact\_cart\_info partition(dt='$do\_date')

select

id,

user\_id,

sku\_id,

cart\_price,

sku\_num,

sku\_name,

create\_time,

operate\_time,

is\_ordered,

order\_time

from ${APP}.ods\_cart\_info

where dt='$do\_date';

insert overwrite table ${APP}.dwd\_fact\_favor\_info partition(dt='$do\_date')

select

id,

user\_id,

sku\_id,

spu\_id,

is\_cancel,

create\_time,

cancel\_time

from ${APP}.ods\_favor\_info

where dt='$do\_date';

insert overwrite table ${APP}.dwd\_fact\_coupon\_use partition(dt)

select

if(new.id is null,old.id,new.id),

if(new.coupon\_id is null,old.coupon\_id,new.coupon\_id),

if(new.user\_id is null,old.user\_id,new.user\_id),

if(new.order\_id is null,old.order\_id,new.order\_id),

if(new.coupon\_status is null,old.coupon\_status,new.coupon\_status),

if(new.get\_time is null,old.get\_time,new.get\_time),

if(new.using\_time is null,old.using\_time,new.using\_time),

if(new.used\_time is null,old.used\_time,new.used\_time),

date\_format(if(new.get\_time is null,old.get\_time,new.get\_time),'yyyy-MM-dd')

from

(

select

id,

coupon\_id,

user\_id,

order\_id,

coupon\_status,

get\_time,

using\_time,

used\_time

from ${APP}.dwd\_fact\_coupon\_use

where dt in

(

select

date\_format(get\_time,'yyyy-MM-dd')

from ${APP}.ods\_coupon\_use

where dt='$do\_date'

)

)old

full outer join

(

select

id,

coupon\_id,

user\_id,

order\_id,

coupon\_status,

get\_time,

using\_time,

used\_time

from ${APP}.ods\_coupon\_use

where dt='$do\_date'

)new

on old.id=new.id;

insert overwrite table ${APP}.dwd\_fact\_order\_info partition(dt)

select

if(new.id is null,old.id,new.id),

if(new.order\_status is null,old.order\_status,new.order\_status),

if(new.user\_id is null,old.user\_id,new.user\_id),

if(new.out\_trade\_no is null,old.out\_trade\_no,new.out\_trade\_no),

if(new.tms['1001'] is null,old.create\_time,new.tms['1001']),--1001对应未支付状态

if(new.tms['1002'] is null,old.payment\_time,new.tms['1002']),

if(new.tms['1003'] is null,old.cancel\_time,new.tms['1003']),

if(new.tms['1004'] is null,old.finish\_time,new.tms['1004']),

if(new.tms['1005'] is null,old.refund\_time,new.tms['1005']),

if(new.tms['1006'] is null,old.refund\_finish\_time,new.tms['1006']),

if(new.province\_id is null,old.province\_id,new.province\_id),

if(new.activity\_id is null,old.activity\_id,new.activity\_id),

if(new.original\_total\_amount is null,old.original\_total\_amount,new.original\_total\_amount),

if(new.benefit\_reduce\_amount is null,old.benefit\_reduce\_amount,new.benefit\_reduce\_amount),

if(new.feight\_fee is null,old.feight\_fee,new.feight\_fee),

if(new.final\_total\_amount is null,old.final\_total\_amount,new.final\_total\_amount),

date\_format(if(new.tms['1001'] is null,old.create\_time,new.tms['1001']),'yyyy-MM-dd')

from

(

select

id,

order\_status,

user\_id,

out\_trade\_no,

create\_time,

payment\_time,

cancel\_time,

finish\_time,

refund\_time,

refund\_finish\_time,

province\_id,

activity\_id,

original\_total\_amount,

benefit\_reduce\_amount,

feight\_fee,

final\_total\_amount

from ${APP}.dwd\_fact\_order\_info

where dt

in

(

select

date\_format(create\_time,'yyyy-MM-dd')

from ${APP}.ods\_order\_info

where dt='$do\_date'

)

)old

full outer join

(

select

info.id,

info.order\_status,

info.user\_id,

info.out\_trade\_no,

info.province\_id,

act.activity\_id,

log.tms,

info.original\_total\_amount,

info.benefit\_reduce\_amount,

info.feight\_fee,

info.final\_total\_amount

from

(

select

order\_id,

str\_to\_map(concat\_ws(',',collect\_set(concat(order\_status,'=',operate\_time))),',','=') tms

from ${APP}.ods\_order\_status\_log

where dt='$do\_date'

group by order\_id

)log

join

(

select \* from ${APP}.ods\_order\_info where dt='$do\_date'

)info

on log.order\_id=info.id

left join

(

select \* from ${APP}.ods\_activity\_order where dt='$do\_date'

)act

on log.order\_id=act.order\_id

)new

on old.id=new.id;

insert overwrite table ${APP}.dwd\_dim\_user\_info\_his\_tmp

select \* from

(

select

id,

name,

birthday,

gender,

email,

user\_level,

create\_time,

operate\_time,

'2020-03-11' start\_date,

'9999-99-99' end\_date

from ${APP}.ods\_user\_info where dt='$do\_date'

union all

select

uh.id,

uh.name,

uh.birthday,

uh.gender,

uh.email,

uh.user\_level,

uh.create\_time,

uh.operate\_time,

uh.start\_date,

if(ui.id is not null and uh.end\_date='9999-99-99', date\_add(ui.dt,-1), uh.end\_date) end\_date

from ${APP}.dwd\_dim\_user\_info\_his uh left join

(

select

\*

from ${APP}.ods\_user\_info

where dt='$do\_date'

) ui on uh.id=ui.id

)his

order by his.id, start\_date;

insert overwrite table ${APP}.dwd\_dim\_user\_info\_his select \* from ${APP}.dwd\_dim\_user\_info\_his\_tmp;

"

sql2="

insert overwrite table ${APP}.dwd\_dim\_base\_province

select

bp.id,

bp.name,

bp.area\_code,

bp.iso\_code,

bp.region\_id,

br.region\_name

from ${APP}.ods\_base\_province bp

join ${APP}.ods\_base\_region br

on bp.region\_id=br.id;

"

case $1 in

"first"){

$hive -e "$sql1"

$hive -e "$sql2"

};;

"all"){

$hive -e "$sql1"

};;

esac

2）增加脚本执行权限

[woaini@hadoop102 bin]$ chmod 777 ods\_to\_dwd\_db.sh

3）执行脚本导入数据

[woaini@hadoop102 bin]$ ods\_to\_dwd\_db.sh all 2020-03-11

4）查看导入数据

hive (gmall)>

select \* from dwd\_fact\_order\_info where dt='2020-03-11';

select \* from dwd\_fact\_order\_detail where dt='2020-03-11';

select \* from dwd\_fact\_comment\_info where dt='2020-03-11';

select \* from dwd\_fact\_order\_refund\_info where dt='2020-03-11';

# 第5章 数仓搭建-DWS层

## 5.1 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。

**2）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。

**3）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。

**4）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。

**5）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。

**6）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。

**7）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。

**8）本周回流用户**

上周未启动过应用，本周启动了应用的用户。

**9）连续n周活跃用户**

连续n周，每周至少启动一次。

**10）忠诚用户**

连续活跃5周以上的用户

**11）连续活跃用户**

连续2周及以上活跃的用户

**12）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）

**13）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%。

**14）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。

**15）单次使用时长**

每次启动使用的时间长度。

**16）日使用时长**

累计一天内的使用时间长度。

**17）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。

## 5.2 系统函数

### 5.2.1 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

### 5.2.2 nvl函数

1）基本语法

NVL（表达式1，表达式2）

如果表达式1为空值，NVL返回值为表达式2的值，否则返回表达式1的值。 该函数的目的是把一个空值（null）转换成一个实际的值。其表达式的值可以是数字型、字符型和日期型。但是表达式1和表达式2的数据类型必须为同一个类型。

### 5.2.3 日期处理函数

1）date\_format函数（根据格式整理日期）

hive (gmall)> select date\_format('2020-03-10','yyyy-MM');

2020-03

2）date\_add函数（加减日期）

hive (gmall)> select date\_add('2020-03-10',-1);

2020-03-09

hive (gmall)> select date\_add('2020-03-10',1);

2020-03-11

3）next\_day函数

（1）取当前天的下一个周一

hive (gmall)> select next\_day('2020-03-12','MO');

2020-03-16

说明：星期一到星期日的英文（Monday，Tuesday、Wednesday、Thursday、Friday、Saturday、Sunday）

（2）取当前周的周一

hive (gmall)> select date\_add(next\_day('2020-03-12','MO'),-7);

2020-03-11

4）last\_day函数（求当月最后一天日期）

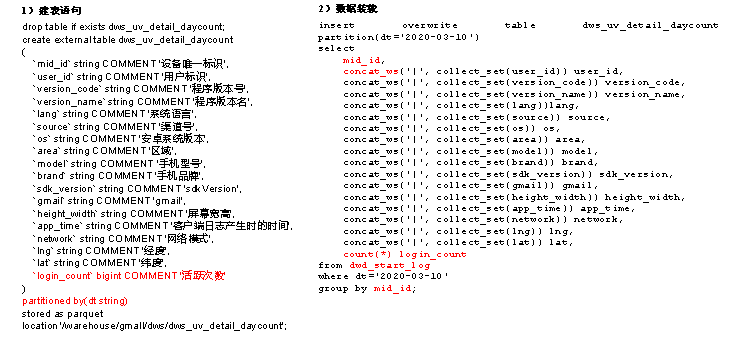
hive (gmall)> select last\_day('2020-03-10');

2020-03-31

## 5.3 DWS层（用户行为）

### 5.3.1 每日设备行为

每日设备行为，主要按照设备id统计。



1）建表语句

hive (gmall)>

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

create external table dws\_uv\_detail\_daycount

(

`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',

`login\_count` bigint COMMENT '活跃次数'

)

partitioned by(dt string)

stored as parquet

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

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dws\_uv\_detail\_daycount partition(dt='2020-03-10')

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,

count(\*) login\_count

from dwd\_start\_log

where dt='2020-03-10'

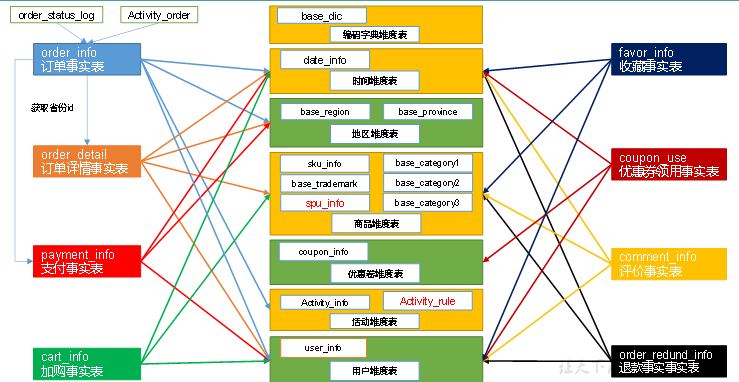
group by mid\_id;

3）查询加载结果

hive (gmall)> select \* from dws\_uv\_detail\_daycount where dt='2020-03-10';

## 5.4 DWS层（业务）

DWS层的宽表字段，是站在不同维度的视角去看事实表。重点关注事实表的度量值。



### 5.4.1 每日会员行为

1）建表语句

hive (gmall)>

drop table if exists dws\_user\_action\_daycount;

create external table dws\_user\_action\_daycount

(

user\_id string comment '用户 id',

login\_count bigint comment '登录次数',

cart\_count bigint comment '加入购物车次数',

cart\_amount double comment '加入购物车金额',

order\_count bigint comment '下单次数',

order\_amount decimal(16,2) comment '下单金额',

payment\_count bigint comment '支付次数',

payment\_amount decimal(16,2) comment '支付金额'

) COMMENT '每日用户行为'

PARTITIONED BY (`dt` string)

stored as parquet

location '/warehouse/gmall/dws/dws\_user\_action\_daycount/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

with

tmp\_login as

(

select

user\_id,

count(\*) login\_count

from dwd\_start\_log

where dt='2020-03-10'

and user\_id is not null

group by user\_id

),

tmp\_cart as

(

select

user\_id,

count(\*) cart\_count,

sum(cart\_price\*sku\_num) cart\_amount

from dwd\_fact\_cart\_info

where dt='2020-03-10'

and user\_id is not null

and date\_format(create\_time,'yyyy-MM-dd')='2020-03-10'

group by user\_id

),

tmp\_order as

(

select

user\_id,

count(\*) order\_count,

sum(final\_total\_amount) order\_amount

from dwd\_fact\_order\_info

where dt='2020-03-10'

group by user\_id

) ,

tmp\_payment as

(

select

user\_id,

count(\*) payment\_count,

sum(payment\_amount) payment\_amount

from dwd\_fact\_payment\_info

where dt='2020-03-10'

group by user\_id

)

insert overwrite table dws\_user\_action\_daycount partition(dt='2020-03-10')

select

user\_actions.user\_id,

sum(user\_actions.login\_count),

sum(user\_actions.cart\_count),

sum(user\_actions.cart\_amount),

sum(user\_actions.order\_count),

sum(user\_actions.order\_amount),

sum(user\_actions.payment\_count),

sum(user\_actions.payment\_amount)

from

(

select

user\_id,

login\_count,

0 cart\_count,

0 cart\_amount,

0 order\_count,

0 order\_amount,

0 payment\_count,

0 payment\_amount

from

tmp\_login

union all

select

user\_id,

0 login\_count,

cart\_count,

cart\_amount,

0 order\_count,

0 order\_amount,

0 payment\_count,

0 payment\_amount

from

tmp\_cart

union all

select

user\_id,

0 login\_count,

0 cart\_count,

0 cart\_amount,

order\_count,

order\_amount,

0 payment\_count,

0 payment\_amount

from tmp\_order

union all

select

user\_id,

0 login\_count,

0 cart\_count,

0 cart\_amount,

0 order\_count,

0 order\_amount,

payment\_count,

payment\_amount

from tmp\_payment

) user\_actions

group by user\_id;

3）查询加载结果

hive (gmall)> select \* from dws\_user\_action\_daycount where dt='2020-03-10';

### 5.4.2 每日商品行为

1）建表语句

hive (gmall)>

drop table if exists dws\_sku\_action\_daycount;

create external table dws\_sku\_action\_daycount

(

sku\_id string comment 'sku\_id',

order\_count bigint comment '被下单次数',

order\_num bigint comment '被下单件数',

order\_amount decimal(16,2) comment '被下单金额',

payment\_count bigint comment '被支付次数',

payment\_num bigint comment '被支付件数',

payment\_amount decimal(16,2) comment '被支付金额',

refund\_count bigint comment '被退款次数',

refund\_num bigint comment '被退款件数',

refund\_amount decimal(16,2) comment '被退款金额',

cart\_count bigint comment '被加入购物车次数',

cart\_num bigint comment '被加入购物车件数',

favor\_count bigint comment '被收藏次数',

appraise\_good\_count bigint comment '好评数',

appraise\_mid\_count bigint comment '中评数',

appraise\_bad\_count bigint comment '差评数',

appraise\_default\_count bigint comment '默认评价数'

) COMMENT '每日商品行为'

PARTITIONED BY (`dt` string)

stored as parquet

location '/warehouse/gmall/dws/dws\_sku\_action\_daycount/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

注意：如果是23点59下单，支付日期跨天。需要从订单详情里面取出支付时间是今天，订单时间是昨天或者今天的订单。

hive (gmall)>

with

tmp\_order as

(

select

sku\_id,

count(\*) order\_count,

sum(sku\_num) order\_num,

sum(total\_amount) order\_amount

from dwd\_fact\_order\_detail

where dt='2020-03-10'

group by sku\_id

),

tmp\_payment as

(

select

sku\_id,

count(\*) payment\_count,

sum(sku\_num) payment\_num,

sum(total\_amount) payment\_amount

from dwd\_fact\_order\_detail

where dt='2020-03-10'

and order\_id in

(

select

id

from dwd\_fact\_order\_info

where (dt='2020-03-10' or dt=date\_add('2020-03-10',-1))

and date\_format(payment\_time,'yyyy-MM-dd')='2020-03-10'

)

group by sku\_id

),

tmp\_refund as

(

select

sku\_id,

count(\*) refund\_count,

sum(refund\_num) refund\_num,

sum(refund\_amount) refund\_amount

from dwd\_fact\_order\_refund\_info

where dt='2020-03-10'

group by sku\_id

),

tmp\_cart as

(

select

sku\_id,

count(\*) cart\_count,

sum(sku\_num) cart\_num

from dwd\_fact\_cart\_info

where dt='2020-03-10'

and date\_format(create\_time,'yyyy-MM-dd')='2020-03-10'

group by sku\_id

),

tmp\_favor as

(

select

sku\_id,

count(\*) favor\_count

from dwd\_fact\_favor\_info

where dt='2020-03-10'

and date\_format(create\_time,'yyyy-MM-dd')='2020-03-10'

group by sku\_id

),

tmp\_appraise as

(

select

sku\_id,

sum(if(appraise='1201',1,0)) appraise\_good\_count,

sum(if(appraise='1202',1,0)) appraise\_mid\_count,

sum(if(appraise='1203',1,0)) appraise\_bad\_count,

sum(if(appraise='1204',1,0)) appraise\_default\_count

from dwd\_fact\_comment\_info

where dt='2020-03-10'

group by sku\_id

)

insert overwrite table dws\_sku\_action\_daycount partition(dt='2020-03-10')

select

sku\_id,

sum(order\_count),

sum(order\_num),

sum(order\_amount),

sum(payment\_count),

sum(payment\_num),

sum(payment\_amount),

sum(refund\_count),

sum(refund\_num),

sum(refund\_amount),

sum(cart\_count),

sum(cart\_num),

sum(favor\_count),

sum(appraise\_good\_count),

sum(appraise\_mid\_count),

sum(appraise\_bad\_count),

sum(appraise\_default\_count)

from

(

select

sku\_id,

order\_count,

order\_num,

order\_amount,

0 payment\_count,

0 payment\_num,

0 payment\_amount,

0 refund\_count,

0 refund\_num,

0 refund\_amount,

0 cart\_count,

0 cart\_num,

0 favor\_count,

0 appraise\_good\_count,

0 appraise\_mid\_count,

0 appraise\_bad\_count,

0 appraise\_default\_count

from tmp\_order

union all

select

sku\_id,

0 order\_count,

0 order\_num,

0 order\_amount,

payment\_count,

payment\_num,

payment\_amount,

0 refund\_count,

0 refund\_num,

0 refund\_amount,

0 cart\_count,

0 cart\_num,

0 favor\_count,

0 appraise\_good\_count,

0 appraise\_mid\_count,

0 appraise\_bad\_count,

0 appraise\_default\_count

from tmp\_payment

union all

select

sku\_id,

0 order\_count,

0 order\_num,

0 order\_amount,

0 payment\_count,

0 payment\_num,

0 payment\_amount,

refund\_count,

refund\_num,

refund\_amount,

0 cart\_count,

0 cart\_num,

0 favor\_count,

0 appraise\_good\_count,

0 appraise\_mid\_count,

0 appraise\_bad\_count,

0 appraise\_default\_count

from tmp\_refund

union all

select

sku\_id,

0 order\_count,

0 order\_num,

0 order\_amount,

0 payment\_count,

0 payment\_num,

0 payment\_amount,

0 refund\_count,

0 refund\_num,

0 refund\_amount,

cart\_count,

cart\_num,

0 favor\_count,

0 appraise\_good\_count,

0 appraise\_mid\_count,

0 appraise\_bad\_count,

0 appraise\_default\_count

from tmp\_cart

union all

select

sku\_id,

0 order\_count,

0 order\_num,

0 order\_amount,

0 payment\_count,

0 payment\_num,

0 payment\_amount,

0 refund\_count,

0 refund\_num,

0 refund\_amount,

0 cart\_count,

0 cart\_num,

favor\_count,

0 appraise\_good\_count,

0 appraise\_mid\_count,

0 appraise\_bad\_count,

0 appraise\_default\_count

from tmp\_favor

union all

select

sku\_id,

0 order\_count,

0 order\_num,

0 order\_amount,

0 payment\_count,

0 payment\_num,

0 payment\_amount,

0 refund\_count,

0 refund\_num,

0 refund\_amount,

0 cart\_count,

0 cart\_num,

0 favor\_count,

appraise\_good\_count,

appraise\_mid\_count,

appraise\_bad\_count,

appraise\_default\_count

from tmp\_appraise

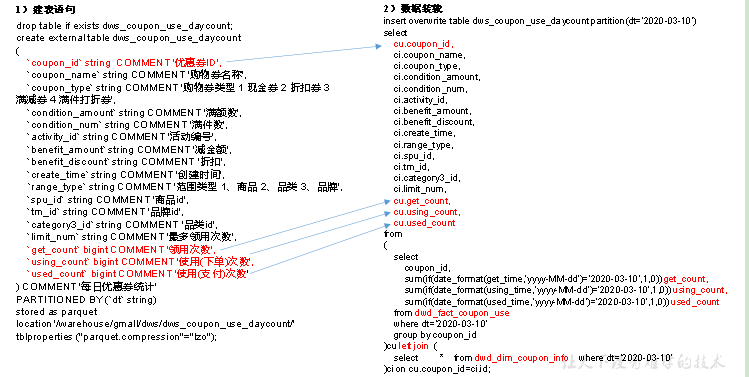
)tmp

group by sku\_id;

3）查询加载结果

hive (gmall)> select \* from dws\_sku\_action\_daycount where dt='2020-03-10';

### 5.4.3 每日优惠券统计（预留）



1）建表语句

hive (gmall)>

drop table if exists dws\_coupon\_use\_daycount;

create external table dws\_coupon\_use\_daycount

(

`coupon\_id` string COMMENT '优惠券ID',

`coupon\_name` string COMMENT '购物券名称',

`coupon\_type` string COMMENT '购物券类型 1 现金券 2 折扣券 3 满减券 4 满件打折券',

`condition\_amount` string COMMENT '满额数',

`condition\_num` string COMMENT '满件数',

`activity\_id` string COMMENT '活动编号',

`benefit\_amount` string COMMENT '减金额',

`benefit\_discount` string COMMENT '折扣',

`create\_time` string COMMENT '创建时间',

`range\_type` string COMMENT '范围类型 1、商品 2、品类 3、品牌',

`spu\_id` string COMMENT '商品id',

`tm\_id` string COMMENT '品牌id',

`category3\_id` string COMMENT '品类id',

`limit\_num` string COMMENT '最多领用次数',

`get\_count` bigint COMMENT '领用次数',

`using\_count` bigint COMMENT '使用(下单)次数',

`used\_count` bigint COMMENT '使用(支付)次数'

) COMMENT '每日优惠券统计'

PARTITIONED BY (`dt` string)

stored as parquet

location '/warehouse/gmall/dws/dws\_coupon\_use\_daycount/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dws\_coupon\_use\_daycount partition(dt='2020-03-10')

select

cu.coupon\_id,

ci.coupon\_name,

ci.coupon\_type,

ci.condition\_amount,

ci.condition\_num,

ci.activity\_id,

ci.benefit\_amount,

ci.benefit\_discount,

ci.create\_time,

ci.range\_type,

ci.spu\_id,

ci.tm\_id,

ci.category3\_id,

ci.limit\_num,

cu.get\_count,

cu.using\_count,

cu.used\_count

from

(

select

coupon\_id,

sum(if(date\_format(get\_time,'yyyy-MM-dd')='2020-03-10',1,0)) get\_count,

sum(if(date\_format(using\_time,'yyyy-MM-dd')='2020-03-10',1,0)) using\_count,

sum(if(date\_format(used\_time,'yyyy-MM-dd')='2020-03-10',1,0)) used\_count

from dwd\_fact\_coupon\_use

where dt='2020-03-10'

group by coupon\_id

)cu

left join

(

select

\*

from dwd\_dim\_coupon\_info

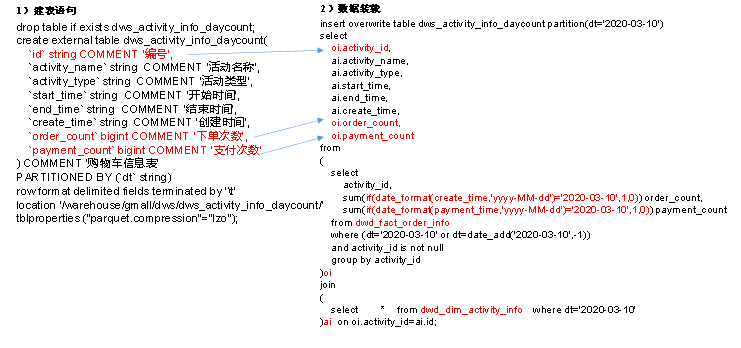
where dt='2020-03-10'

)ci on cu.coupon\_id=ci.id;

3）查询加载结果

hive (gmall)> select \* from dws\_coupon\_use\_daycount where dt='2020-03-10';

### 5.4.4 每日活动统计（预留）



1）建表语句

hive (gmall)>

drop table if exists dws\_activity\_info\_daycount;

create external table dws\_activity\_info\_daycount(

`id` string COMMENT '编号',

`activity\_name` string COMMENT '活动名称',

`activity\_type` string COMMENT '活动类型',

`start\_time` string COMMENT '开始时间',

`end\_time` string COMMENT '结束时间',

`create\_time` string COMMENT '创建时间',

`order\_count` bigint COMMENT '下单次数',

`payment\_count` bigint COMMENT '支付次数'

) COMMENT '活动信息表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

location '/warehouse/gmall/dws/dws\_activity\_info\_daycount/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dws\_activity\_info\_daycount partition(dt='2020-03-10')

select

oi.activity\_id,

ai.activity\_name,

ai.activity\_type,

ai.start\_time,

ai.end\_time,

ai.create\_time,

oi.order\_count,

oi.payment\_count

from

(

select

activity\_id,

sum(if(date\_format(create\_time,'yyyy-MM-dd')='2020-03-10',1,0)) order\_count,

sum(if(date\_format(payment\_time,'yyyy-MM-dd')='2020-03-10',1,0)) payment\_count

from dwd\_fact\_order\_info

where (dt='2020-03-10' or dt=date\_add('2020-03-10',-1))

and activity\_id is not null

group by activity\_id

)oi

join

(

select

\*

from dwd\_dim\_activity\_info

where dt='2020-03-10'

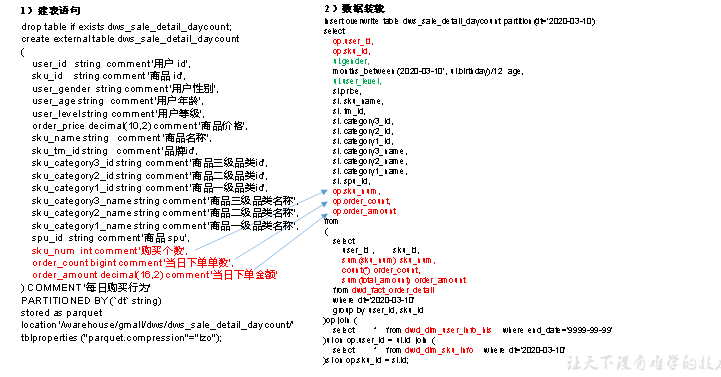
)ai

on oi.activity\_id=ai.id;

3）查询加载结果

hive (gmall)> select \* from dws\_activity\_info\_daycount where dt='2020-03-10';

### 5.4.5 每日购买行为



1）建表语句

hive (gmall)>

drop table if exists dws\_sale\_detail\_daycount;

create external table dws\_sale\_detail\_daycount

(

user\_id string comment '用户 id',

sku\_id string comment '商品 id',

user\_gender string comment '用户性别',

user\_age string comment '用户年龄',

user\_level string comment '用户等级',

order\_price decimal(10,2) comment '商品价格',

sku\_name string comment '商品名称',

sku\_tm\_id string comment '品牌id',

sku\_category3\_id string comment '商品三级品类id',

sku\_category2\_id string comment '商品二级品类id',

sku\_category1\_id string comment '商品一级品类id',

sku\_category3\_name string comment '商品三级品类名称',

sku\_category2\_name string comment '商品二级品类名称',

sku\_category1\_name string comment '商品一级品类名称',

spu\_id string comment '商品 spu',

sku\_num int comment '购买个数',

order\_count bigint comment '当日下单单数',

order\_amount decimal(16,2) comment '当日下单金额'

) COMMENT '每日购买行为'

PARTITIONED BY (`dt` string)

stored as parquet

location '/warehouse/gmall/dws/dws\_sale\_detail\_daycount/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dws\_sale\_detail\_daycount partition(dt='2020-03-10')

select

op.user\_id,

op.sku\_id,

ui.gender,

months\_between('2020-03-10', ui.birthday)/12 age,

ui.user\_level,

si.price,

si.sku\_name,

si.tm\_id,

si.category3\_id,

si.category2\_id,

si.category1\_id,

si.category3\_name,

si.category2\_name,

si.category1\_name,

si.spu\_id,

op.sku\_num,

op.order\_count,

op.order\_amount

from

(

select

user\_id,

sku\_id,

sum(sku\_num) sku\_num,

count(\*) order\_count,

sum(total\_amount) order\_amount

from dwd\_fact\_order\_detail

where dt='2020-03-10'

group by user\_id, sku\_id

)op

join

(

select

\*

from dwd\_dim\_user\_info\_his

where end\_date='9999-99-99'

)ui on op.user\_id = ui.id

join

(

select

\*

from dwd\_dim\_sku\_info

where dt='2020-03-10'

)si on op.sku\_id = si.id;

3）查询加载结果

hive (gmall)> select \* from dws\_sale\_detail\_daycount where dt='2020-03-10';

## 5.5 DWS层数据导入脚本

1）在/home/woaini/bin目录下创建脚本dwd\_to\_dws.sh

[woaini@hadoop102 bin]$ vim dwd\_to\_dws.sh

在脚本中填写如下内容

#!/bin/bash

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="

insert overwrite table ${APP}.dws\_uv\_detail\_daycount partition(dt='$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,

count(\*) login\_count

from ${APP}.dwd\_start\_log

where dt='$do\_date'

group by mid\_id;

with

tmp\_login as

(

select

user\_id,

count(\*) login\_count

from ${APP}.dwd\_start\_log

where dt='$do\_date'

and user\_id is not null

group by user\_id

),

tmp\_cart as

(

select

user\_id,

count(\*) cart\_count,

sum(cart\_price\*sku\_num) cart\_amount

from ${APP}.dwd\_fact\_cart\_info

where dt='$do\_date'

and user\_id is not null

and date\_format(create\_time,'yyyy-MM-dd')='$do\_date'

group by user\_id

),

tmp\_order as

(

select

user\_id,

count(\*) order\_count,

sum(final\_total\_amount) order\_amount

from ${APP}.dwd\_fact\_order\_info

where dt='$do\_date'

group by user\_id

) ,

tmp\_payment as

(

select

user\_id,

count(\*) payment\_count,

sum(payment\_amount) payment\_amount

from ${APP}.dwd\_fact\_payment\_info

where dt='$do\_date'

group by user\_id

)

insert overwrite table ${APP}.dws\_user\_action\_daycount partition(dt='$do\_date')

select

user\_actions.user\_id,

sum(user\_actions.login\_count),

sum(user\_actions.cart\_count),

sum(user\_actions.cart\_amount),

sum(user\_actions.order\_count),

sum(user\_actions.order\_amount),

sum(user\_actions.payment\_count),

sum(user\_actions.payment\_amount)

from

(

select

user\_id,

login\_count,

0 cart\_count,

0 cart\_amount,

0 order\_count,

0 order\_amount,

0 payment\_count,

0 payment\_amount

from

tmp\_login

union all

select

user\_id,

0 login\_count,

cart\_count,

cart\_amount,

0 order\_count,

0 order\_amount,

0 payment\_count,

0 payment\_amount

from

tmp\_cart

union all

select

user\_id,

0 login\_count,

0 cart\_count,

0 cart\_amount,

order\_count,

order\_amount,

0 payment\_count,

0 payment\_amount

from tmp\_order

union all

select

user\_id,

0 login\_count,

0 cart\_count,

0 cart\_amount,

0 order\_count,

0 order\_amount,

payment\_count,

payment\_amount

from tmp\_payment

) user\_actions

group by user\_id;

with

tmp\_order as

(

select

sku\_id,

count(\*) order\_count,

sum(sku\_num) order\_num,

sum(total\_amount) order\_amount

from ${APP}.dwd\_fact\_order\_detail

where dt='$do\_date'

group by sku\_id

),

tmp\_payment as

(

select

sku\_id,

count(\*) payment\_count,

sum(sku\_num) payment\_num,

sum(total\_amount) payment\_amount

from ${APP}.dwd\_fact\_order\_detail

where dt='$do\_date'

and order\_id in

(

select

id

from ${APP}.dwd\_fact\_order\_info

where (dt='$do\_date' or dt=date\_add('$do\_date',-1))

and date\_format(payment\_time,'yyyy-MM-dd')='$do\_date'

)

group by sku\_id

),

tmp\_refund as

(

select

sku\_id,

count(\*) refund\_count,

sum(refund\_num) refund\_num,

sum(refund\_amount) refund\_amount

from ${APP}.dwd\_fact\_order\_refund\_info

where dt='$do\_date'

group by sku\_id

),

tmp\_cart as

(

select

sku\_id,

count(\*) cart\_count,

sum(sku\_num) cart\_num

from ${APP}.dwd\_fact\_cart\_info

where dt='$do\_date'

and date\_format(create\_time,'yyyy-MM-dd')='$do\_date'

group by sku\_id

),

tmp\_favor as

(

select

sku\_id,

count(\*) favor\_count

from ${APP}.dwd\_fact\_favor\_info

where dt='$do\_date'

and date\_format(create\_time,'yyyy-MM-dd')='$do\_date'

group by sku\_id

),

tmp\_appraise as

(

select

sku\_id,

sum(if(appraise='1201',1,0)) appraise\_good\_count,

sum(if(appraise='1202',1,0)) appraise\_mid\_count,

sum(if(appraise='1203',1,0)) appraise\_bad\_count,

sum(if(appraise='1204',1,0)) appraise\_default\_count

from ${APP}.dwd\_fact\_comment\_info

where dt='$do\_date'

group by sku\_id

)

insert overwrite table ${APP}.dws\_sku\_action\_daycount partition(dt='$do\_date')

select

sku\_id,

sum(order\_count),

sum(order\_num),

sum(order\_amount),

sum(payment\_count),

sum(payment\_num),

sum(payment\_amount),

sum(refund\_count),

sum(refund\_num),

sum(refund\_amount),

sum(cart\_count),

sum(cart\_num),

sum(favor\_count),

sum(appraise\_good\_count),

sum(appraise\_mid\_count),

sum(appraise\_bad\_count),

sum(appraise\_default\_count)

from

(

select

sku\_id,

order\_count,

order\_num,

order\_amount,

0 payment\_count,

0 payment\_num,

0 payment\_amount,

0 refund\_count,

0 refund\_num,

0 refund\_amount,

0 cart\_count,

0 cart\_num,

0 favor\_count,

0 appraise\_good\_count,

0 appraise\_mid\_count,

0 appraise\_bad\_count,

0 appraise\_default\_count

from tmp\_order

union all

select

sku\_id,

0 order\_count,

0 order\_num,

0 order\_amount,

payment\_count,

payment\_num,

payment\_amount,

0 refund\_count,

0 refund\_num,

0 refund\_amount,

0 cart\_count,

0 cart\_num,

0 favor\_count,

0 appraise\_good\_count,

0 appraise\_mid\_count,

0 appraise\_bad\_count,

0 appraise\_default\_count

from tmp\_payment

union all

select

sku\_id,

0 order\_count,

0 order\_num,

0 order\_amount,

0 payment\_count,

0 payment\_num,

0 payment\_amount,

refund\_count,

refund\_num,

refund\_amount,

0 cart\_count,

0 cart\_num,

0 favor\_count,

0 appraise\_good\_count,

0 appraise\_mid\_count,

0 appraise\_bad\_count,

0 appraise\_default\_count

from tmp\_refund

union all

select

sku\_id,

0 order\_count,

0 order\_num,

0 order\_amount,

0 payment\_count,

0 payment\_num,

0 payment\_amount,

0 refund\_count,

0 refund\_num,

0 refund\_amount,

cart\_count,

cart\_num,

0 favor\_count,

0 appraise\_good\_count,

0 appraise\_mid\_count,

0 appraise\_bad\_count,

0 appraise\_default\_count

from tmp\_cart

union all

select

sku\_id,

0 order\_count,

0 order\_num,

0 order\_amount,

0 payment\_count,

0 payment\_num,

0 payment\_amount,

0 refund\_count,

0 refund\_num,

0 refund\_amount,

0 cart\_count,

0 cart\_num,

favor\_count,

0 appraise\_good\_count,

0 appraise\_mid\_count,

0 appraise\_bad\_count,

0 appraise\_default\_count

from tmp\_favor

union all

select

sku\_id,

0 order\_count,

0 order\_num,

0 order\_amount,

0 payment\_count,

0 payment\_num,

0 payment\_amount,

0 refund\_count,

0 refund\_num,

0 refund\_amount,

0 cart\_count,

0 cart\_num,

0 favor\_count,

appraise\_good\_count,

appraise\_mid\_count,

appraise\_bad\_count,

appraise\_default\_count

from tmp\_appraise

)tmp

group by sku\_id;

insert overwrite table ${APP}.dws\_coupon\_use\_daycount partition(dt='$do\_date')

select

cu.coupon\_id,

ci.coupon\_name,

ci.coupon\_type,

ci.condition\_amount,

ci.condition\_num,

ci.activity\_id,

ci.benefit\_amount,

ci.benefit\_discount,

ci.create\_time,

ci.range\_type,

ci.spu\_id,

ci.tm\_id,

ci.category3\_id,

ci.limit\_num,

cu.get\_count,

cu.using\_count,

cu.used\_count

from

(

select

coupon\_id,

sum(if(date\_format(get\_time,'yyyy-MM-dd')='$do\_date',1,0)) get\_count,

sum(if(date\_format(using\_time,'yyyy-MM-dd')='$do\_date',1,0)) using\_count,

sum(if(date\_format(used\_time,'yyyy-MM-dd')='$do\_date',1,0)) used\_count

from ${APP}.dwd\_fact\_coupon\_use

where dt='$do\_date'

group by coupon\_id

)cu

left join

(

select

\*

from ${APP}.dwd\_dim\_coupon\_info

where dt='$do\_date'

)ci on cu.coupon\_id=ci.id;

insert overwrite table ${APP}.dws\_activity\_info\_daycount partition(dt='$do\_date')

select

oi.activity\_id,

ai.activity\_name,

ai.activity\_type,

ai.start\_time,

ai.end\_time,

ai.create\_time,

oi.order\_count,

oi.payment\_count

from

(

select

activity\_id,

sum(if(date\_format(create\_time,'yyyy-MM-dd')='$do\_date',1,0)) order\_count,

sum(if(date\_format(payment\_time,'yyyy-MM-dd')='$do\_date',1,0)) payment\_count

from ${APP}.dwd\_fact\_order\_info

where (dt='$do\_date' or dt=date\_add('$do\_date',-1))

and activity\_id is not null

group by activity\_id

)oi

join

(

select

\*

from ${APP}.dwd\_dim\_activity\_info

where dt='$do\_date'

)ai

on oi.activity\_id=ai.id;

insert overwrite table ${APP}.dws\_sale\_detail\_daycount partition(dt='$do\_date')

select

op.user\_id,

op.sku\_id,

ui.gender,

months\_between('$do\_date', ui.birthday)/12 age,

ui.user\_level,

si.price,

si.sku\_name,

si.tm\_id,

si.category3\_id,

si.category2\_id,

si.category1\_id,

si.category3\_name,

si.category2\_name,

si.category1\_name,

si.spu\_id,

op.sku\_num,

op.order\_count,

op.order\_amount

from

(

select

user\_id,

sku\_id,

sum(sku\_num) sku\_num,

count(\*) order\_count,

sum(total\_amount) order\_amount

from ${APP}.dwd\_fact\_order\_detail

where dt='$do\_date'

group by user\_id, sku\_id

)op

join

(

select

\*

from ${APP}.dwd\_dim\_user\_info\_his

where end\_date='9999-99-99'

)ui on op.user\_id = ui.id

join

(

select

\*

from ${APP}.dwd\_dim\_sku\_info

where dt='$do\_date'

)si on op.sku\_id = si.id;

"

$hive -e "$sql"

2）增加脚本执行权限

[woaini@hadoop102 bin]$ chmod 777 dwd\_to\_dws.sh

3）执行脚本导入数据

[woaini@hadoop102 bin]$ dwd\_to\_dws.sh 2020-03-11

4）查看导入数据

hive (gmall)>

select \* from dws\_uv\_detail\_daycount where dt='2020-03-11';

select \* from dws\_user\_action\_daycount where dt='2020-03-11';

select \* from dws\_sku\_action\_daycount where dt='2020-03-11';

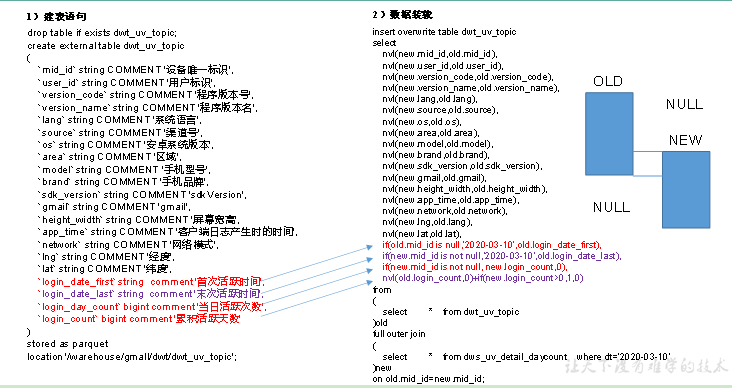
select \* from dws\_sale\_detail\_daycount where dt='2020-03-11';

select \* from dws\_coupon\_use\_daycount where dt='2020-03-11';

select \* from dws\_activity\_info\_daycount where dt='2020-03-11';

# 第6章 数仓搭建-DWT层

## 6.1 设备主题宽表



1）建表语句

hive (gmall)>

drop table if exists dwt\_uv\_topic;

create external table dwt\_uv\_topic

(

`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',

`login\_date\_first` string comment '首次活跃时间',

`login\_date\_last` string comment '末次活跃时间',

`login\_day\_count` bigint comment '当日活跃次数',

`login\_count` bigint comment '累积活跃天数'

)

stored as parquet

location '/warehouse/gmall/dwt/dwt\_uv\_topic'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwt\_uv\_topic

select

nvl(new.mid\_id,old.mid\_id),

nvl(new.user\_id,old.user\_id),

nvl(new.version\_code,old.version\_code),

nvl(new.version\_name,old.version\_name),

nvl(new.lang,old.lang),

nvl(new.source,old.source),

nvl(new.os,old.os),

nvl(new.area,old.area),

nvl(new.model,old.model),

nvl(new.brand,old.brand),

nvl(new.sdk\_version,old.sdk\_version),

nvl(new.gmail,old.gmail),

nvl(new.height\_width,old.height\_width),

nvl(new.app\_time,old.app\_time),

nvl(new.network,old.network),

nvl(new.lng,old.lng),

nvl(new.lat,old.lat),

if(old.mid\_id is null,'2020-03-10',old.login\_date\_first),

if(new.mid\_id is not null,'2020-03-10',old.login\_date\_last),

if(new.mid\_id is not null, new.login\_count,0),

nvl(old.login\_count,0)+if(new.login\_count>0,1,0)

from

(

select

\*

from dwt\_uv\_topic

)old

full outer join

(

select

\*

from dws\_uv\_detail\_daycount

where dt='2020-03-10'

)new

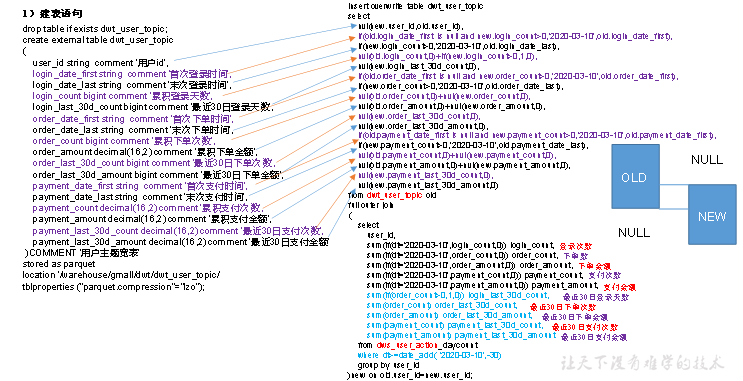
on old.mid\_id=new.mid\_id;

3）查询加载结果

hive (gmall)> select \* from dwt\_uv\_topic limit 5;

## 6.2 会员主题宽表

宽表字段怎么来？维度关联的事实表度量值+开头、结尾+累积+累积一个时间段。



1）建表语句

hive (gmall)>

drop table if exists dwt\_user\_topic;

create external table dwt\_user\_topic

(

user\_id string comment '用户id',

login\_date\_first string comment '首次登录时间',

login\_date\_last string comment '末次登录时间',

login\_count bigint comment '累积登录天数',

login\_last\_30d\_count bigint comment '最近30日登录天数',

order\_date\_first string comment '首次下单时间',

order\_date\_last string comment '末次下单时间',

order\_count bigint comment '累积下单次数',

order\_amount decimal(16,2) comment '累积下单金额',

order\_last\_30d\_count bigint comment '最近30日下单次数',

order\_last\_30d\_amount bigint comment '最近30日下单金额',

payment\_date\_first string comment '首次支付时间',

payment\_date\_last string comment '末次支付时间',

payment\_count decimal(16,2) comment '累积支付次数',

payment\_amount decimal(16,2) comment '累积支付金额',

payment\_last\_30d\_count decimal(16,2) comment '最近30日支付次数',

payment\_last\_30d\_amount decimal(16,2) comment '最近30日支付金额'

)COMMENT '用户主题宽表'

stored as parquet

location '/warehouse/gmall/dwt/dwt\_user\_topic/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwt\_user\_topic

select

nvl(new.user\_id,old.user\_id),

if(old.login\_date\_first is null and new.login\_count>0,'2020-03-10',old.login\_date\_first),

if(new.login\_count>0,'2020-03-10',old.login\_date\_last),

nvl(old.login\_count,0)+if(new.login\_count>0,1,0),

nvl(new.login\_last\_30d\_count,0),

if(old.order\_date\_first is null and new.order\_count>0,'2020-03-10',old.order\_date\_first),

if(new.order\_count>0,'2020-03-10',old.order\_date\_last),

nvl(old.order\_count,0)+nvl(new.order\_count,0),

nvl(old.order\_amount,0)+nvl(new.order\_amount,0),

nvl(new.order\_last\_30d\_count,0),

nvl(new.order\_last\_30d\_amount,0),

if(old.payment\_date\_first is null and new.payment\_count>0,'2020-03-10',old.payment\_date\_first),

if(new.payment\_count>0,'2020-03-10',old.payment\_date\_last),

nvl(old.payment\_count,0)+nvl(new.payment\_count,0),

nvl(old.payment\_amount,0)+nvl(new.payment\_amount,0),

nvl(new.payment\_last\_30d\_count,0),

nvl(new.payment\_last\_30d\_amount,0)

from

dwt\_user\_topic old

full outer join

(

select

user\_id,

sum(if(dt='2020-03-10',login\_count,0)) login\_count,

sum(if(dt='2020-03-10',order\_count,0)) order\_count,

sum(if(dt='2020-03-10',order\_amount,0)) order\_amount,

sum(if(dt='2020-03-10',payment\_count,0)) payment\_count,

sum(if(dt='2020-03-10',payment\_amount,0)) payment\_amount,

sum(if(login\_count>0,1,0)) login\_last\_30d\_count,

sum(order\_count) order\_last\_30d\_count,

sum(order\_amount) order\_last\_30d\_amount,

sum(payment\_count) payment\_last\_30d\_count,

sum(payment\_amount) payment\_last\_30d\_amount

from dws\_user\_action\_daycount

where dt>=date\_add( '2020-03-10',-30)

group by user\_id

)new

on old.user\_id=new.user\_id;

3）查询加载结果

hive (gmall)> select \* from dwt\_user\_topic limit 5;

## 6.3 商品主题宽表

1）建表语句

hive (gmall)>

drop table if exists dwt\_sku\_topic;

create external table dwt\_sku\_topic

(

sku\_id string comment 'sku\_id',

spu\_id string comment 'spu\_id',

order\_last\_30d\_count bigint comment '最近30日被下单次数',

order\_last\_30d\_num bigint comment '最近30日被下单件数',

order\_last\_30d\_amount decimal(16,2) comment '最近30日被下单金额',

order\_count bigint comment '累积被下单次数',

order\_num bigint comment '累积被下单件数',

order\_amount decimal(16,2) comment '累积被下单金额',

payment\_last\_30d\_count bigint comment '最近30日被支付次数',

payment\_last\_30d\_num bigint comment '最近30日被支付件数',

payment\_last\_30d\_amount decimal(16,2) comment '最近30日被支付金额',

payment\_count bigint comment '累积被支付次数',

payment\_num bigint comment '累积被支付件数',

payment\_amount decimal(16,2) comment '累积被支付金额',

refund\_last\_30d\_count bigint comment '最近三十日退款次数',

refund\_last\_30d\_num bigint comment '最近三十日退款件数',

refund\_last\_30d\_amount decimal(10,2) comment '最近三十日退款金额',

refund\_count bigint comment '累积退款次数',

refund\_num bigint comment '累积退款件数',

refund\_amount decimal(10,2) comment '累积退款金额',

cart\_last\_30d\_count bigint comment '最近30日被加入购物车次数',

cart\_last\_30d\_num bigint comment '最近30日被加入购物车件数',

cart\_count bigint comment '累积被加入购物车次数',

cart\_num bigint comment '累积被加入购物车件数',

favor\_last\_30d\_count bigint comment '最近30日被收藏次数',

favor\_count bigint comment '累积被收藏次数',

appraise\_last\_30d\_good\_count bigint comment '最近30日好评数',

appraise\_last\_30d\_mid\_count bigint comment '最近30日中评数',

appraise\_last\_30d\_bad\_count bigint comment '最近30日差评数',

appraise\_last\_30d\_default\_count bigint comment '最近30日默认评价数',

appraise\_good\_count bigint comment '累积好评数',

appraise\_mid\_count bigint comment '累积中评数',

appraise\_bad\_count bigint comment '累积差评数',

appraise\_default\_count bigint comment '累积默认评价数'

)COMMENT '商品主题宽表'

stored as parquet

location '/warehouse/gmall/dwt/dwt\_sku\_topic/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwt\_sku\_topic

select

nvl(new.sku\_id,old.sku\_id),

dwd\_dim\_sku\_info.spu\_id,

nvl(new.order\_count30,0),

nvl(new.order\_num30,0),

nvl(new.order\_amount30,0),

nvl(old.order\_count,0) + nvl(new.order\_count,0),

nvl(old.order\_num,0) + nvl(new.order\_num,0),

nvl(old.order\_amount,0) + nvl(new.order\_amount,0),

nvl(new.payment\_count30,0),

nvl(new.payment\_num30,0),

nvl(new.payment\_amount30,0),

nvl(old.payment\_count,0) + nvl(new.payment\_count,0),

nvl(old.payment\_num,0) + nvl(new.payment\_count,0),

nvl(old.payment\_amount,0) + nvl(new.payment\_count,0),

nvl(new.refund\_count30,0),

nvl(new.refund\_num30,0),

nvl(new.refund\_amount30,0),

nvl(old.refund\_count,0) + nvl(new.refund\_count,0),

nvl(old.refund\_num,0) + nvl(new.refund\_num,0),

nvl(old.refund\_amount,0) + nvl(new.refund\_amount,0),

nvl(new.cart\_count30,0),

nvl(new.cart\_num30,0),

nvl(old.cart\_count,0) + nvl(new.cart\_count,0),

nvl(old.cart\_num,0) + nvl(new.cart\_num,0),

nvl(new.favor\_count30,0),

nvl(old.favor\_count,0) + nvl(new.favor\_count,0),

nvl(new.appraise\_good\_count30,0),

nvl(new.appraise\_mid\_count30,0),

nvl(new.appraise\_bad\_count30,0),

nvl(new.appraise\_default\_count30,0) ,

nvl(old.appraise\_good\_count,0) + nvl(new.appraise\_good\_count,0),

nvl(old.appraise\_mid\_count,0) + nvl(new.appraise\_mid\_count,0),

nvl(old.appraise\_bad\_count,0) + nvl(new.appraise\_bad\_count,0),

nvl(old.appraise\_default\_count,0) + nvl(new.appraise\_default\_count,0)

from

(

select

sku\_id,

spu\_id,

order\_last\_30d\_count,

order\_last\_30d\_num,

order\_last\_30d\_amount,

order\_count,

order\_num,

order\_amount ,

payment\_last\_30d\_count,

payment\_last\_30d\_num,

payment\_last\_30d\_amount,

payment\_count,

payment\_num,

payment\_amount,

refund\_last\_30d\_count,

refund\_last\_30d\_num,

refund\_last\_30d\_amount,

refund\_count,

refund\_num,

refund\_amount,

cart\_last\_30d\_count,

cart\_last\_30d\_num,

cart\_count,

cart\_num,

favor\_last\_30d\_count,

favor\_count,

appraise\_last\_30d\_good\_count,

appraise\_last\_30d\_mid\_count,

appraise\_last\_30d\_bad\_count,

appraise\_last\_30d\_default\_count,

appraise\_good\_count,

appraise\_mid\_count,

appraise\_bad\_count,

appraise\_default\_count

from dwt\_sku\_topic

)old

full outer join

(

select

sku\_id,

sum(if(dt='2020-03-10', order\_count,0 )) order\_count,

sum(if(dt='2020-03-10',order\_num ,0 )) order\_num,

sum(if(dt='2020-03-10',order\_amount,0 )) order\_amount ,

sum(if(dt='2020-03-10',payment\_count,0 )) payment\_count,

sum(if(dt='2020-03-10',payment\_num,0 )) payment\_num,

sum(if(dt='2020-03-10',payment\_amount,0 )) payment\_amount,

sum(if(dt='2020-03-10',refund\_count,0 )) refund\_count,

sum(if(dt='2020-03-10',refund\_num,0 )) refund\_num,

sum(if(dt='2020-03-10',refund\_amount,0 )) refund\_amount,

sum(if(dt='2020-03-10',cart\_count,0 )) cart\_count,

sum(if(dt='2020-03-10',cart\_num,0 )) cart\_num,

sum(if(dt='2020-03-10',favor\_count,0 )) favor\_count,

sum(if(dt='2020-03-10',appraise\_good\_count,0 )) appraise\_good\_count,

sum(if(dt='2020-03-10',appraise\_mid\_count,0 ) ) appraise\_mid\_count ,

sum(if(dt='2020-03-10',appraise\_bad\_count,0 )) appraise\_bad\_count,

sum(if(dt='2020-03-10',appraise\_default\_count,0 )) appraise\_default\_count,

sum(order\_count) order\_count30 ,

sum(order\_num) order\_num30,

sum(order\_amount) order\_amount30,

sum(payment\_count) payment\_count30,

sum(payment\_num) payment\_num30,

sum(payment\_amount) payment\_amount30,

sum(refund\_count) refund\_count30,

sum(refund\_num) refund\_num30,

sum(refund\_amount) refund\_amount30,

sum(cart\_count) cart\_count30,

sum(cart\_num) cart\_num30,

sum(favor\_count) favor\_count30,

sum(appraise\_good\_count) appraise\_good\_count30,

sum(appraise\_mid\_count) appraise\_mid\_count30,

sum(appraise\_bad\_count) appraise\_bad\_count30,

sum(appraise\_default\_count) appraise\_default\_count30

from dws\_sku\_action\_daycount

where dt >= date\_add ('2020-03-10', -30)

group by sku\_id

)new

on new.sku\_id = old.sku\_id

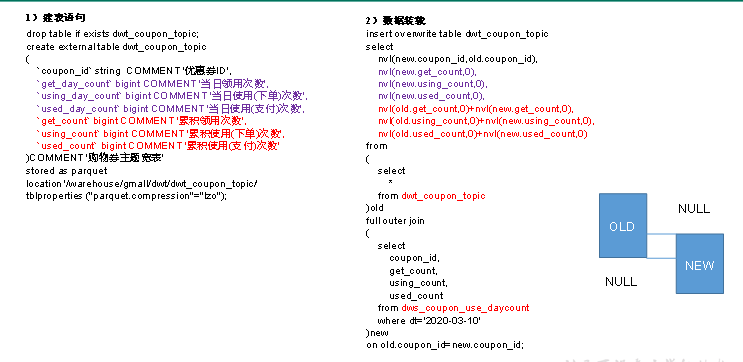
left join dwd\_dim\_sku\_info

on new.sku\_id = dwd\_dim\_sku\_info.id;

3）查询加载结果

hive (gmall)> select \* from dwt\_sku\_topic limit 5;

## 6.4 优惠券主题宽表（预留）



1）建表语句

hive (gmall)>

drop table if exists dwt\_coupon\_topic;

create external table dwt\_coupon\_topic

(

`coupon\_id` string COMMENT '优惠券ID',

`get\_day\_count` bigint COMMENT '当日领用次数',

`using\_day\_count` bigint COMMENT '当日使用(下单)次数',

`used\_day\_count` bigint COMMENT '当日使用(支付)次数',

`get\_count` bigint COMMENT '累积领用次数',

`using\_count` bigint COMMENT '累积使用(下单)次数',

`used\_count` bigint COMMENT '累积使用(支付)次数'

)COMMENT '购物券主题宽表'

stored as parquet

location '/warehouse/gmall/dwt/dwt\_coupon\_topic/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwt\_coupon\_topic

select

nvl(new.coupon\_id,old.coupon\_id),

nvl(new.get\_count,0),

nvl(new.using\_count,0),

nvl(new.used\_count,0),

nvl(old.get\_count,0)+nvl(new.get\_count,0),

nvl(old.using\_count,0)+nvl(new.using\_count,0),

nvl(old.used\_count,0)+nvl(new.used\_count,0)

from

(

select

\*

from dwt\_coupon\_topic

)old

full outer join

(

select

coupon\_id,

get\_count,

using\_count,

used\_count

from dws\_coupon\_use\_daycount

where dt='2020-03-10'

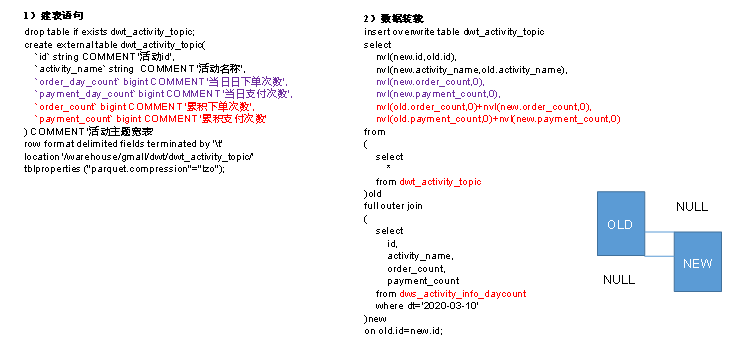
)new

on old.coupon\_id=new.coupon\_id;

3）查询加载结果

hive (gmall)> select \* from dwt\_coupon\_topic limit 5;

## 6.5 活动主题宽表（预留）



1）建表语句

hive (gmall)>

drop table if exists dwt\_activity\_topic;

create external table dwt\_activity\_topic(

`id` string COMMENT '活动id',

`activity\_name` string COMMENT '活动名称',

`order\_day\_count` bigint COMMENT '当日日下单次数',

`payment\_day\_count` bigint COMMENT '当日支付次数',

`order\_count` bigint COMMENT '累积下单次数',

`payment\_count` bigint COMMENT '累积支付次数'

) COMMENT '活动主题宽表'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/dwt/dwt\_activity\_topic/'

tblproperties ("parquet.compression"="lzo");

2）数据装载

hive (gmall)>

insert overwrite table dwt\_activity\_topic

select

nvl(new.id,old.id),

nvl(new.activity\_name,old.activity\_name),

nvl(new.order\_count,0),

nvl(new.payment\_count,0),

nvl(old.order\_count,0)+nvl(new.order\_count,0),

nvl(old.payment\_count,0)+nvl(new.payment\_count,0)

from

(

select

\*

from dwt\_activity\_topic

)old

full outer join

(

select

id,

activity\_name,

order\_count,

payment\_count

from dws\_activity\_info\_daycount

where dt='2020-03-10'

)new

on old.id=new.id;

3）查询加载结果

hive (gmall)> select \* from dwt\_activity\_topic limit 5;

## 6.6 DWT层数据导入脚本

1）在/home/woaini/bin目录下创建脚本dws\_to\_dwt.sh

[woaini@hadoop102 bin]$ vim dws\_to\_dwt.sh

在脚本中填写如下内容

#!/bin/bash

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="

insert overwrite table ${APP}.dwt\_uv\_topic

select

nvl(new.mid\_id,old.mid\_id),

nvl(new.user\_id,old.user\_id),

nvl(new.version\_code,old.version\_code),

nvl(new.version\_name,old.version\_name),

nvl(new.lang,old.lang),

nvl(new.source,old.source),

nvl(new.os,old.os),

nvl(new.area,old.area),

nvl(new.model,old.model),

nvl(new.brand,old.brand),

nvl(new.sdk\_version,old.sdk\_version),

nvl(new.gmail,old.gmail),

nvl(new.height\_width,old.height\_width),

nvl(new.app\_time,old.app\_time),

nvl(new.network,old.network),

nvl(new.lng,old.lng),

nvl(new.lat,old.lat),

nvl(old.login\_date\_first,'$do\_date'),

if(new.login\_count>0,'$do\_date',old.login\_date\_last),

nvl(new.login\_count,0),

nvl(new.login\_count,0)+nvl(old.login\_count,0)

from

(

select

\*

from ${APP}.dwt\_uv\_topic

)old

full outer join

(

select

\*

from ${APP}.dws\_uv\_detail\_daycount

where dt='$do\_date'

)new

on old.mid\_id=new.mid\_id;

insert overwrite table ${APP}.dwt\_user\_topic

select

nvl(new.user\_id,old.user\_id),

if(old.login\_date\_first is null and new.login\_count>0,'$do\_date',old.login\_date\_first),

if(new.login\_count>0,'$do\_date',old.login\_date\_last),

nvl(old.login\_count,0)+if(new.login\_count>0,1,0),

nvl(new.login\_last\_30d\_count,0),

if(old.order\_date\_first is null and new.order\_count>0,'$do\_date',old.order\_date\_first),

if(new.order\_count>0,'$do\_date',old.order\_date\_last),

nvl(old.order\_count,0)+nvl(new.order\_count,0),

nvl(old.order\_amount,0)+nvl(new.order\_amount,0),

nvl(new.order\_last\_30d\_count,0),

nvl(new.order\_last\_30d\_amount,0),

if(old.payment\_date\_first is null and new.payment\_count>0,'$do\_date',old.payment\_date\_first),

if(new.payment\_count>0,'$do\_date',old.payment\_date\_last),

nvl(old.payment\_count,0)+nvl(new.payment\_count,0),

nvl(old.payment\_amount,0)+nvl(new.payment\_amount,0),

nvl(new.payment\_last\_30d\_count,0),

nvl(new.payment\_last\_30d\_amount,0)

from

(

select

\*

from ${APP}.dwt\_user\_topic

)old

full outer join

(

select

user\_id,

sum(if(dt='$do\_date',login\_count,0)) login\_count,

sum(if(dt='$do\_date',order\_count,0)) order\_count,

sum(if(dt='$do\_date',order\_amount,0)) order\_amount,

sum(if(dt='$do\_date',payment\_count,0)) payment\_count,

sum(if(dt='$do\_date',payment\_amount,0)) payment\_amount,

sum(if(order\_count>0,1,0)) login\_last\_30d\_count,

sum(order\_count) order\_last\_30d\_count,

sum(order\_amount) order\_last\_30d\_amount,

sum(payment\_count) payment\_last\_30d\_count,

sum(payment\_amount) payment\_last\_30d\_amount

from ${APP}.dws\_user\_action\_daycount

where dt>=date\_add( '$do\_date',-30)

group by user\_id

)new

on old.user\_id=new.user\_id;

with

sku\_act as

(

select

sku\_id,

sum(if(dt='$do\_date', order\_count,0 )) order\_count,

sum(if(dt='$do\_date',order\_num ,0 )) order\_num,

sum(if(dt='$do\_date',order\_amount,0 )) order\_amount ,

sum(if(dt='$do\_date',payment\_count,0 )) payment\_count,

sum(if(dt='$do\_date',payment\_num,0 )) payment\_num,

sum(if(dt='$do\_date',payment\_amount,0 )) payment\_amount,

sum(if(dt='$do\_date',refund\_count,0 )) refund\_count,

sum(if(dt='$do\_date',refund\_num,0 )) refund\_num,

sum(if(dt='$do\_date',refund\_amount,0 )) refund\_amount,

sum(if(dt='$do\_date',cart\_count,0 )) cart\_count,

sum(if(dt='$do\_date',cart\_num,0 )) cart\_num,

sum(if(dt='$do\_date',favor\_count,0 )) favor\_count,

sum(if(dt='$do\_date',appraise\_good\_count,0 )) appraise\_good\_count,

sum(if(dt='$do\_date',appraise\_mid\_count,0 ) ) appraise\_mid\_count ,

sum(if(dt='$do\_date',appraise\_bad\_count,0 )) appraise\_bad\_count,

sum(if(dt='$do\_date',appraise\_default\_count,0 )) appraise\_default\_count,

sum( order\_count ) order\_count30 ,

sum( order\_num ) order\_num30,

sum(order\_amount ) order\_amount30,

sum(payment\_count ) payment\_count30,

sum(payment\_num ) payment\_num30,

sum(payment\_amount ) payment\_amount30,

sum(refund\_count ) refund\_count30,

sum(refund\_num ) refund\_num30,

sum(refund\_amount ) refund\_amount30,

sum(cart\_count ) cart\_count30,

sum(cart\_num ) cart\_num30,

sum(favor\_count ) favor\_count30,

sum(appraise\_good\_count ) appraise\_good\_count30,

sum(appraise\_mid\_count ) appraise\_mid\_count30,

sum(appraise\_bad\_count ) appraise\_bad\_count30,

sum(appraise\_default\_count ) appraise\_default\_count30

from ${APP}.dws\_sku\_action\_daycount

where dt>=date\_add ( '$do\_date',-30)

group by sku\_id

),

sku\_topic

as

(

select

sku\_id,

spu\_id,

order\_last\_30d\_count,

order\_last\_30d\_num,

order\_last\_30d\_amount,

order\_count,

order\_num,

order\_amount ,

payment\_last\_30d\_count,

payment\_last\_30d\_num,

payment\_last\_30d\_amount,

payment\_count,

payment\_num,

payment\_amount,

refund\_last\_30d\_count,

refund\_last\_30d\_num,

refund\_last\_30d\_amount ,

refund\_count ,

refund\_num ,

refund\_amount ,

cart\_last\_30d\_count ,

cart\_last\_30d\_num ,

cart\_count ,

cart\_num ,

favor\_last\_30d\_count ,

favor\_count ,

appraise\_last\_30d\_good\_count ,

appraise\_last\_30d\_mid\_count ,

appraise\_last\_30d\_bad\_count ,

appraise\_last\_30d\_default\_count ,

appraise\_good\_count ,

appraise\_mid\_count ,

appraise\_bad\_count ,

appraise\_default\_count

from ${APP}.dwt\_sku\_topic

)

insert overwrite table ${APP}.dwt\_sku\_topic

select

nvl(sku\_act.sku\_id,sku\_topic.sku\_id) ,

dwd\_dim\_sku\_info.spu\_id,

nvl (sku\_act.order\_count30,0) ,

nvl (sku\_act.order\_num30,0) ,

nvl (sku\_act.order\_amount30,0) ,

nvl(sku\_topic.order\_count,0)+ nvl (sku\_act.order\_count,0) ,

nvl(sku\_topic.order\_num,0)+ nvl (sku\_act.order\_num,0) ,

nvl(sku\_topic.order\_amount,0)+ nvl (sku\_act.order\_amount,0),

nvl (sku\_act.payment\_count30,0),

nvl (sku\_act.payment\_num30,0),

nvl (sku\_act.payment\_amount30,0),

nvl(sku\_topic.payment\_count,0)+ nvl (sku\_act.payment\_count,0) ,

nvl(sku\_topic.payment\_num,0)+ nvl (sku\_act.payment\_count,0) ,

nvl(sku\_topic.payment\_amount,0)+ nvl (sku\_act.payment\_count,0) ,

nvl (refund\_count30,0),

nvl (sku\_act.refund\_num30,0),

nvl (sku\_act.refund\_amount30,0),

nvl(sku\_topic.refund\_count,0)+ nvl (sku\_act.refund\_count,0),

nvl(sku\_topic.refund\_num,0)+ nvl (sku\_act.refund\_num,0),

nvl(sku\_topic.refund\_amount,0)+ nvl (sku\_act.refund\_amount,0),

nvl(sku\_act.cart\_count30,0) ,

nvl(sku\_act.cart\_num30,0) ,

nvl(sku\_topic.cart\_count ,0)+ nvl (sku\_act.cart\_count,0),

nvl( sku\_topic.cart\_num ,0)+ nvl (sku\_act.cart\_num,0),

nvl(sku\_act.favor\_count30 ,0) ,

nvl (sku\_topic.favor\_count ,0)+ nvl (sku\_act.favor\_count,0),

nvl (sku\_act.appraise\_good\_count30 ,0) ,

nvl (sku\_act.appraise\_mid\_count30 ,0) ,

nvl (sku\_act.appraise\_bad\_count30 ,0) ,

nvl (sku\_act.appraise\_default\_count30 ,0) ,

nvl (sku\_topic.appraise\_good\_count ,0)+ nvl (sku\_act.appraise\_good\_count,0) ,

nvl (sku\_topic.appraise\_mid\_count ,0)+ nvl (sku\_act.appraise\_mid\_count,0) ,

nvl (sku\_topic.appraise\_bad\_count ,0)+ nvl (sku\_act.appraise\_bad\_count,0) ,

nvl (sku\_topic.appraise\_default\_count ,0)+ nvl (sku\_act.appraise\_default\_count,0)

from sku\_act

full outer join sku\_topic

on sku\_act.sku\_id =sku\_topic.sku\_id

left join ${APP}.dwd\_dim\_sku\_info

on sku\_act.sku\_id = dwd\_dim\_sku\_info.id;

insert overwrite table ${APP}.dwt\_coupon\_topic

select

nvl(new.coupon\_id,old.coupon\_id),

nvl(new.get\_count,0),

nvl(new.using\_count,0),

nvl(new.used\_count,0),

nvl(old.get\_count,0)+nvl(new.get\_count,0),

nvl(old.using\_count,0)+nvl(new.using\_count,0),

nvl(old.used\_count,0)+nvl(new.used\_count,0)

from

(

select

\*

from ${APP}.dwt\_coupon\_topic

)old

full outer join

(

select

coupon\_id,

get\_count,

using\_count,

used\_count

from ${APP}.dws\_coupon\_use\_daycount

where dt='$do\_date'

)new

on old.coupon\_id=new.coupon\_id;

insert overwrite table ${APP}.dwt\_activity\_topic

select

nvl(new.id,old.id),

nvl(new.activity\_name,old.activity\_name),

nvl(new.order\_count,0),

nvl(new.payment\_count,0),

nvl(old.order\_count,0)+nvl(new.order\_count,0),

nvl(old.payment\_count,0)+nvl(new.payment\_count,0)

from

(

select

\*

from ${APP}.dwt\_activity\_topic

)old

full outer join

(

select

id,

activity\_name,

order\_count,

payment\_count

from ${APP}.dws\_activity\_info\_daycount

where dt='$do\_date'

)new

on old.id=new.id;

"

$hive -e "$sql"

2）增加脚本执行权限

[woaini@hadoop102 bin]$ chmod 777 dws\_to\_dwt.sh

3）执行脚本导入数据

[woaini@hadoop102 bin]$ dws\_to\_dwt.sh 2020-03-11

4）查看导入数据

hive (gmall)>

select \* from dwt\_uv\_topic limit 5;

select \* from dwt\_user\_topic limit 5;

select \* from dwt\_sku\_topic limit 5;

select \* from dwt\_coupon\_topic limit 5;

select \* from dwt\_activity\_topic limit 5;

# 第7章 数仓搭建-ADS层

## 7.1 设备主题

### 7.1.1 活跃设备数（日、周、月）

需求定义：

日活：当日活跃的**设备数**

周活：当周活跃的**设备数**

月活：当月活跃的**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 '活跃设备数'

row format delimited fields terminated by '\t'

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

2）导入数据

hive (gmall)>

insert into table ads\_uv\_count

select

'2020-03-10' dt,

daycount.ct,

wkcount.ct,

mncount.ct,

if(date\_add(next\_day('2020-03-10','MO'),-1)='2020-03-10','Y','N') ,

if(last\_day('2020-03-10')='2020-03-10','Y','N')

from

(

select

'2020-03-10' dt,

count(\*) ct

from dwt\_uv\_topic

where login\_date\_last='2020-03-10'

)daycount join

(

select

'2020-03-10' dt,

count (\*) ct

from dwt\_uv\_topic

where login\_date\_last>=date\_add(next\_day('2020-03-10','MO'),-7)

and login\_date\_last<= date\_add(next\_day('2020-03-10','MO'),-1)

) wkcount on daycount.dt=wkcount.dt

join

(

select

'2020-03-10' dt,

count (\*) ct

from dwt\_uv\_topic

where date\_format(login\_date\_last,'yyyy-MM')=date\_format('2020-03-10','yyyy-MM')

)mncount on daycount.dt=mncount.dt;

3）查询导入结果

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

### 7.1.2 每日新增设备

1）建表语句

hive (gmall)>

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

create external 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

login\_date\_first,

count(\*)

from dwt\_uv\_topic

where login\_date\_first='2020-03-10'

group by login\_date\_first;

3）查询导入数据

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

### 7.1.3 沉默用户数

需求定义：

沉默用户：只在安装当天启动过，且启动时间是在7天前

1）建表语句

hive (gmall)>

drop table if exists ads\_silent\_count;

create external table ads\_silent\_count(

`dt` string COMMENT '统计日期',

`silent\_count` bigint COMMENT '沉默设备数'

)

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_silent\_count';

2）导入2020-03-20数据

hive (gmall)>

insert into table ads\_silent\_count

select

'2020-03-15',

count(\*)

from dwt\_uv\_topic

where login\_date\_first=login\_date\_last

and login\_date\_last<=date\_add('2020-03-15',-7);

3）查询导入数据

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

### 7.1.4 本周回流用户数

需求定义：

本周回流用户：上周未活跃，本周活跃的设备，且不是本周新增设备

1）建表语句

hive (gmall)>

drop table if exists ads\_back\_count;

create external table ads\_back\_count(

`dt` string COMMENT '统计日期',

`wk\_dt` string COMMENT '统计日期所在周',

`wastage\_count` bigint COMMENT '回流设备数'

)

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_back\_count';

2）导入数据：

hive (gmall)>

insert into table ads\_back\_count

select

'2020-03-15',

count(\*)

from

(

select

mid\_id

from dwt\_uv\_topic

where login\_date\_last>=date\_add(next\_day('2020-03-15','MO'),-7)

and login\_date\_last<= date\_add(next\_day('2020-03-15','MO'),-1)

and login\_date\_first<date\_add(next\_day('2020-03-15','MO'),-7)

)current\_wk

left join

(

select

mid\_id

from dws\_uv\_detail\_daycount

where dt>=date\_add(next\_day('2020-03-15','MO'),-7\*2)

and dt<= date\_add(next\_day('2020-03-15','MO'),-7-1)

group by mid\_id

)last\_wk

on current\_wk.mid\_id=last\_wk.mid\_id

where last\_wk.mid\_id is null;

3）查询结果

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

### 7.1.5 流失用户数

需求定义：

流失用户：连续7天未活跃的设备

1）建表语句

hive (gmall)>

drop table if exists ads\_wastage\_count;

create external table ads\_wastage\_count(

`dt` string COMMENT '统计日期',

`wastage\_count` bigint COMMENT '流失设备数'

)

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_wastage\_count';

2）导入2020-03-20数据

hive (gmall)>

insert into table ads\_wastage\_count

select

'2020-03-20',

count(\*)

from

(

select

mid\_id

from dwt\_uv\_topic

where login\_date\_last<=date\_add('2020-03-20',-7)

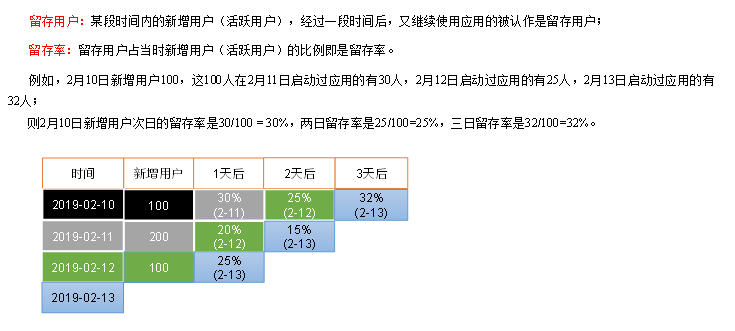
group by mid\_id

)t1;

3）查询结果

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

### 7.1.6 留存率



1）建表语句

hive (gmall)>

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

create external 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`     bigint comment '设备新增数量',

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

) COMMENT '每日用户留存情况'

row format delimited fields terminated by '\t'

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

'2020-03-10',--统计日期

date\_add('2020-03-10',-1),--新增日期

1,--留存天数

sum(if(login\_date\_first=date\_add('2020-03-10',-1) and login\_date\_last='2020-03-10',1,0)),--2020-03-09的1日留存数

sum(if(login\_date\_first=date\_add('2020-03-10',-1),1,0)),--2020-03-09新增

sum(if(login\_date\_first=date\_add('2020-03-10',-1) and login\_date\_last='2020-03-10',1,0))/sum(if(login\_date\_first=date\_add('2020-03-10',-1),1,0))\*100

from dwt\_uv\_topic

union all

select

'2020-03-10',--统计日期

date\_add('2020-03-10',-2),--新增日期

2,--留存天数

sum(if(login\_date\_first=date\_add('2020-03-10',-2) and login\_date\_last='2020-03-10',1,0)),--2020-03-08的2日留存数

sum(if(login\_date\_first=date\_add('2020-03-10',-2),1,0)),--2020-03-08新增

sum(if(login\_date\_first=date\_add('2020-03-10',-2) and login\_date\_last='2020-03-10',1,0))/sum(if(login\_date\_first=date\_add('2020-03-10',-2),1,0))\*100

from dwt\_uv\_topic

union all

select

'2020-03-10',--统计日期

date\_add('2020-03-10',-3),--新增日期

3,--留存天数

sum(if(login\_date\_first=date\_add('2020-03-10',-3) and login\_date\_last='2020-03-10',1,0)),--2020-03-07的3日留存数

sum(if(login\_date\_first=date\_add('2020-03-10',-3),1,0)),--2020-03-07新增

sum(if(login\_date\_first=date\_add('2020-03-10',-3) and login\_date\_last='2020-03-10',1,0))/sum(if(login\_date\_first=date\_add('2020-03-10',-3),1,0))\*100

from dwt\_uv\_topic;

3）查询导入数据

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

### 7.1.7 最近连续三周活跃用户数

1）建表语句

hive (gmall)>

drop table if exists ads\_continuity\_wk\_count;

create external table ads\_continuity\_wk\_count(

`dt` string COMMENT '统计日期,一般用结束周周日日期,如果每天计算一次,可用当天日期',

`wk\_dt` string COMMENT '持续时间',

`continuity\_count` bigint COMMENT '活跃次数'

)

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_continuity\_wk\_count';

2）导入2020-03-20所在周的数据

hive (gmall)>

insert into table ads\_continuity\_wk\_count

select

'2020-03-15',

concat(date\_add(next\_day('2020-03-15','MO'),-7\*3),'\_',date\_add(next\_day('2020-03-15','MO'),-1)),

count(\*)

from

(

select

mid\_id

from

(

select

mid\_id

from dws\_uv\_detail\_daycount

where dt>=date\_add(next\_day('2020-03-10','monday'),-7)

and dt<=date\_add(next\_day('2020-03-10','monday'),-1)

group by mid\_id

union all

select

mid\_id

from dws\_uv\_detail\_daycount

where dt>=date\_add(next\_day('2020-03-10','monday'),-7\*2)

and dt<=date\_add(next\_day('2020-03-10','monday'),-7-1)

group by mid\_id

union all

select

mid\_id

from dws\_uv\_detail\_daycount

where dt>=date\_add(next\_day('2020-03-10','monday'),-7\*3)

and dt<=date\_add(next\_day('2020-03-10','monday'),-7\*2-1)

group by mid\_id

)t1

group by mid\_id

having count(\*)=3

)t2

3）查询

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

### 7.1.8 最近七天内连续三天活跃用户数

1）建表语句

hive (gmall)>

drop table if exists ads\_continuity\_uv\_count;

create external table ads\_continuity\_uv\_count(

`dt` string COMMENT '统计日期',

`wk\_dt` string COMMENT '最近7天日期',

`continuity\_count` bigint

) COMMENT '连续活跃设备数'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_continuity\_uv\_count';

2）写出导入数据的SQL语句

hive (gmall)>

insert into table ads\_continuity\_uv\_count

select

'2020-03-12',

concat(date\_add('2020-03-12',-6),'\_','2020-03-12'),

count(\*)

from

(

select mid\_id

from

(

select mid\_id

from

(

select

mid\_id,

date\_sub(dt,rank) date\_dif

from

(

select

mid\_id,

dt,

rank() over(partition by mid\_id order by dt) rank

from dws\_uv\_detail\_daycount

where dt>=date\_add('2020-03-12',-6) and dt<='2020-03-12'

)t1

)t2

group by mid\_id,date\_dif

having count(\*)>=3

)t3

group by mid\_id

)t4;

3）查询

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

## 7.2 会员主题

### 7.2.1 会员主题信息

1）建表

hive (gmall)>

drop table if exists ads\_user\_topic;

create external table ads\_user\_topic(

`dt` string COMMENT '统计日期',

`day\_users` string COMMENT '活跃会员数',

`day\_new\_users` string COMMENT '新增会员数',

`day\_new\_payment\_users` string COMMENT '新增消费会员数',

`payment\_users` string COMMENT '总付费会员数',

`users` string COMMENT '总会员数',

`day\_users2users` decimal(10,2) COMMENT '会员活跃率',

`payment\_users2users` decimal(10,2) COMMENT '会员付费率',

`day\_new\_users2users` decimal(10,2) COMMENT '会员新鲜度'

) COMMENT '会员主题信息表'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_user\_topic';

2）导入数据

hive (gmall)>

insert into table ads\_user\_topic

select

'2020-03-10',

sum(if(login\_date\_last='2020-03-10',1,0)),

sum(if(login\_date\_first='2020-03-10',1,0)),

sum(if(payment\_date\_first='2020-03-10',1,0)),

sum(if(payment\_count>0,1,0)),

count(\*),

sum(if(login\_date\_last='2020-03-10',1,0))/count(\*),

sum(if(payment\_count>0,1,0))/count(\*),

sum(if(login\_date\_first='2020-03-10',1,0))/sum(if(login\_date\_last='2020-03-10',1,0))

from dwt\_user\_topic

3）查询数据

hive (gmall)> select \* from ads\_user\_topic;

4）在/home/woaini/bin目录下编写脚本ads\_user\_topic.sh

[woaini@hadoop102 bin]$ vim ads\_user\_topic.sh

添加如下内容：

#!/bin/bash

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="

with

tmp\_day\_users as

(

select

'$do\_date' dt,

count(\*) day\_users

from

${APP}.dwt\_user\_topic

where

login\_date\_last='$do\_date'

),

tmp\_day\_new\_users as

(

select

'$do\_date' dt,

count(\*) day\_new\_users

from

${APP}.dwt\_user\_topic

where

login\_date\_last='$do\_date' and login\_date\_first='$do\_date'

),

tmp\_day\_new\_payment\_users as

(

select

'$do\_date' dt,

count(\*) day\_new\_payment\_users

from

${APP}.dwt\_user\_topic

where

payment\_date\_first='$do\_date'

),

tmp\_payment\_users as

(

select

'$do\_date' dt,

count(\*) payment\_users

from

${APP}.dwt\_user\_topic

where

payment\_date\_first is not null

),

tmp\_users as

(

select

'$do\_date' dt,

count(\*) users

from

${APP}.dwt\_user\_topic

tmp\_users

)

insert into table ${APP}.ads\_user\_topic

select

'$do\_date' dt,

day\_users,

day\_new\_users,

day\_new\_payment\_users,

payment\_users,

users,

day\_users/users,

payment\_users/users,

day\_new\_users/users

from

tmp\_day\_users

join

tmp\_day\_new\_users

on

tmp\_day\_users.dt=tmp\_day\_new\_users.dt

join

tmp\_day\_new\_payment\_users

on

tmp\_day\_users.dt=tmp\_day\_new\_payment\_users.dt

join

tmp\_payment\_users

on

tmp\_day\_users.dt=tmp\_payment\_users.dt

join

tmp\_users

on

tmp\_day\_users.dt=tmp\_users.dt;

"

$hive -e "$sql"

5）增加脚本执行权限

[woaini@hadoop102 bin]$ chmod 777 ads\_user\_topic.sh

6）执行脚本导入数据

[woaini@hadoop102 bin]$ ads\_user\_topic.sh 2020-03-11

7）查看导入数据

hive (gmall)> select \* from ads\_user\_topic;

### 7.2.2 漏斗分析

统计“浏览->购物车->下单->支付”的转化率

思路：统计各个行为的人数，然后计算比值。

1）建表语句

hive (gmall)>

drop table if exists ads\_user\_action\_convert\_day;

create external table ads\_user\_action\_convert\_day(

`dt` string COMMENT '统计日期',

`total\_visitor\_m\_count` bigint COMMENT '总访问人数',

`cart\_u\_count` bigint COMMENT '加入购物车的人数',

`visitor2cart\_convert\_ratio` decimal(10,2) COMMENT '访问到加入购物车转化率',

`order\_u\_count` bigint COMMENT '下单人数',

`cart2order\_convert\_ratio` decimal(10,2) COMMENT '加入购物车到下单转化率',

`payment\_u\_count` bigint COMMENT '支付人数',

`order2payment\_convert\_ratio` decimal(10,2) COMMENT '下单到支付的转化率'

) COMMENT '用户行为漏斗分析'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_user\_action\_convert\_day/';

2）数据装载

hive (gmall)>

insert into table ads\_user\_action\_convert\_day

select

'2020-03-10',

uv.day\_count,

ua.cart\_count,

cast(ua.cart\_count/uv.day\_count as decimal(10,2)) visitor2cart\_convert\_ratio,

ua.order\_count,

cast(ua.order\_count/ua.cart\_count as decimal(10,2)) visitor2order\_convert\_ratio,

ua.payment\_count,

cast(ua.payment\_count/ua.order\_count as decimal(10,2)) order2payment\_convert\_ratio

from

(

select

dt,

sum(if(cart\_count>0,1,0)) cart\_count,

sum(if(order\_count>0,1,0)) order\_count,

sum(if(payment\_count>0,1,0)) payment\_count

from dws\_user\_action\_daycount

where dt='2020-03-10'

group by dt

)ua join ads\_uv\_count uv on uv.dt=ua.dt;

3）查询加载数据

hive (gmall)> select \* from ads\_user\_action\_convert\_day;

## 7.3 商品主题

### 7.3.1 商品个数信息

1）建表语句

hive (gmall)>

drop table if exists ads\_product\_info;

create external table ads\_product\_info(

`dt` string COMMENT '统计日期',

`sku\_num` string COMMENT 'sku个数',

`spu\_num` string COMMENT 'spu个数'

) COMMENT '商品个数信息'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_product\_info';

2）导入数据

hive (gmall)>

insert into table ads\_product\_info

select

'2020-03-10' dt,

sku\_num,

spu\_num

from

(

select

'2020-03-10' dt,

count(\*) sku\_num

from

dwt\_sku\_topic

) tmp\_sku\_num

join

(

select

'2020-03-10' dt,

count(\*) spu\_num

from

(

select

spu\_id

from

dwt\_sku\_topic

group by

spu\_id

) tmp\_spu\_id

) tmp\_spu\_num

on

tmp\_sku\_num.dt=tmp\_spu\_num.dt;

3）查询结果数据

hive (gmall)> select \* from ads\_product\_info;

### 7.3.2 商品销量排名

1）建表语句

hive (gmall)>

drop table if exists ads\_product\_sale\_topN;

create external table ads\_product\_sale\_topN(

`dt` string COMMENT '统计日期',

`sku\_id` string COMMENT '商品ID',

`payment\_amount` bigint COMMENT '销量'

) COMMENT '商品个数信息'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_product\_sale\_topN';

2）导入数据

hive (gmall)>

insert into table ads\_product\_sale\_topN

select

'2020-03-10' dt,

sku\_id,

payment\_amount

from

dws\_sku\_action\_daycount

where

dt='2020-03-10'

order by payment\_amount desc

limit 10;

3）查询结果数据

hive (gmall)> select \* from ads\_product\_sale\_topN;

### 7.3.3 商品收藏排名

1）建表语句

hive (gmall)>

drop table if exists ads\_product\_favor\_topN;

create external table ads\_product\_favor\_topN(

`dt` string COMMENT '统计日期',

`sku\_id` string COMMENT '商品ID',

`favor\_count` bigint COMMENT '收藏量'

) COMMENT '商品收藏TopN'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_product\_favor\_topN';

2）导入数据

hive (gmall)>

insert into table ads\_product\_favor\_topN

select

'2020-03-10' dt,

sku\_id,

favor\_count

from

dws\_sku\_action\_daycount

where

dt='2020-03-10'

order by favor\_count desc

limit 10;

3）查询数据

hive (gmall)> select \* from ads\_product\_favor\_topN;

### 7.3.4 商品加入购物车排名

1）建表语句

hive (gmall)>

drop table if exists ads\_product\_cart\_topN;

create external table ads\_product\_cart\_topN(

`dt` string COMMENT '统计日期',

`sku\_id` string COMMENT '商品ID',

`cart\_num` bigint COMMENT '加入购物车数量'

) COMMENT '商品加入购物车TopN'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_product\_cart\_topN';

2）导入数据

hive (gmall)>

insert into table ads\_product\_cart\_topN

select

'2020-03-10' dt,

sku\_id,

cart\_num

from

dws\_sku\_action\_daycount

where

dt='2020-03-10'

order by cart\_num desc

limit 10;

3）查询数据

hive (gmall)> select \* from ads\_product\_cart\_topN;

### 7.3.5 商品退款率排名(最近30天)

1）建表语句

hive (gmall)>

drop table if exists ads\_product\_refund\_topN;

create external table ads\_product\_refund\_topN(

`dt` string COMMENT '统计日期',

`sku\_id` string COMMENT '商品ID',

`refund\_ratio` decimal(10,2) COMMENT '退款率'

) COMMENT '商品退款率TopN'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_product\_refund\_topN';

2）导入数据

hive (gmall)>

insert into table ads\_product\_refund\_topN

select

'2020-03-10',

sku\_id,

refund\_last\_30d\_count/payment\_last\_30d\_count\*100 refund\_ratio

from dwt\_sku\_topic

order by refund\_ratio desc

limit 10;

3）查询数据

hive (gmall)> select \* from ads\_product\_refund\_topN;

### 7.3.6 商品差评率

1）建表语句

hive (gmall)>

drop table if exists ads\_appraise\_bad\_topN;

create external table ads\_appraise\_bad\_topN(

`dt` string COMMENT '统计日期',

`sku\_id` string COMMENT '商品ID',

`appraise\_bad\_ratio` decimal(10,2) COMMENT '差评率'

) COMMENT '商品差评率TopN'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_appraise\_bad\_topN';

2）导入数据

hive (gmall)>

insert into table ads\_appraise\_bad\_topN

select

'2020-03-10' dt,

sku\_id,

appraise\_bad\_count/(appraise\_good\_count+appraise\_mid\_count+appraise\_bad\_count+appraise\_default\_count) appraise\_bad\_ratio

from

dws\_sku\_action\_daycount

where

dt='2020-03-10'

order by appraise\_bad\_ratio desc

limit 10;

3）查询数据

hive (gmall)> select \* from ads\_appraise\_bad\_topN;

## 7.4 营销主题（用户+商品+购买行为）

### 7.4.1 下单数目统计

需求分析：统计每日下单数，下单金额及下单用户数。

1）建表语句

hive (gmall)>

drop table if exists ads\_order\_daycount;

create external table ads\_order\_daycount(

dt string comment '统计日期',

order\_count bigint comment '单日下单笔数',

order\_amount bigint comment '单日下单金额',

order\_users bigint comment '单日下单用户数'

) comment '每日订单总计表'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_order\_daycount';

2）导入数据

hive (gmall)>

insert into table ads\_order\_daycount

select

'2020-03-10' dt,

order\_count,

order\_amount,

order\_users

from

(

select

'2020-03-10' dt,

sum(order\_count) order\_count,

sum(order\_amount) order\_amount

from

dws\_sku\_action\_daycount

where

dt='2020-03-10'

) tmp\_order\_num

join

(

select

'2020-03-10' dt,

sum(if(order\_count>0,1,0)) order\_users

from

dws\_user\_action\_daycount

where

dt='2020-03-10'

) tmp\_order\_users

on

tmp\_order\_num.dt=tmp\_order\_users.dt;

3）查询数据

hive (gmall)> select \* from ads\_order\_daycount;

### 7.4.2 支付信息统计

每日支付金额、支付人数、支付商品数、支付笔数以及下单到支付的平均时长（取自DWD）

1）建表

hive (gmall)>

drop table if exists ads\_payment\_daycount;

create external table ads\_payment\_daycount(

dt string comment '统计日期',

order\_count bigint comment '单日支付笔数',

order\_amount bigint comment '单日支付金额',

payment\_user\_count bigint comment '单日支付人数',

payment\_sku\_count bigint comment '单日支付商品数',

payment\_avg\_time double comment '下单到支付的平均时长，取分钟数'

) comment '每日订单总计表'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_payment\_daycount';

2）导入数据

hive (gmall)>

insert into table ads\_payment\_daycount

select

tmp\_payment.dt,

tmp\_payment.payment\_count,

tmp\_payment.payment\_amount,

tmp\_payment.payment\_user\_count,

tmp\_skucount.payment\_sku\_count,

tmp\_time.payment\_avg\_time

from

(

select

'2020-03-15' dt,

sum(payment\_count) payment\_count,

sum(payment\_amount) payment\_amount,

sum(if(payment\_count>0,1,0)) payment\_user\_count

from dws\_user\_action\_daycount

where dt='2020-03-15'

)tmp\_payment

join

(

select

'2020-03-15' dt,

sum(if(payment\_count>0,1,0)) payment\_sku\_count

from dws\_sku\_action\_daycount

where dt='2020-03-15'

)tmp\_skucount on tmp\_payment.dt=tmp\_skucount.dt

join

(

select

'2020-03-15' dt,

sum(unix\_timestamp(payment\_time)-unix\_timestamp(create\_time))/count(\*)/60 payment\_avg\_time

from dwd\_fact\_order\_info

where dt='2020-03-15'

and payment\_time is not null

)tmp\_time on tmp\_payment.dt=tmp\_time.dt

3）查询数据

hive (gmall)> select \* from ads\_payment\_daycount;

### 7.4.3 复购率

1）建表语句

hive (gmall)>

drop table ads\_sale\_tm\_category1\_stat\_mn;

create external table ads\_sale\_tm\_category1\_stat\_mn

(

tm\_id string comment '品牌id',

category1\_id string comment '1级品类id ',

category1\_name string comment '1级品类名称 ',

buycount bigint comment '购买人数',

buy\_twice\_last bigint comment '两次以上购买人数',

buy\_twice\_last\_ratio decimal(10,2) comment '单次复购率',

buy\_3times\_last bigint comment '三次以上购买人数',

buy\_3times\_last\_ratio decimal(10,2) comment '多次复购率',

stat\_mn string comment '统计月份',

stat\_date string comment '统计日期'

) COMMENT '复购率统计'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_sale\_tm\_category1\_stat\_mn/';

2）数据导入

hive (gmall)>

insert into table ads\_sale\_tm\_category1\_stat\_mn

select

mn.sku\_tm\_id,

mn.sku\_category1\_id,

mn.sku\_category1\_name,

sum(if(mn.order\_count>=1,1,0)) buycount,

sum(if(mn.order\_count>=2,1,0)) buyTwiceLast,

sum(if(mn.order\_count>=2,1,0))/sum( if(mn.order\_count>=1,1,0)) buyTwiceLastRatio,

sum(if(mn.order\_count>=3,1,0)) buy3timeLast ,

sum(if(mn.order\_count>=3,1,0))/sum( if(mn.order\_count>=1,1,0)) buy3timeLastRatio ,

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

'2019-02-10' stat\_date

from

(

select

user\_id,

sd.sku\_tm\_id,

sd.sku\_category1\_id,

sd.sku\_category1\_name,

sum(order\_count) order\_count

from dws\_sale\_detail\_daycount sd

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

group by user\_id, sd.sku\_tm\_id, sd.sku\_category1\_id, sd.sku\_category1\_name

) mn

group by mn.sku\_tm\_id, mn.sku\_category1\_id, mn.sku\_category1\_name;

3）查询导入数据

hive (gmall)> select \* from ads\_sale\_tm\_category1\_stat\_mn;

4）数据导入脚本

（1）在/home/woaini/bin目录下创建脚本ads\_sale.sh

[woaini@hadoop102 bin]$ vim ads\_sale.sh

（2）在脚本中填写如下内容

#!/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\_sale\_tm\_category1\_stat\_mn

select

mn.sku\_tm\_id,

mn.sku\_category1\_id,

mn.sku\_category1\_name,

sum(if(mn.order\_count>=1,1,0)) buycount,

sum(if(mn.order\_count>=2,1,0)) buyTwiceLast,

sum(if(mn.order\_count>=2,1,0))/sum( if(mn.order\_count>=1,1,0)) buyTwiceLastRatio,

sum(if(mn.order\_count>=3,1,0)) buy3timeLast,

sum(if(mn.order\_count>=3,1,0))/sum( if(mn.order\_count>=1,1,0)) buy3timeLastRatio ,

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

'$do\_date' stat\_date

from

(

select

user\_id,

od.sku\_tm\_id,

od.sku\_category1\_id,

od.sku\_category1\_name,

sum(order\_count) order\_count

from "$APP".dws\_sale\_detail\_daycount od

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

group by user\_id, od.sku\_tm\_id, od.sku\_category1\_id, od.sku\_category1\_name

) mn

group by mn.sku\_tm\_id, mn.sku\_category1\_id, mn.sku\_category1\_name;

"

$hive -e "$sql"

（3）增加脚本执行权限

[woaini@hadoop102 bin]$ chmod 777 ads\_sale.sh

（4）执行脚本导入数据

[woaini@hadoop102 bin]$ ads\_sale.sh 2019-03-11

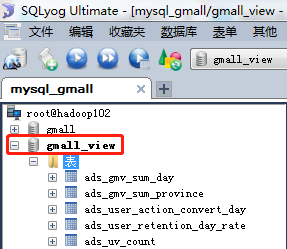
（5）查看导入数据

hive (gmall)>select \* from ads\_sale\_tm\_category1\_stat\_mn limit 2;

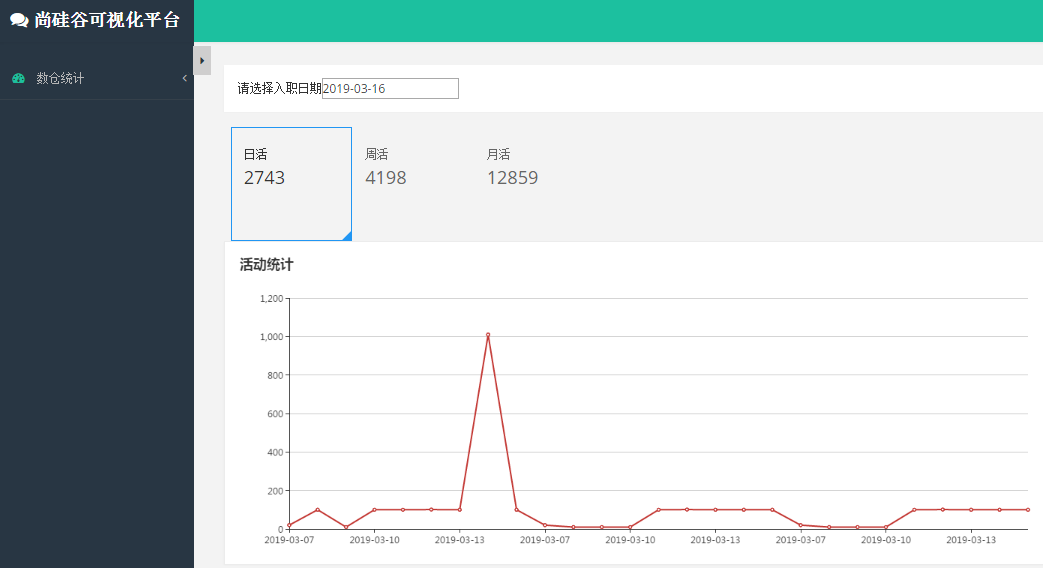
# 第8章 数据可视化

## 8.1 在MySQL中创建表

在hadoop102的MySql中创建gmall\_view数据库。



### 8.1.1 每日活跃统计



1）在MySQL中创建ads\_uv\_count表

DROP TABLE IF EXISTS `ads\_uv\_count`;

CREATE TABLE `ads\_uv\_count` (

`dt` varchar(255) PRIMARY KEY COMMENT '统计日期',

`day\_count` bigint(200) DEFAULT NULL COMMENT '当日用户数量',

`wk\_count` bigint(200) DEFAULT NULL COMMENT '当周用户数量',

`mn\_count` bigint(200) DEFAULT NULL COMMENT '当月用户数量',

`is\_weekend` varchar(200) CHARACTER SET utf8 COLLATE utf8\_general\_ci DEFAULT NULL COMMENT 'Y,N是否是周末,用于得到本周最终结果',

`is\_monthend` varchar(200) CHARACTER SET utf8 COLLATE utf8\_general\_ci DEFAULT NULL COMMENT 'Y,N是否是月末,用于得到本月最终结果'

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '每日活跃用户数量' ROW\_FORMAT = Dynamic;

2）向MySQL中插入如下数据

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-01 14:10:04', 20, 30, 100, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-02 14:12:48', 35, 50, 100, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-03 14:14:07', 25, 640, 3300, 'Y', 'Y');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-04 14:14:14', 10, 23, 123, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-05 14:14:21', 80, 121, 131, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-06 14:14:38', 30, 53, 453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-07 14:33:27', 20, 31, 453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-08 14:33:39', 10, 53, 453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-09 14:33:47', 10, 34, 453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-10 14:33:54', 10, 653, 8453, 'Y', 'Y');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-11 14:34:04', 100, 453, 1453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-12 14:34:10', 101, 153, 134, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-13 14:34:16', 100, 286, 313, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-14 14:34:22', 100, 45, 453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-15 14:34:29', 100, 345, 3453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-16 14:34:35', 101, 453, 453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-17 14:34:41', 100, 678, 9812, 'Y', 'Y');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-18 14:34:46', 100, 186, 193, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-19 14:34:53', 453, 686, 712, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-20 14:34:57', 452, 786, 823, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-21 14:35:02', 214, 58, 213, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-22 14:35:08', 76, 78, 95, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-23 14:35:13', 76, 658, 745, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-24 14:35:19', 76, 687, 9300, 'Y', 'Y');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-25 14:35:25', 76, 876, 923, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-26 14:35:30', 76, 456, 511, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-27 14:35:35', 76, 456, 623, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-28 14:35:41', 43, 753, 4000, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-29 14:35:47', 76, 876, 4545, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-30 14:35:57', 76, 354, 523, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2020-03-31 14:36:02', 43, 634, 6213, 'Y', 'Y');

### 8.1.2 留存率统计

1）在MySQL中创建ads\_user\_retention\_day\_rate表

DROP TABLE IF EXISTS `ads\_user\_retention\_day\_rate`;

CREATE TABLE `ads\_user\_retention\_day\_rate` (

`stat\_date` varchar(255) PRIMARY KEY COMMENT '统计日期',

`create\_date` varchar(255) DEFAULT NULL COMMENT '设备新增日期',

`retention\_day` bigint(200) DEFAULT NULL COMMENT '截止当前日期留存天数',

`retention\_count` bigint(200) DEFAULT NULL COMMENT '留存数量',

`new\_mid\_count` bigint(200) DEFAULT NULL COMMENT '当日设备新增数量',

`retention\_ratio` decimal(10, 2) DEFAULT NULL COMMENT '留存率'

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '每日用户留存情况' ROW\_FORMAT = Dynamic;

2）向MySQL中插入数据

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-09','2020-03-08', 1,88, 99, 0.78);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-10','2020-03-08', 2,77, 88, 0.68);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-11','2020-03-08', 3,66, 77, 0.58);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-12','2020-03-08', 4,55, 66, 0.48);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-13','2020-03-08', 5,44, 55, 0.38);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-14','2020-03-08', 6,33, 44, 0.28);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-10','2020-03-09', 1,77, 88, 0.56);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-11','2020-03-09', 2,66, 77, 0.46);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-12','2020-03-09', 3,55, 66, 0.36);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-13','2020-03-09', 4,44, 55, 0.26);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-14','2020-03-09', 5,33, 44, 0.16);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-11','2020-03-10', 1,66, 77, 0.55);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-12','2020-03-10', 2,55, 66, 0.45);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-13','2020-03-10', 3,44, 55, 0.35);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2020-03-14','2020-03-10', 4,33, 44, 0.25);

### 8.1.3 漏斗分析



1）在MySQL中创建ads\_user\_action\_convert\_day表

DROP TABLE IF EXISTS `ads\_user\_action\_convert\_day`;

CREATE TABLE `ads\_user\_action\_convert\_day` (

`dt` varchar(200) PRIMARY KEY COMMENT '统计日期',

`total\_visitor\_m\_count` bigint(20) DEFAULT NULL COMMENT '总访问人数',

`order\_u\_count` bigint(20) DEFAULT NULL COMMENT '下单人数',

`visitor2order\_convert\_ratio` decimal(10, 2) DEFAULT NULL COMMENT '购物车到下单转化率',

`payment\_u\_count` bigint(20) DEFAULT NULL COMMENT '支付人数',

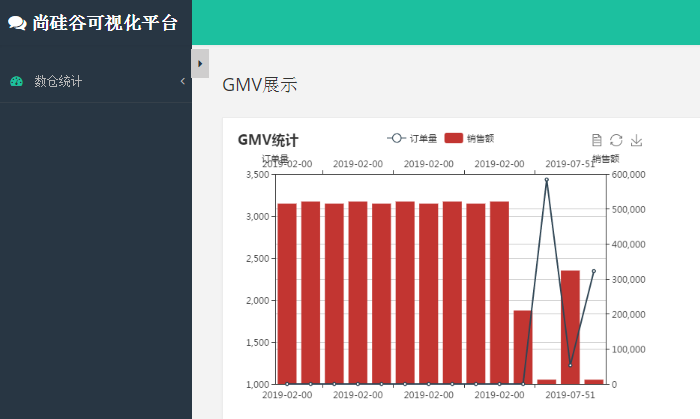
`order2payment\_convert\_ratio` decimal(10, 2) DEFAULT NULL COMMENT '下单到支付的转化率'

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '每日用户行为转化率统计' ROW\_FORMAT = Dynamic;

2）向MySQL中插入数据

INSERT INTO `ads\_user\_action\_convert\_day` VALUES ('2020-04-28 19:36:18', 10000, 3000, 0.25, 2000, 0.15);

### 8.1.4 GMV统计



1）在MySQL中创建ads\_gmv\_sum\_day表

DROP TABLE IF EXISTS ads\_gmv\_sum\_day;

CREATE TABLE ads\_gmv\_sum\_day(

`dt` varchar(200) PRIMARY KEY COMMENT '统计日期',

`gmv\_count` bigint(20) DEFAULT NULL COMMENT '当日gmv订单个数',

`gmv\_amount` decimal(16, 2) DEFAULT NULL COMMENT '当日gmv订单总金额',

`gmv\_payment` decimal(16, 2) DEFAULT NULL COMMENT '当日支付金额'

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '每日活跃用户数量' ROW\_FORMAT = Dynamic;

2）向MySQL中插入数据

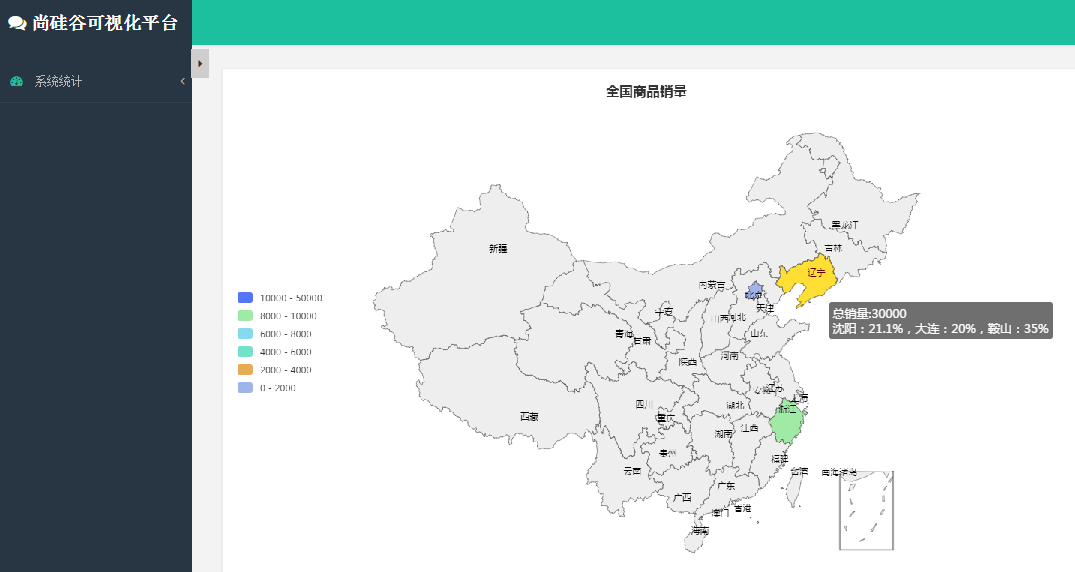
INSERT INTO `ads\_gmv\_sum\_day` VALUES ('2020-03-01 22:51:37', 1000, 210000.00, 2000.00);

INSERT INTO `ads\_gmv\_sum\_day` VALUES ('2020-05-08 22:52:32', 3434, 12413.00, 1.00);

INSERT INTO `ads\_gmv\_sum\_day` VALUES ('2020-07-13 22:52:51', 1222, 324345.00, 1.00);

INSERT INTO `ads\_gmv\_sum\_day` VALUES ('2020-09-13 22:53:08', 2344, 12312.00, 1.00);

### 8.1.5 全国商品销售



1）在MySQL中创建ads\_gmv\_sum\_province表

DROP TABLE IF EXISTS `ads\_gmv\_sum\_province`;

CREATE TABLE `ads\_gmv\_sum\_province` (

`province` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci DEFAULT NULL,

`gmv` bigint(255) DEFAULT NULL,

`remark` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci DEFAULT NULL

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci ROW\_FORMAT = Dynamic;

2）向MySQL中插入数据

INSERT INTO `ads\_gmv\_sum\_province` VALUES ('北京', 2000, '');

INSERT INTO `ads\_gmv\_sum\_province` VALUES ('辽宁', 30000, '沈阳：21.1%，大连：20%，鞍山：35%');

INSERT INTO `ads\_gmv\_sum\_province` VALUES ('浙江', 8002, '杭州：20%，舟山：50%');

### 8.1.6 会员主题信息

1）在MySQL中创建ads\_user\_topic表

DROP TABLE IF EXISTS `ads\_user\_topic`;

CREATE TABLE `ads\_user\_topic`(

`stat\_date` varchar(255) PRIMARY KEY COMMENT '统计日期',

`day\_users` bigint(200) DEFAULT NULL COMMENT '活跃会员数',

`day\_new\_users` bigint(200) DEFAULT NULL COMMENT '新增会员数',

`day\_new\_payment\_users` bigint(200) DEFAULT NULL COMMENT '新增消费会员数',

`payment\_users` bigint(200) DEFAULT NULL COMMENT '总付费会员数',

`users` bigint(200) DEFAULT NULL COMMENT '总会员数',

`day\_users2users` decimal(10,2) DEFAULT NULL COMMENT '会员活跃率',

`payment\_users2users` decimal(10,2) DEFAULT NULL COMMENT '会员付费率',

`day\_new\_users2users` decimal(10,2) DEFAULT NULL COMMENT '会员新鲜度'

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '每日会员信息统计' ROW\_FORMAT = Dynamic;

## 8.2 WEB页面查看

1）运行spring-boot-echarts-master程序

2）在web页面上查看显示结果

http://localhost:8080/active

## 8.3 Sqoop导出脚本

1）编写Sqoop导出脚本

在/home/woaini/bin目录下创建脚本hiveTomysql.sh

[woaini@hadoop102 bin]$ vim hiveTomysql.sh

在脚本中填写如下内容

#!/bin/bash

hive\_db\_name=gmall

mysql\_db\_name=gmall\_view

export\_data() {

/opt/module/sqoop/bin/sqoop export \

--connect "jdbc:mysql://hadoop102:3306/${mysql\_db\_name}?useUnicode=true&characterEncoding=utf-8" \

--username root \

--password 000000 \

--table $1 \

--num-mappers 1 \

--export-dir /warehouse/$hive\_db\_name/ads/$1 \

--input-fields-terminated-by "\t" \

--update-mode allowinsert \

--update-key $2 \

--input-null-string '\\N' \

--input-null-non-string '\\N'

}

case $1 in

"ads\_uv\_count")

export\_data "ads\_uv\_count" "dt"

;;

"ads\_user\_action\_convert\_day")

export\_data "ads\_user\_action\_convert\_day" "dt"

;;

"ads\_gmv\_sum\_day")

export\_data "ads\_gmv\_sum\_day" "dt"

;;

"ads\_user\_topic")

export\_data "ads\_user\_topic" "dt"

;;

"all")

export\_data "ads\_uv\_count" "dt"

export\_data "ads\_user\_action\_convert\_day" "dt"

export\_data "ads\_gmv\_sum\_day" "dt"

export\_data "ads\_user\_topic" "dt"

;;

esac

 关于导出update还是insert的问题

* --update-mode：

updateonly   只更新，无法插入新数据

        allowinsert   允许新增

* --update-key：允许更新的情况下，指定哪些字段匹配视为同一条数据，进行更新而不增加。多个字段用逗号分隔。
* --input-null-string和--input-null-non-string：

分别表示，将字符串列和非字符串列的空串和“null”转义。

官网地址：http://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html

Sqoop will by default import NULL values as string null. Hive is however using string \N to denote NULL values and therefore predicates dealing with NULL(like IS NULL) will not work correctly. You should append parameters --null-string and --null-non-string in case of import job or --input-null-string and --input-null-non-string in case of an export job if you wish to properly preserve NULL values. Because sqoop is using those parameters in generated code, you need to properly escape value \N to [\\N](file:///\\\\N):

Hive中的Null在底层是以“\N”来存储，而MySQL中的Null在底层就是Null，为了保证数据两端的一致性。在导出数据时采用--input-null-string和--input-null-non-string两个参数。导入数据时采用--null-string和--null-non-string。

3）执行Sqoop导出脚本

[woaini@hadoop102 bin]$ chmod 777 hiveTomysql.sh

[woaini@hadoop102 bin]$ hiveTomysql.sh all

4）在MySQL中查看结果

SELECT \* FROM ads\_uv\_count;

SELECT \* FROM ads\_user\_retention\_day\_rate;

SELECT \* FROM ads\_user\_action\_convert\_day;

SELECT \* FROM ads\_gmv\_sum\_day;

SELECT \* FROM ads\_gmv\_sum\_province;

# 第9章 任务调度器（Azkaban）

## 9.1 Azkaban安装

详见：hi大数据技术之Azkaban



## 9.2 会员主题指标获取的全调度流程

1）修改配置文件中时间，生成数据

[woaini@hadoop102 db\_log]$ java -jar gmall-mock-db.jar

2）编写Azkaban程序运行job

（1）import.job文件

type=command

do\_date=${dt}

command=/home/woaini/bin/sqoop\_import.sh all ${do\_date}

（2）ods.job文件

type=command

do\_date=${dt}

dependencies=import

command=/home/woaini/bin/hdfs\_to\_ods\_db.sh ${do\_date}

（3）dwd.job文件

type=command

do\_date=${dt}

dependencies=ods

command=/home/woaini/bin/ods\_to\_dwd\_db.sh ${do\_date}

（4）dws.job文件

type=command

do\_date=${dt}

dependencies=dwd

command=/home/woaini/bin/dwd\_to\_dws.sh ${do\_date}

（5）dwt.job文件

type=command

do\_date=${dt}

dependencies=dws

command=/home/woaini/bin/dws\_to\_dwt.sh ${do\_date}

（6）ads.job文件

type=command

do\_date=${dt}

dependencies=dwt

command=/home/woaini/bin/ads\_user\_topic.sh ${do\_date}

（7）export.job文件

type=command

dependencies=ads

do\_date=${dt}

command=/home/woaini/bin/hiveTomysql.sh ads\_user\_topic ${do\_date}

（8）将以上6个文件压缩成user-topic-job.zip文件

3）创建Azkaban工程，并上传user-topic-job.zip文件。

4）在浏览器中输入**https://hadoop102:8443，**并在页面上创建工程执行gmv-job.zip任务。

5）等待大约20分钟，在MySQL中查看结果

select \* from ads\_user\_topic;

# 第10章 需求实施流程

以下是**活跃用户**需求的整体开发流程。

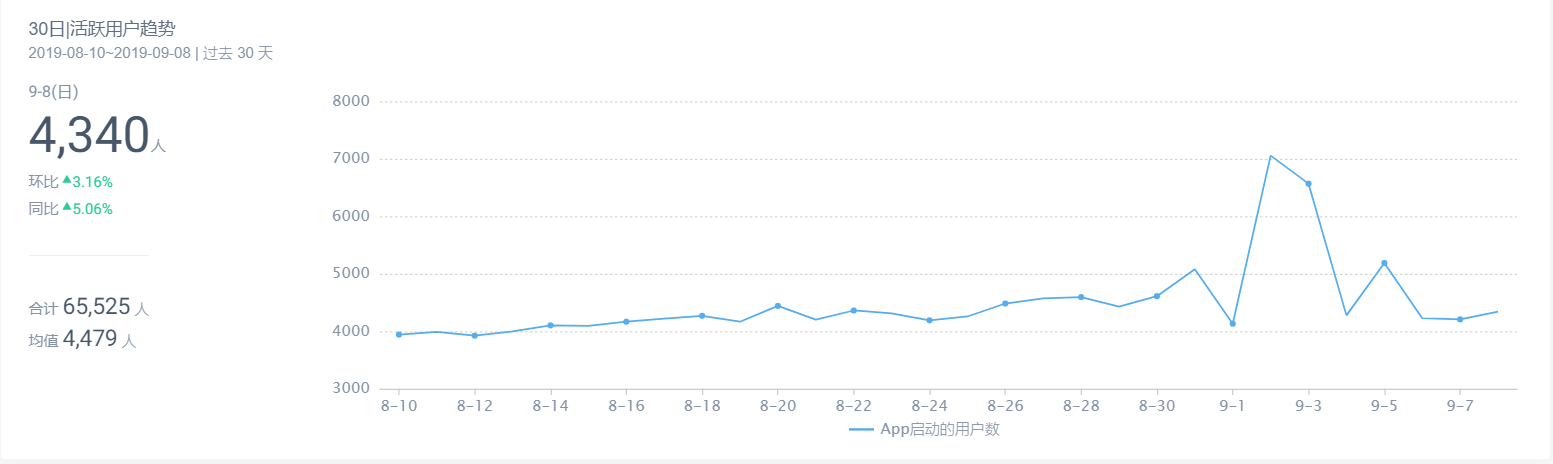
**第1步：确定指标的业务口径**

由产品经理主导，找到提出该指标的运营负责人沟通。首先要问清楚**指标是怎么定义的**，比如活跃用户是指启动过APP的用户。

沟通方式：邮件/需求文档-》不要口头

**第2步：需求评审**

由产品经理主导设计原型，对于活跃主题，我们最终要展示的是**最近n天的活跃用户数变化趋势** ，效果如下图所示。此处大数据开发工程师、后端开发工程师、前端开发工程师一同参与，一起说明整个功能的价值和详细的操作流程，确保大家理解的一致。



**第3步：大数据开发**

大数据开发工程师，通过数据同步的工具如Flume、Sqoop等将数据同步到ODS层，然后就是一层一层的通过SQL计算到DWD、DWS层，最后形成可为应用直接服务的数据填充到ADS层。

**第4步：后端开发**

后端工程师负责，为大数据工程师提供业务数据接口；

同时还负责读取ADS层分析后，写入MySQL中的数据。

**第5步：前端开发**

前端工程师负责，前端埋点。

对分析后的结果数据进行可视化展示。

**第6步：联调**

此时数据开发工程师、前端开发工程师、后端开发工程师都要参与进来。此时会要求大数据开发工程师基于历史的数据执行计算任务，大数据开发工程师承担数据准确性的校验。前后端解决用户操作的相关BUG保证不出现低级的问题完成自测。

**第7步：测试**

测试工程师对整个大数据系统进行测试。测试的手段包括，边界值、等价类等。

提交测试异常的软件有：禅道、bugzila

**第8步：上线**

运维工程师会配合我们的前后端开发工程师更新最新的版本到服务器。此时产品经理要找到该指标的负责人长期跟进指标的准确性。重要的指标还要每过一个周期内部再次验证，从而保证数据的准确性。