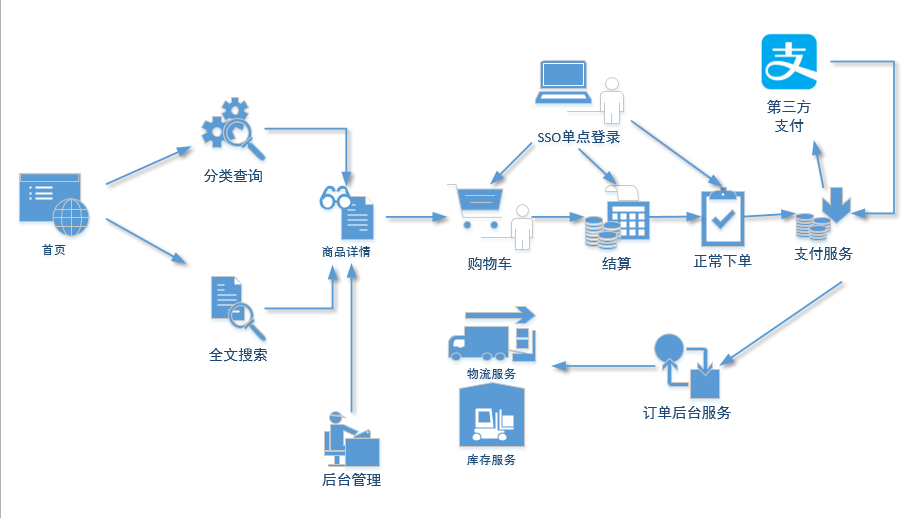
大数据项目之电商数仓（系统业务数据仓库）

(作者：杨晓凯)

版本：V1.0

# 第1章 电商业务与数据结构简介

## 1.1 电商业务流程



## 1.2 电商常识（SKU、SPU）

 SKU=Stock Keeping Unit（库存量基本单位）。现在已经被引申为产品统一编号的简称，每种产品均对应有唯一的SKU号。

 SPU(Standard Product Unit)：是商品信息聚合的最小单位，是一组可复用、易检索的标准化信息集合。

比如，咱们购买一台iPhoneX手机，iPhoneX手机就是一个SPU，但是你购买的时候，不可能是以iPhoneX手机为单位买的，商家也不可能以iPhoneX为单位记录库存SKU。必须要以什么颜色什么版本的iPhoneX为单位。比如，你购买的是一台银色、128G内存的、支持联通网络的iPhoneX，商家也会以这个单位来记录库存数。那这个更细致的单位就叫库存单元（SKU）。

那SPU又是干什么的呢？



SPU表示一类商品。好处就是：可以共用商品图片，海报、销售属性等。

## 1.3 电商表结构



### 1.3.1 订单表（order\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 订单编号 | |
| total\_amount | 订单金额 | |
| order\_status | 订单状态 | |
| user\_id | 用户id | |
| payment\_way | 支付方式 | |
| out\_trade\_no | 支付流水号 | |
| create\_time | 创建时间 | |
| operate\_time | 操作时间 | |

### 1.3.2 订单详情表（order\_detail）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 订单编号 | |
| order\_id | 订单号 | |
| user\_id | 用户id | |
| sku\_id | 商品id | |
| sku\_name | 商品名称 | |
| order\_price | 商品价格 | |
| sku\_num | 商品数量 | |
| create\_time | 创建时间 | |

### 1.3.3 商品表

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | skuId | |
| spu\_id | spuid | |
| price | 价格 | |
| sku\_name | 商品名称 | |
| sku\_desc | 商品描述 | |
| weight | 重量 | |
| tm\_id | 品牌id | |
| category3\_id | 品类id | |
| create\_time | 创建时间 | |

### 1.3.4 用户表

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 用户id | |
| name | 姓名 | |
| birthday | 生日 | |
| gender | 性别 | |
| email | 邮箱 | |
| user\_level | 用户等级 | |
| create\_time | 创建时间 | |

### 1.3.5 商品一级分类表

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | id | |
| name | 名称 | |

### 1.3.6 商品二级分类表

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | id | |
| name | 名称 | |
| category1\_id | 一级品类id | |

### 1.3.7 商品三级分类表

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | id | |
| name | 名称 | |
| Category2\_id | 二级品类id | |

### 1.3.8 支付流水表

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 编号 | |
| out\_trade\_no | 对外业务编号 | |
| order\_id | 订单编号 | |
| user\_id | 用户编号 | |
| alipay\_trade\_no | 支付宝交易流水编号 | |
| total\_amount | 支付金额 | |
| subject | 交易内容 | |
| payment\_type | 支付类型 | |
| payment\_time | 支付时间 | |

# 第2章 数仓理论（面试重点）

## 2.1 表的分类

### 2.1.1 实体表

**实体表**，一般是指一个现实存在的业务对象，比如用户，商品，商家，销售员等等。

用户表：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用户id | 姓名 | 生日 | 性别 | 邮箱 | 用户等级 | 创建时间 |
| 1 | 张三 | 2011-11-11 | 男 | zs@163.com | 2 | 2018-11-11 |
| 2 | 李四 | 2011-11-11 | 女 | ls@163.com | 3 | 2018-11-11 |
| 3 | 王五 | 2011-11-11 | 中性 | ww@163.com | 1 | 2018-11-11 |
| … | … | … | … | … | … | … |

### 2.1.2 维度表

**维度表**，一般是指对应一些业务状态，编号的解释表。也可以称之为码表。

比如地区表，订单状态，支付方式，审批状态，商品分类等等。

订单状态表：

|  |  |
| --- | --- |
| 订单状态编号 | 订单状态名称 |
| 1 | 未支付 |
| 2 | 支付 |
| 3 | 发货中 |
| 4 | 已发货 |
| 5 | 已完成 |

商品分类表：

|  |  |
| --- | --- |
| 商品分类编号 | 分类名称 |
| 1 | 服装 |
| 2 | 保健 |
| 3 | 电器 |
| 4 | 图书 |

### 2.1.3 事务型事实表

**事务型事实表**，一般指随着业务发生不断产生的数据。特点是一旦发生不会再变化。

一般比如，交易流水，操作日志，出库入库记录等等。

交易流水表：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 对外业务编号 | 订单编号 | 用户编号 | 支付宝交易流水编号 | 支付金额 | 交易内容 | 支付类型 | 支付时间 |
| 1 | 7577697945 | 1 | 111 | QEyF-63000323 | 223.00 | 海狗人参丸1 | alipay | 2019-02-10 00:50:02 |
| 2 | 0170099522 | 2 | 222 | qdwV-25111279 | 589.00 | 海狗人参丸2 | wechatpay | 2019-02-10 00:50:02 |
| 3 | 1840931679 | 3 | 666 | hSUS-65716585 | 485.00 | 海狗人参丸3 | unionpay | 2019-02-10 00:50:02 |
| 。。。 | 。。。 | 。。。 | 。。。 | 。。。 | 。。。 | 。。。 | 。。。 | 。。。 |

### 2.1.4 周期型事实表

**周期型事实表**，一般指随着业务发生不断产生的数据。

与事务型不同的是，数据会随着业务周期性的推进而变化。

 比如订单，其中订单状态会周期性变化。再比如，请假、贷款申请，随着批复状态在周期性变化。

订单表：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 订单编号 | 订单金额 | 订单状态 | 用户id | 支付方式 | 支付流水号 | 创建时间 | 操作时间 |
| 1 | 223.00 | 2 | 111 | alipay | QEyF-63000323 | 2019-02-10 00:01:29 | 2019-02-10 00:01:29 |
| 2 | 589.00 | 2 | 222 | wechatpay | qdwV-25111279 | 2019-02-10 00:05:02 | 2019-02-10 00:05:02 |
| 3 | 485.00 | 1 | 666 | unionpay | hSUS-65716585 | 2019-02-10 00:50:02 | 2019-02-10 00:50:02 |
| 。。。 | 。。。 | 。。。 | 。。。 | 。。。 | 。。。 | 。。。 | 。。。 |

## 2.2 同步策略

数据同步策略的类型包括：全量表、增量表、新增及变化表、拉链表

* 全量表：存储完整的数据。
* 增量表：存储新增加的数据。
* 新增及变化表：存储新增加的数据和变化的数据。
* 拉链表：对新增及变化表做定期合并。

### 2.2.1 实体表同步策略

实体表：比如用户，商品，商家，销售员等

实体表数据量比较小：通常可以做每日全量，就是每天存一份完整数据。即每日全量。

### 2.2.2 维度表同步策略

维度表：比如订单状态，审批状态，商品分类

维度表数据量比较小：通常可以做每日全量，就是每天存一份完整数据。即每日全量。

说明：

1）针对可能会有变化的状态数据可以存储每日全量。

2）没变化的客观世界的维度（比如性别，地区，民族，政治成分，鞋子尺码）可以只存一份固定值。

### 2.2.3 事务型事实表同步策略

事务型事实表：比如，交易流水，操作日志，出库入库记录等。

因为数据不会变化，而且数据量巨大，所以每天只同步新增数据即可，所以可以做成每日增量表，即每日创建一个分区存储。

### 2.2.4 周期型事实表同步策略

周期型事实表：比如，订单、请假、贷款申请等

这类表从数据量的角度，存每日全量的话，数据量太大，冗余也太大。如果用每日增量的话无法反应数据变化。

 每日新增及变化量，包括了当日的新增和修改。一般来说这个表，足够计算大部分当日数据的。但是这种依然无法解决能够得到某一个历史时间点（时间切片）的切片数据。

 所以要用利用每日新增和变化表，制作一张拉链表，以方便的取到某个时间切片的快照数据。所以我们需要得到每日新增及变化量。

拉链表：

|  |  |  |
| --- | --- | --- |
| name姓名 | start新名字创建时间 | end名字更改时间 |
| 张三 | 1990/1/1 | 2018/12/31 |
| 张小三 | 2019/1/1 | 2019/4/30 |
| 张大三 | 2019/5/1 | 9999-99-99 |
| 。。。 | 。。。 | 。。。 |

select \* from user where start =<’2019-1-2’ and end>=’2019-1-2’

## 2.3 范式理论

### 2.3.1 范式概念

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，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)。

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。

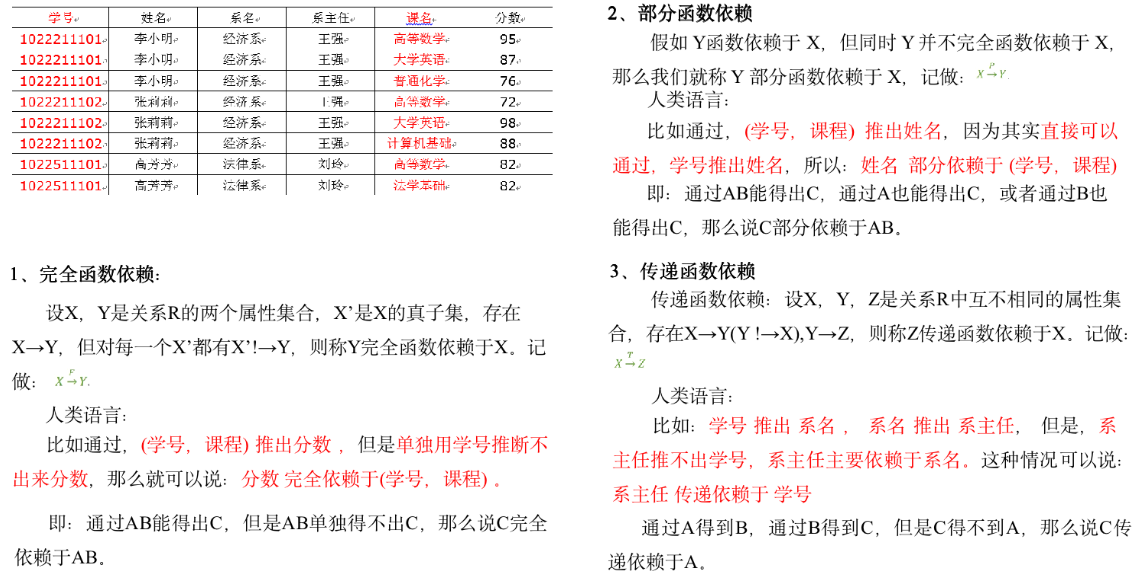
 使用范式的根本目的是：

  1）减少数据冗余，尽量让每个数据只出现一次。

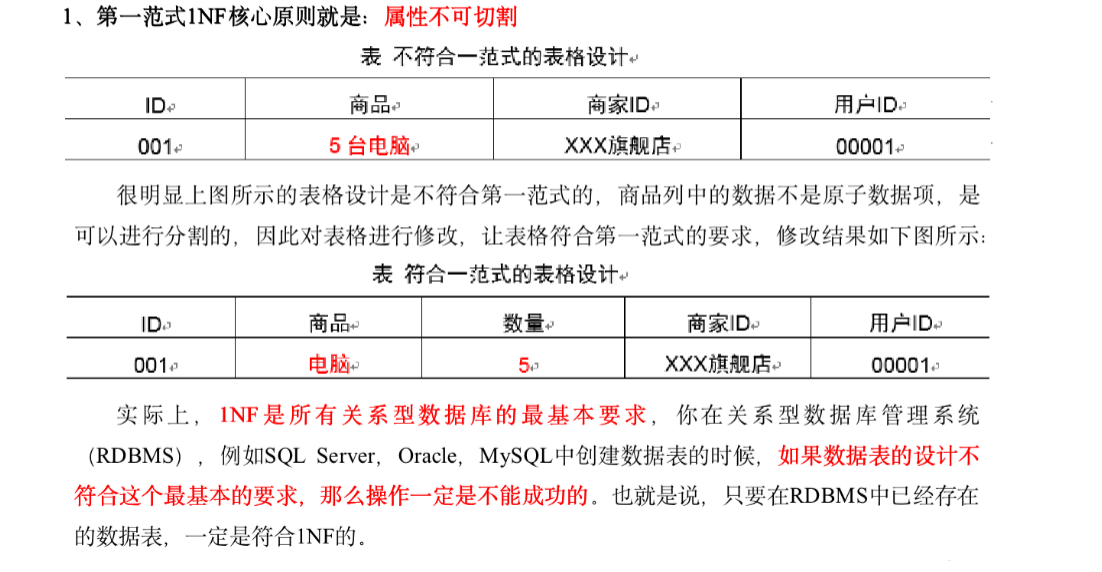
  2）保证数据一致性

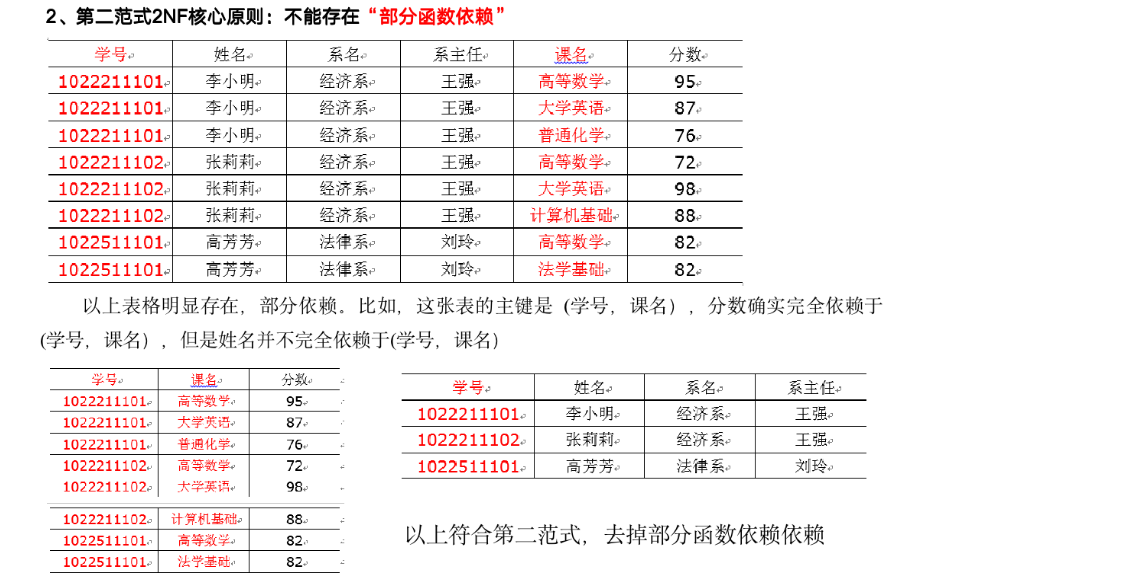
 缺点是获取数据时，需要通过Join拼接出最后的数据。

### 2.3.2 函数依赖



### 2.3.3 三范式区分

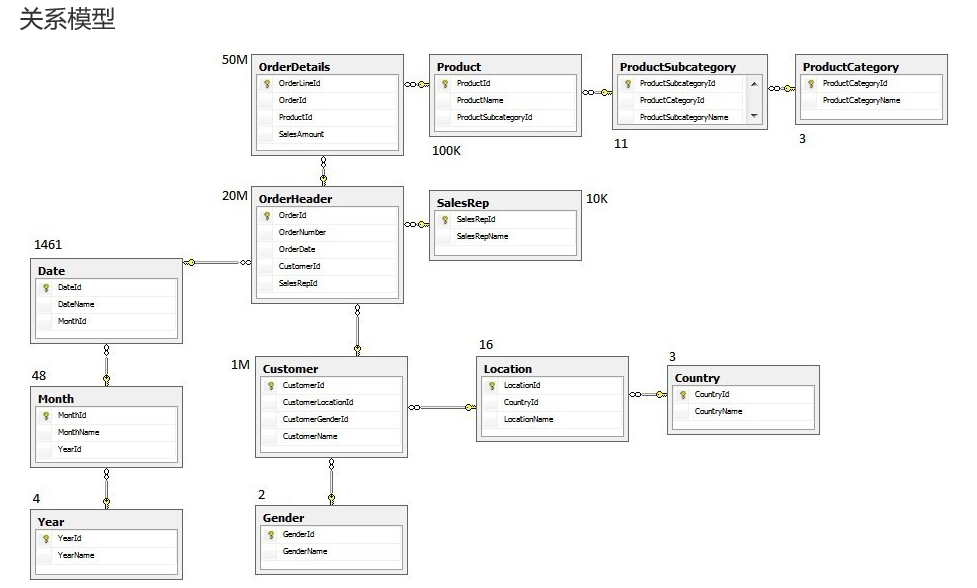






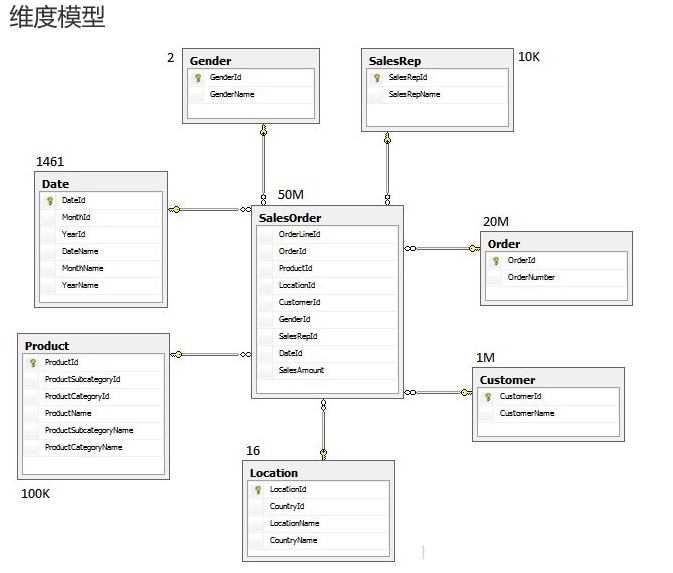
## 2.4 关系建模与维度建模

* 关系模型



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。

* 维度模型

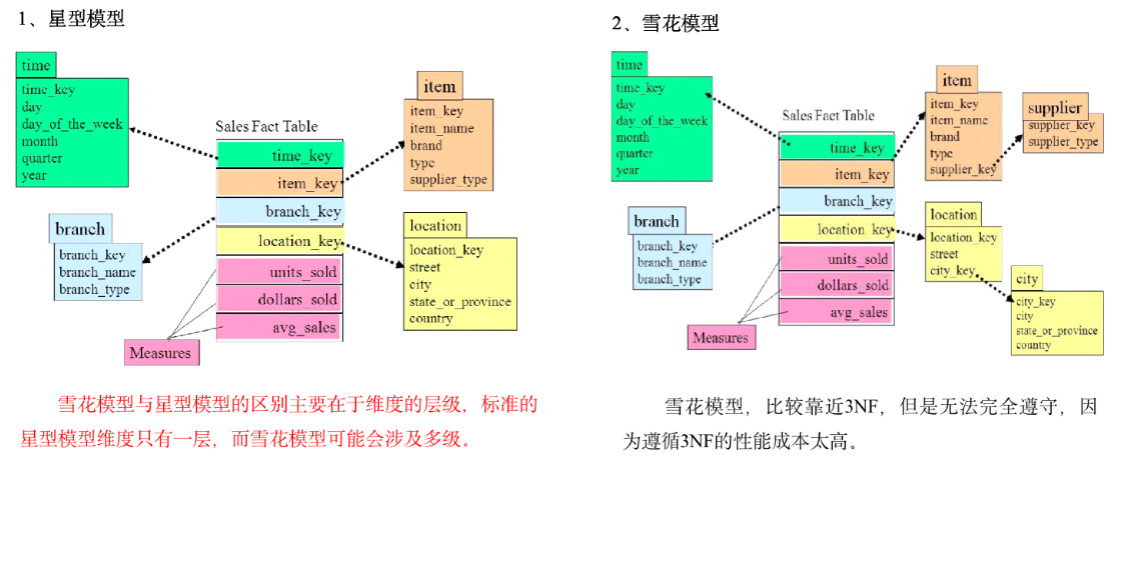


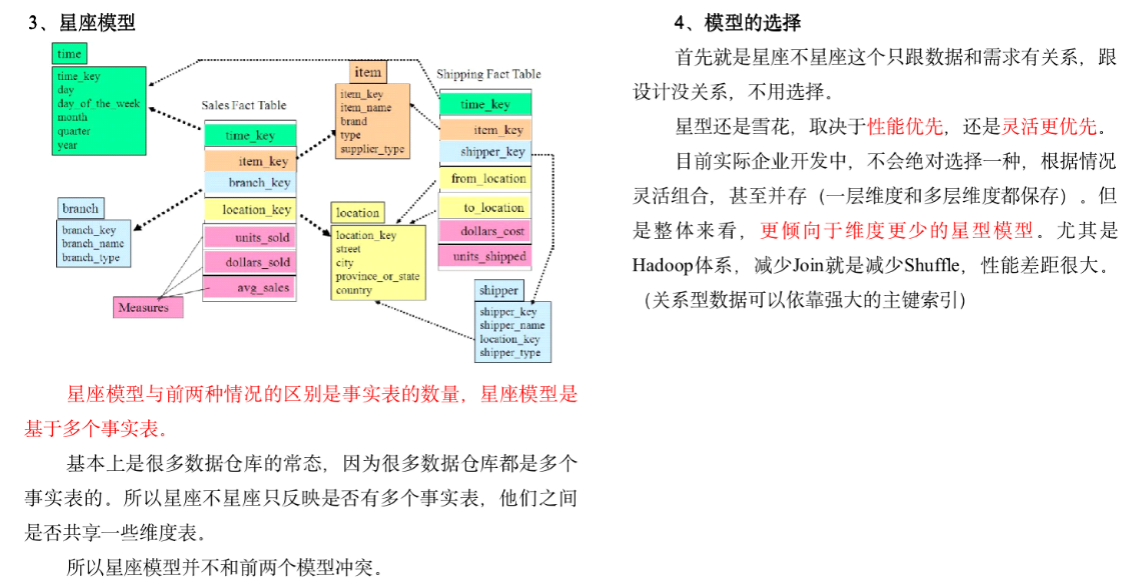
维度模型主要应用于OLAP系统中，因为关系模型虽然冗余少，但是在大规模数据，跨表分析统计查询过程中，会造成多表关联，这会大大降低执行效率。

所以把相关各种表整理成两种：事实表和维度表两种。所有维度表围绕着事实表进行解释。

## 2.5 雪花模型、星型模型和星座模型

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。





# 第3章 数仓搭建

## 3.0 配置Hadoop支持Snappy压缩

1）将编译后支持Snappy压缩的Hadoop jar包解压缩，并将lib/native目录中所有文件上传到hadoop102的/opt/module/hadoop-2.7.2/lib/native目录，并分发到hadoop103 hadoop104。

2）重新启动Hadoop。

3）检查支持的压缩方式

[bawei@hadoop102 native]$ hadoop checknative

hadoop: true /opt/module/hadoop-2.7.2/lib/native/libhadoop.so

zlib: true /lib64/libz.so.1

snappy: true /opt/module/hadoop-2.7.2/lib/native/libsnappy.so.1

lz4: true revision:99

bzip2: false

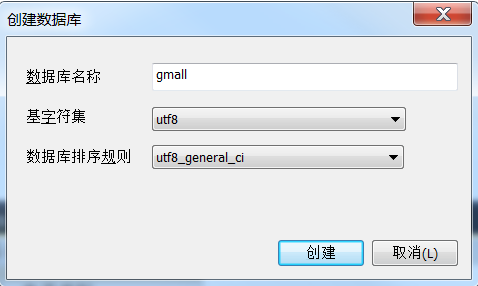
## 3.1 业务数据生成

### 3.1.1 建表语句

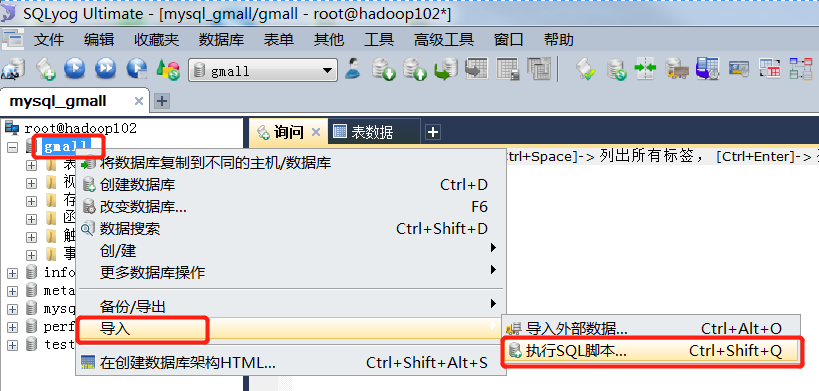
1）通过SQLyog创建数据库gmall



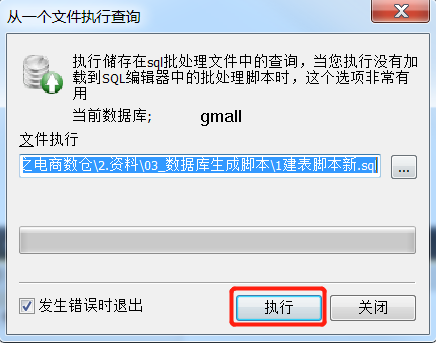
2）设置数据库编码



3）导入建表语句（1建表脚本）



选择->1建表脚本.sql



4）重复步骤3的导入方式，依次导入：2商品分类数据插入脚本、3函数脚本、4存储过程脚本。

### 3.1.2 生成业务数据

1）生成业务数据函数说明

init\_data ( do\_date\_string VARCHAR(20) , order\_incr\_num INT, user\_incr\_num INT , sku\_num INT , if\_truncate BOOLEAN )：

参数一：do\_date\_string生成数据日期

参数二：order\_incr\_num订单id个数

参数三：user\_incr\_num用户id个数

参数四：sku\_num商品sku个数

参数五：if\_truncate是否删除数据

2）案例测试：

（1）需求：生成日期2019年2月10日数据、订单1000个、用户200个、商品sku300个、删除原始数据。



CALL init\_data('2019-02-10',1000,200,300,TRUE);

（2）查询生成数据结果

SELECT \* from base\_category1;

SELECT \* from base\_category2;

SELECT \* from base\_category3;

SELECT \* from order\_info;

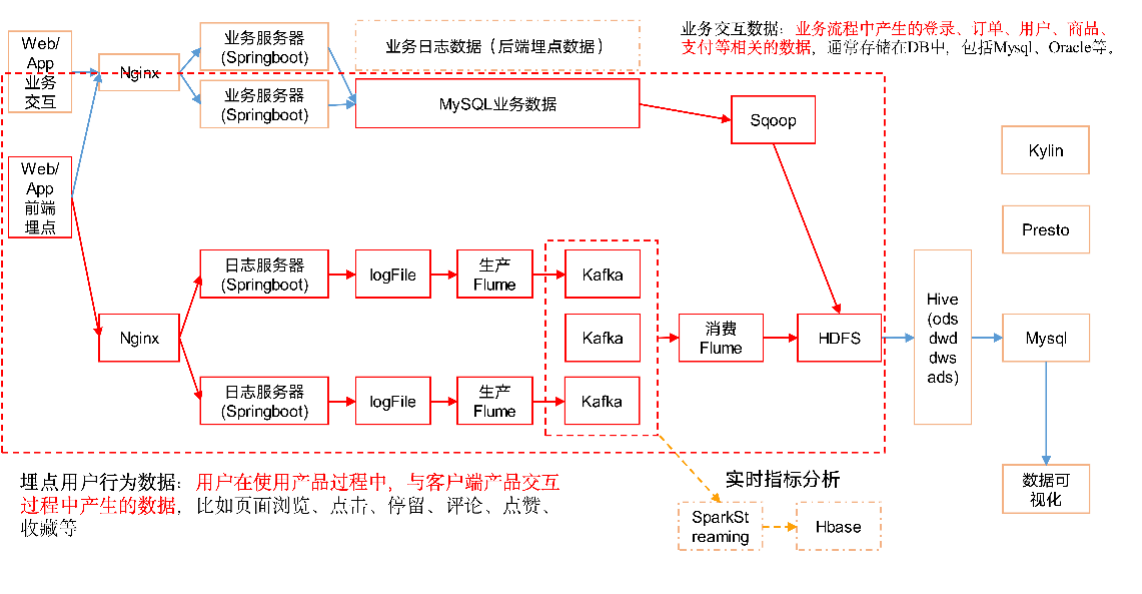
SELECT \* from order\_detail;

SELECT \* from sku\_info;

SELECT \* from user\_info;

SELECT \* from payment\_info;

## 3.2 业务数据导入数仓



### 3.2.1 Sqoop安装

详见大数据技术之Sqoop



### 3.2.2 Sqoop导入命令

/opt/module/sqoop/bin/sqoop import \

--connect \

--username \

--password \

--target-dir \

--delete-target-dir \

--num-mappers \

--fields-terminated-by \

--query "$2" ' and $CONDITIONS;'

### 3.2.3 分析表



### 3.2.4 Sqoop定时导入脚本

1）在/home/bawei/bin目录下创建脚本sqoop\_import.sh

[bawei@hadoop102 bin]$ vim sqoop\_import.sh

在脚本中填写如下内容

#!/bin/bash

db\_date=$2

echo $db\_date

db\_name=gmall

import\_data() {

/opt/module/sqoop/bin/sqoop import \

--connect jdbc:mysql://hadoop102:3306/$db\_name \

--username root \

--password 000000 \

--target-dir /origin\_data/$db\_name/db/$1/$db\_date \

--delete-target-dir \

--num-mappers 1 \

--fields-terminated-by "\t" \

--query "$2"' and $CONDITIONS;'

}

import\_sku\_info(){

import\_data "sku\_info" "select

id, spu\_id, price, sku\_name, sku\_desc, weight, tm\_id,

category3\_id, create\_time

from sku\_info where 1=1"

}

import\_user\_info(){

import\_data "user\_info" "select

id, name, birthday, gender, email, user\_level,

create\_time

from user\_info where 1=1"

}

import\_base\_category1(){

import\_data "base\_category1" "select

id, name from base\_category1 where 1=1"

}

import\_base\_category2(){

import\_data "base\_category2" "select

id, name, category1\_id from base\_category2 where 1=1"

}

import\_base\_category3(){

import\_data "base\_category3" "select id, name, category2\_id from base\_category3 where 1=1"

}

import\_order\_detail(){

import\_data "order\_detail" "select

od.id,

order\_id,

user\_id,

sku\_id,

sku\_name,

order\_price,

sku\_num,

o.create\_time

from order\_info o, order\_detail od

where o.id=od.order\_id

and DATE\_FORMAT(create\_time,'%Y-%m-%d')='$db\_date'"

}

import\_payment\_info(){

import\_data "payment\_info" "select

id,

out\_trade\_no,

order\_id,

user\_id,

alipay\_trade\_no,

total\_amount,

subject,

payment\_type,

payment\_time

from payment\_info

where DATE\_FORMAT(payment\_time,'%Y-%m-%d')='$db\_date'"

}

import\_order\_info(){

import\_data "order\_info" "select

id,

total\_amount,

order\_status,

user\_id,

payment\_way,

out\_trade\_no,

create\_time,

operate\_time

from order\_info

where (DATE\_FORMAT(create\_time,'%Y-%m-%d')='$db\_date' or DATE\_FORMAT(operate\_time,'%Y-%m-%d')='$db\_date')"

}

case $1 in

"base\_category1")

import\_base\_category1

;;

"base\_category2")

import\_base\_category2

;;

"base\_category3")

import\_base\_category3

;;

"order\_info")

import\_order\_info

;;

"order\_detail")

import\_order\_detail

;;

"sku\_info")

import\_sku\_info

;;

"user\_info")

import\_user\_info

;;

"payment\_info")

import\_payment\_info

;;

"all")

import\_base\_category1

import\_base\_category2

import\_base\_category3

import\_order\_info

import\_order\_detail

import\_sku\_info

import\_user\_info

import\_payment\_info

;;

esac

2）增加脚本执行权限

[bawei@hadoop102 bin]$ chmod 777 sqoop\_import.sh

3）执行脚本导入数据

[bawei@hadoop102 bin]$ sqoop\_import.sh all 2019-02-10

4）在SQLyog中生成2019年2月11日数据

CALL init\_data('2019-02-11',1000,200,300,TRUE);

5）执行脚本导入数据

[bawei@hadoop102 bin]$ sqoop\_import.sh all 2019-02-11

### 3.2.5 Sqoop导入数据异常处理

1）问题描述：执行Sqoop导入数据脚本时，发生如下异常

java.sql.SQLException: Streaming result set com.mysql.jdbc.RowDataDynamic@65d6b83b is still active. No statements may be issued when any streaming result sets are open and in use on a given connection. Ensure that you have called .close() on any active streaming result sets before attempting more queries.

at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:930)

at com.mysql.jdbc.MysqlIO.checkForOutstandingStreamingData(MysqlIO.java:2646)

at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:1861)

at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2101)

at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2548)

at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2477)

at com.mysql.jdbc.StatementImpl.executeQuery(StatementImpl.java:1422)

at com.mysql.jdbc.ConnectionImpl.getMaxBytesPerChar(ConnectionImpl.java:2945)

at com.mysql.jdbc.Field.getMaxBytesPerCharacter(Field.java:582)

2）问题解决方案：增加如下导入参数

--driver com.mysql.jdbc.Driver \

## 3.3 ODS层

完全仿照业务数据库中的表字段，一模一样的创建ODS层对应表。

### 3.3.1 创建订单表

hive (gmall)>

drop table if exists ods\_order\_info;

create external table ods\_order\_info (

  `id` string COMMENT '订单编号',

  `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',

    `payment\_way` string COMMENT '支付方式',

    `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'

) COMMENT '订单表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

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

;

### 3.3.2 创建订单详情表

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` string COMMENT '商品价格',

    `sku\_num` string COMMENT '商品数量',

    `create\_time` string COMMENT '创建时间'

) COMMENT '订单明细表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

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

;

### 3.3.3 创建商品表

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 '商品表'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

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

;

### 3.3.4 创建用户表

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'

) COMMENT '用户信息'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

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

;

### 3.3.5 创建商品一级分类表

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'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

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

;

### 3.3.6 创建商品二级分类表

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'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

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

;

### 3.3.7 创建商品三级分类表

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'

PARTITIONED BY (`dt` string)

row format delimited fields terminated by '\t'

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

;

### 3.3.8 创建支付流水表

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'

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

;

### 3.3.9 ODS层数据导入脚本

1）在/home/bawei/bin目录下创建脚本ods\_db.sh

[bawei@hadoop102 bin]$ vim 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 "$1" ] ;then

do\_date=$1

else

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

fi

sql="

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');

"

$hive -e "$sql"

2）增加脚本执行权限

[bawei@hadoop102 bin]$ chmod 777 ods\_db.sh

3）执行脚本导入数据

[bawei@hadoop102 bin]$ ods\_db.sh 2019-02-10

[bawei@hadoop102 bin]$ ods\_db.sh 2019-02-11

4）查询导入数据

hive (gmall)>

select \* from ods\_order\_info where dt='2019-02-10' limit 1;

select \* from ods\_order\_info where dt='2019-02-11' limit 1;

## 3.4 DWD层

对ODS层数据进行判空过滤。对商品分类表进行维度退化（降维）。

### 3.4.1 创建订单表

hive (gmall)>

drop table if exists dwd\_order\_info;

create external table dwd\_order\_info (

  `id` string COMMENT '',

  `total\_amount` decimal(10,2) COMMENT '',

  `order\_status` string COMMENT ' 1 2 3 4 5',

   `user\_id` string COMMENT 'id',

    `payment\_way` string COMMENT '',

    `out\_trade\_no` string COMMENT '',

    `create\_time` string COMMENT '',

    `operate\_time` string COMMENT ''

)

PARTITIONED BY (`dt` string)

stored as parquet

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

tblproperties ("parquet.compression"="snappy")

;

### 3.4.2 创建订单详情表

hive (gmall)>

drop table if exists dwd\_order\_detail;

create external table dwd\_order\_detail(

  `id` string COMMENT '',

  `order\_id` decimal(10,2) COMMENT '',

   `user\_id` string COMMENT 'id',

    `sku\_id` string COMMENT 'id',

    `sku\_name` string COMMENT '',

    `order\_price` string COMMENT '',

    `sku\_num` string COMMENT '',

    `create\_time` string COMMENT ''

)

PARTITIONED BY (`dt` string)

stored as parquet

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

tblproperties ("parquet.compression"="snappy")

;

### 3.4.3 创建用户表

hive (gmall)>

drop table if exists dwd\_user\_info;

create external table dwd\_user\_info(

  `id` string COMMENT 'id',

  `name` string COMMENT '',

   `birthday` string COMMENT '',

    `gender` string COMMENT '',

    `email` string COMMENT '',

    `user\_level` string COMMENT '',

    `create\_time` string COMMENT ''

)

PARTITIONED BY (`dt` string)

stored as parquet

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

tblproperties ("parquet.compression"="snappy")

;

### 3.4.4 创建支付流水表

hive (gmall)>

drop table if exists dwd\_payment\_info;

create external table dwd\_payment\_info(

    `id`   bigint COMMENT '',

    `out\_trade\_no`   string COMMENT '',

    `order\_id`        string COMMENT '',

    `user\_id`         string COMMENT '',

    `alipay\_trade\_no` string COMMENT '',

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

    `subject`         string COMMENT '',

    `payment\_type` string COMMENT '',

    `payment\_time`   string COMMENT ''

   )

PARTITIONED BY (`dt` string)

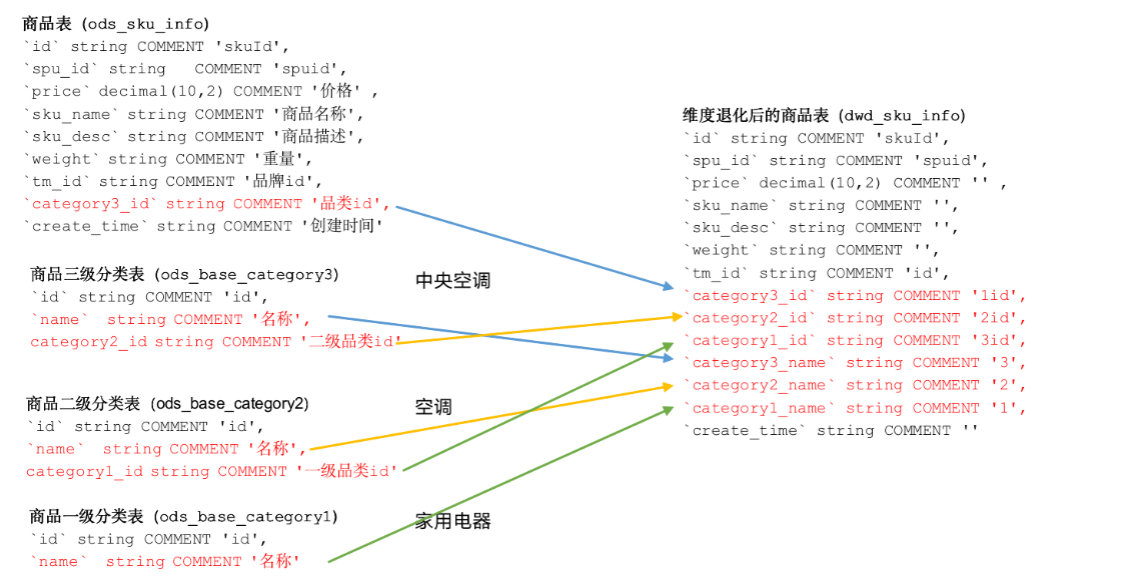
stored as parquet

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

tblproperties ("parquet.compression"="snappy")

;

### 3.4.5 创建商品表（增加分类）



hive (gmall)>

drop table if exists dwd\_sku\_info;

create external table dwd\_sku\_info(

  `id` string COMMENT 'skuId',

  `spu\_id` string COMMENT 'spuid',

   `price` decimal(10,2) COMMENT '',

    `sku\_name` string COMMENT '',

    `sku\_desc` string COMMENT '',

    `weight` string COMMENT '',

    `tm\_id` string COMMENT 'id',

    `category3\_id` string COMMENT '1id',

    `category2\_id` string COMMENT '2id',

    `category1\_id` string COMMENT '3id',

    `category3\_name` string COMMENT '3',

    `category2\_name` string COMMENT '2',

    `category1\_name` string COMMENT '1',

    `create\_time` string COMMENT ''

)

PARTITIONED BY (`dt` string)

stored as parquet

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

tblproperties ("parquet.compression"="snappy")

;

### 3.4.6 DWD层数据导入脚本

1）在/home/bawei/bin目录下创建脚本dwd\_db.sh

[bawei@hadoop102 bin]$ vim dwd\_db.sh

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 overwrite table "$APP".dwd\_order\_info partition(dt)

select \* from "$APP".ods\_order\_info

where dt='$do\_date' and id is not null;

insert overwrite table "$APP".dwd\_order\_detail partition(dt)

select \* from "$APP".ods\_order\_detail

where dt='$do\_date' and id is not null;

insert overwrite table "$APP".dwd\_user\_info partition(dt)

select \* from "$APP".ods\_user\_info

where dt='$do\_date' and id is not null;

insert overwrite table "$APP".dwd\_payment\_info partition(dt)

select \* from "$APP".ods\_payment\_info

where dt='$do\_date' and id is not null;

insert overwrite table "$APP".dwd\_sku\_info partition(dt)

select

sku.id,

sku.spu\_id,

sku.price,

sku.sku\_name,

sku.sku\_desc,

sku.weight,

sku.tm\_id,

sku.category3\_id,

c2.id category2\_id,

c1.id category1\_id,

c3.name category3\_name,

c2.name category2\_name,

c1.name category1\_name,

sku.create\_time,

sku.dt

from

"$APP".ods\_sku\_info sku

join "$APP".ods\_base\_category3 c3 on sku.category3\_id=c3.id

join "$APP".ods\_base\_category2 c2 on c3.category2\_id=c2.id

join "$APP".ods\_base\_category1 c1 on c2.category1\_id=c1.id

where sku.dt='$do\_date' and c2.dt='$do\_date'

and c3.dt='$do\_date' and c1.dt='$do\_date'

and sku.id is not null;

"

$hive -e "$sql"

2）增加脚本执行权限

[bawei@hadoop102 bin]$ chmod 777 dwd\_db.sh

3）执行脚本导入数据

[bawei@hadoop102 bin]$ dwd\_db.sh 2019-02-10

[bawei@hadoop102 bin]$ dwd\_db.sh 2019-02-11

4）查看导入数据

hive (gmall)>

select \* from dwd\_sku\_info where dt='2019-02-10' limit 2;

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

### 3.4.3 小结

思考：

1）维度退化要付出什么代价？

如果被退化的维度，还有其他业务表使用，退化后处理起来就麻烦些。

2）想想在实际业务中还有那些维度表可以退化

城市的三级分类（省、市、县）等

## 3.5 DWS层之用户行为宽表

1）为什么要建宽表

需求目标，把每个用户单日的行为聚合起来组成一张多列宽表，以便之后关联用户维度信息后进行，不同角度的统计分析。

### 3.5.1 创建用户行为宽表

hive (gmall)>

drop table if exists dws\_user\_action;

create external table dws\_user\_action

(

user\_id string comment '用户 id',

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\_count bigint comment '评论次数'

) COMMENT '每日用户行为宽表'

PARTITIONED BY (`dt` string)

stored as parquet

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

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

### 3.5.2 向用户行为宽表导入数据

1）导入数据

hive (gmall)>

with

tmp\_order as

(

select

user\_id,

count(\*) order\_count,

sum(oi.total\_amount) order\_amount

from dwd\_order\_info oi

where date\_format(oi.create\_time,'yyyy-MM-dd')='2019-02-10'

group by user\_id

) ,

tmp\_payment as

(

select

user\_id,

sum(pi.total\_amount) payment\_amount,

count(\*) payment\_count

from dwd\_payment\_info pi

where date\_format(pi.payment\_time,'yyyy-MM-dd')='2019-02-10'

group by user\_id

),

tmp\_comment as

(

select

user\_id,

count(\*) comment\_count

from dwd\_comment\_log c

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

group by user\_id

)

insert overwrite table dws\_user\_action partition(dt='2019-02-10')

select

user\_actions.user\_id,

sum(user\_actions.order\_count),

sum(user\_actions.order\_amount),

sum(user\_actions.payment\_count),

sum(user\_actions.payment\_amount),

sum(user\_actions.comment\_count)

from

(

select

user\_id,

order\_count,

order\_amount,

0 payment\_count,

0 payment\_amount,

0 comment\_count

from tmp\_order

union all

select

user\_id,

0,

0,

payment\_count,

payment\_amount,

0

from tmp\_payment

union all

select

user\_id,

0,

0,

0,

0,

comment\_count

from tmp\_comment

) user\_actions

group by user\_id;

2）查询导入结果

hive (gmall)> select \* from dws\_user\_action;

### 3.5.3 用户行为数据宽表导入脚本

1）在/home/bawei/bin目录下创建脚本dws\_db\_wide.sh

[bawei@hadoop102 bin]$ vim dws\_db\_wide.sh

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="

with

tmp\_order as

(

select

user\_id,

sum(oi.total\_amount) order\_amount,

count(\*) order\_count

from "$APP".dwd\_order\_info oi

where date\_format(oi.create\_time,'yyyy-MM-dd')='$do\_date'

group by user\_id

) ,

tmp\_payment as

(

select

user\_id,

sum(pi.total\_amount) payment\_amount,

count(\*) payment\_count

from "$APP".dwd\_payment\_info pi

where date\_format(pi.payment\_time,'yyyy-MM-dd')='$do\_date'

group by user\_id

),

tmp\_comment as

(

select

user\_id,

count(\*) comment\_count

from "$APP".dwd\_comment\_log c

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

group by user\_id

)

Insert overwrite table "$APP".dws\_user\_action partition(dt='$do\_date')

select

user\_actions.user\_id,

sum(user\_actions.order\_count),

sum(user\_actions.order\_amount),

sum(user\_actions.payment\_count),

sum(user\_actions.payment\_amount),

sum(user\_actions.comment\_count)

from

(

select

user\_id,

order\_count,

order\_amount,

0 payment\_count,

0 payment\_amount,

0 comment\_count

from tmp\_order

union all

select

user\_id,

0,

0,

payment\_count,

payment\_amount,

0

from tmp\_payment

union all

select

user\_id,

0,

0,

0,

0,

comment\_count

from tmp\_comment

) user\_actions

group by user\_id;

"

$hive -e "$sql"

2）增加脚本执行权限

[bawei@hadoop102 bin]$ chmod 777 dws\_db\_wide.sh

3）执行脚本导入数据

[bawei@hadoop102 bin]$ dws\_db\_wide.sh 2019-02-11

4）查看导入数据

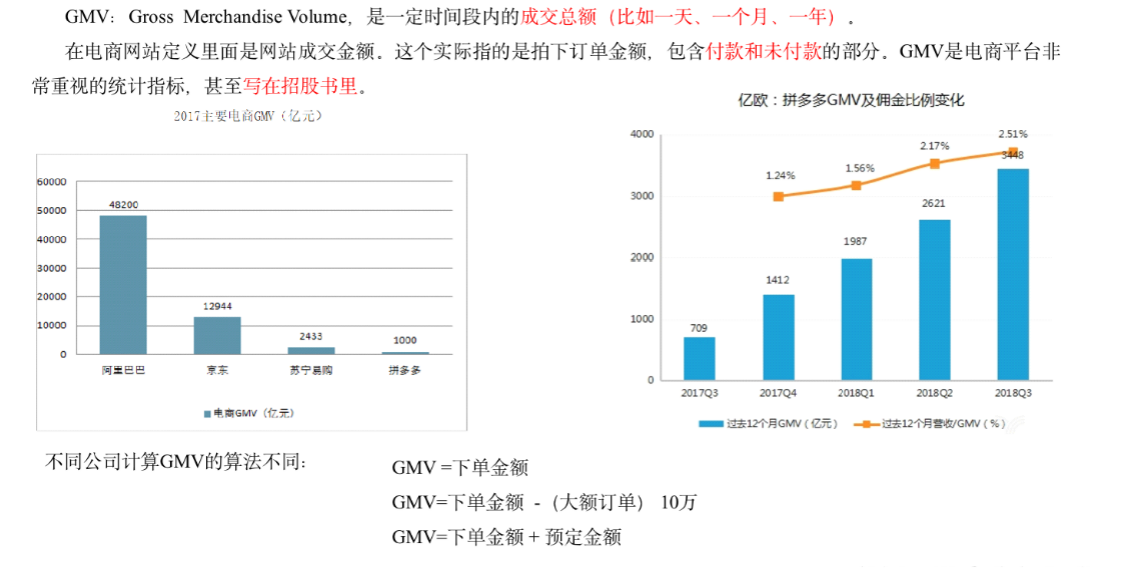
hive (gmall)>

select \* from dws\_user\_action where dt='2019-02-11' limit 2;

# 第4章 需求一：GMV成交总额

## 4.1 ADS层

### 4.1.1 什么是GMV



### 4.1.2 建表语句

hive (gmall)>

drop table if exists ads\_gmv\_sum\_day;

create external table ads\_gmv\_sum\_day(

    `dt` string COMMENT '统计日期',

    `gmv\_count` bigint COMMENT '当日gmv订单个数',

    `gmv\_amount` decimal(16,2) COMMENT '当日gmv订单总金额',

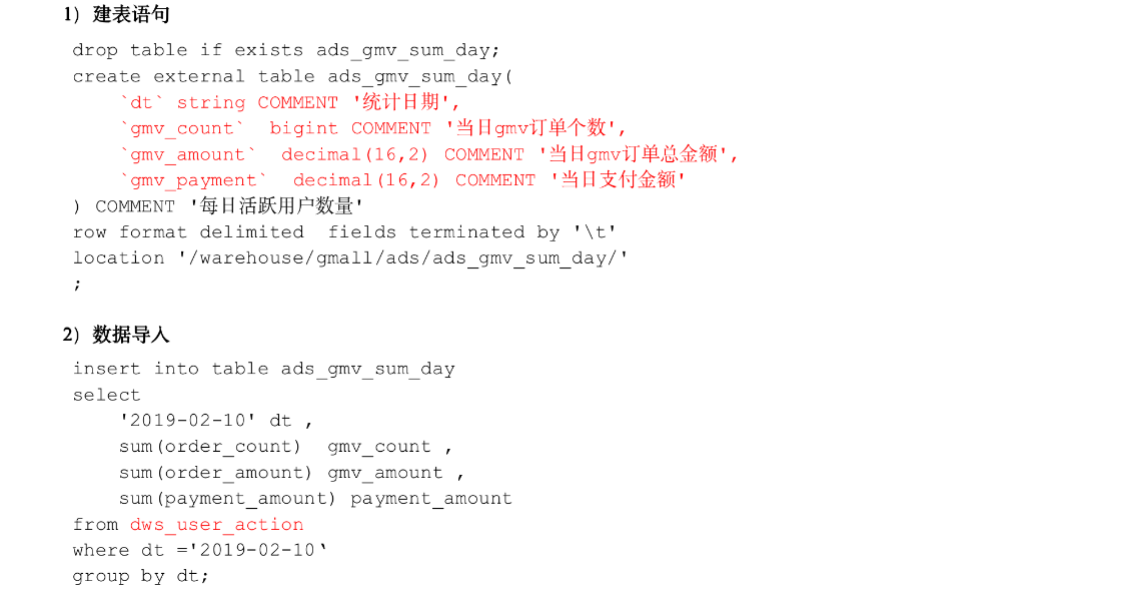
    `gmv\_payment` decimal(16,2) COMMENT '当日支付金额'

) COMMENT 'GMV'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_gmv\_sum\_day/'

;



### 4.1.3 数据导入

1）数据导入

hive (gmall)>

insert into table ads\_gmv\_sum\_day

select

'2019-02-10' dt,

sum(order\_count) gmv\_count,

sum(order\_amount) gmv\_amount,

sum(payment\_amount) payment\_amount

from dws\_user\_action

where dt ='2019-02-10'

group by dt

;

2）查询导入数据

hive (gmall)> select \* from ads\_gmv\_sum\_day;

### 4.1.4 数据导入脚本

1）在/home/bawei/bin目录下创建脚本ads\_db\_gmv.sh

[bawei@hadoop102 bin]$ vim ads\_db\_gmv.sh

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="

insert into table "$APP".ads\_gmv\_sum\_day

select

'$do\_date' dt,

sum(order\_count) gmv\_count,

sum(order\_amount) gmv\_amount,

sum(payment\_amount) payment\_amount

from "$APP".dws\_user\_action

where dt ='$do\_date'

group by dt;

"

$hive -e "$sql"

2）增加脚本执行权限

[bawei@hadoop102 bin]$ chmod 777 ads\_db\_gmv.sh

3）执行脚本导入数据

[bawei@hadoop102 bin]$ ads\_db\_gmv.sh 2019-02-11

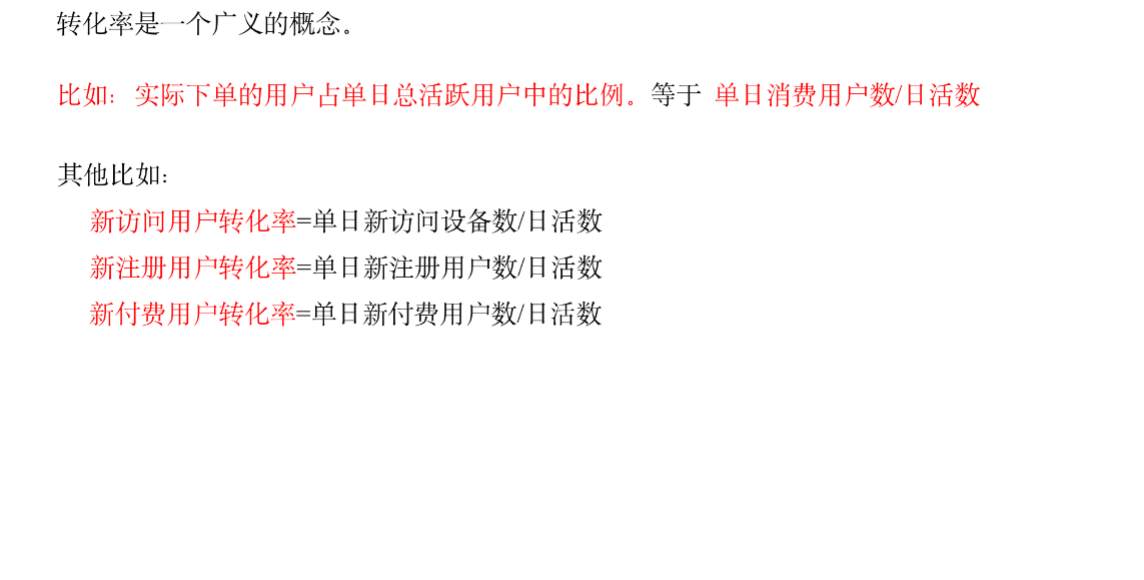
4）查看导入数据

hive (gmall)>

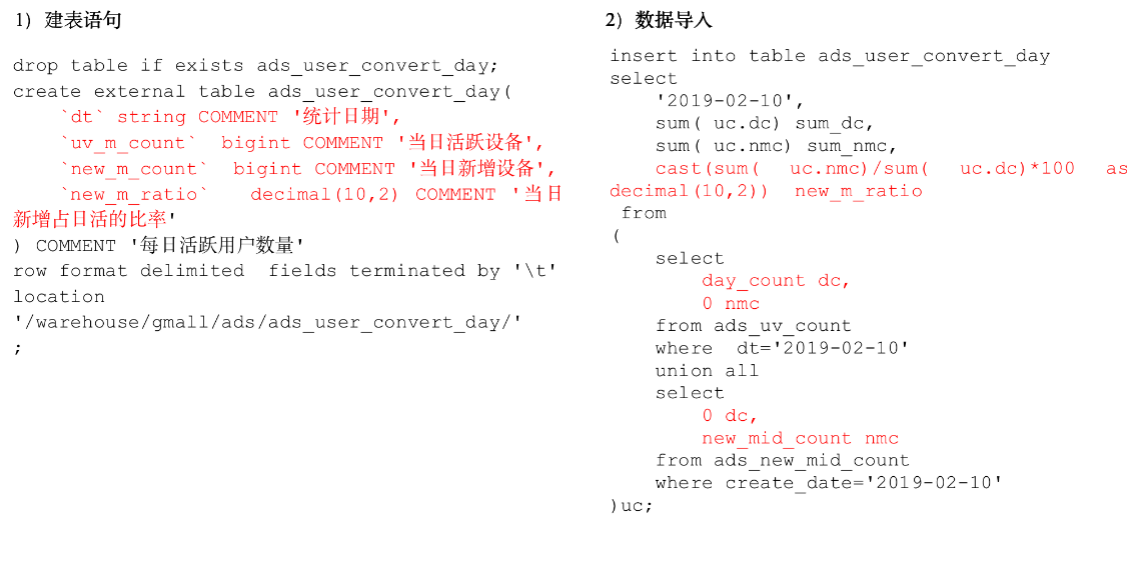
select \* from ads\_gmv\_sum\_day where dt='2019-02-11' limit 2;

# 第5章 需求二：转化率之用户新鲜度及漏斗分析

## 5.1 什么是转化率



## 5.2 ADS层之新增用户占日活跃用户比率（用户新鲜度）



### 5.2.1 建表语句

hive (gmall)>

drop table if exists ads\_user\_convert\_day;

create external table ads\_user\_convert\_day(

`dt` string COMMENT '统计日期',

`uv\_m\_count` bigint COMMENT '当日活跃设备',

`new\_m\_count` bigint COMMENT '当日新增设备',

`new\_m\_ratio` decimal(10,2) COMMENT '当日新增占日活的比率'

) COMMENT '转化率'

row format delimited fields terminated by '\t'

location '/warehouse/gmall/ads/ads\_user\_convert\_day/'

;

### 5.2.2 数据导入

1）数据导入

hive (gmall)>

insert into table ads\_user\_convert\_day

select

'2019-02-10',

sum(uc.dc) sum\_dc,

sum(uc.nmc) sum\_nmc,

cast(sum( uc.nmc)/sum( uc.dc)\*100 as decimal(10,2)) new\_m\_ratio

from

(

select

day\_count dc,

0 nmc

from ads\_uv\_count

where dt='2019-02-10'

union all

select

0 dc,

new\_mid\_count nmc

from ads\_new\_mid\_count

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

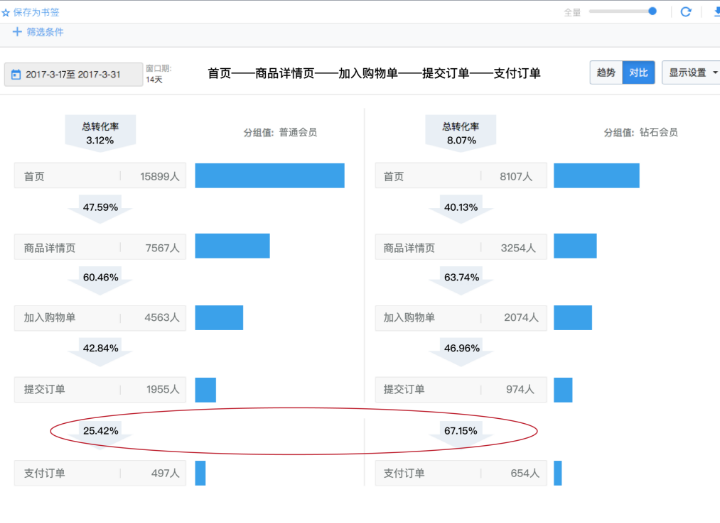
)uc;

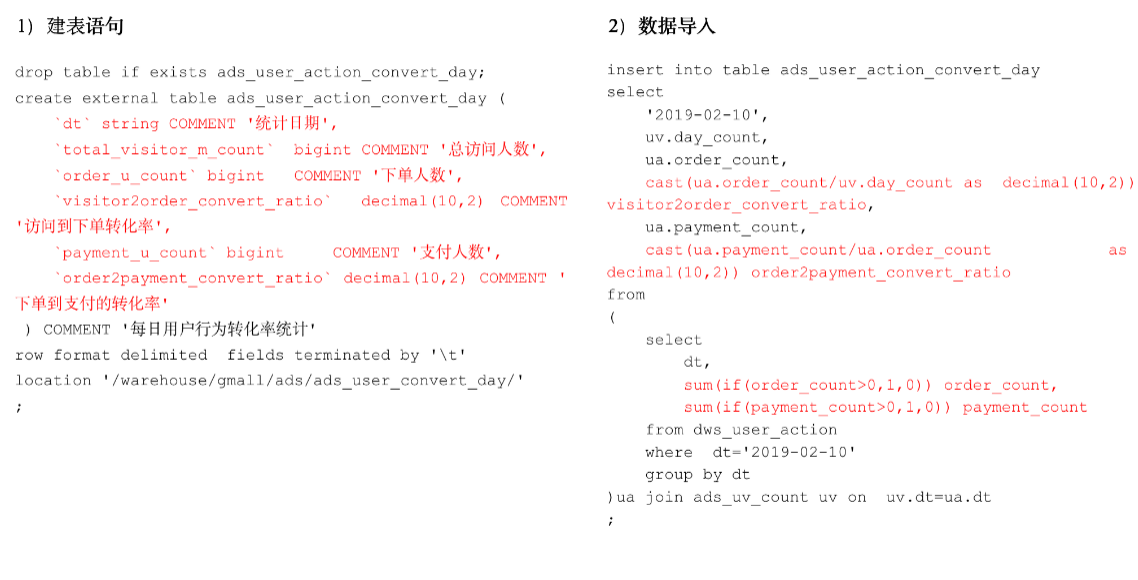
2）查看导入数据

hive (gmall)>

select \* from ads\_user\_convert\_day;

## 5.3 ADS层之用户行为漏斗分析





### 5.3.1 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',

`order\_u\_count` bigint COMMENT '下单人数',

`visitor2order\_convert\_ratio` decimal(10,2) COMMENT '访问到下单转化率',

`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/'

;

### 5.3.2 数据导入

1）数据导入

hive (gmall)>

insert into table ads\_user\_action\_convert\_day

select

'2019-02-10',

uv.day\_count,

ua.order\_count,

cast(ua.order\_count/uv.day\_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(order\_count>0,1,0)) order\_count,

sum(if(payment\_count>0,1,0)) payment\_count

from dws\_user\_action

where dt='2019-02-10'

group by dt

)ua join ads\_uv\_count uv on uv.dt=ua.dt

;

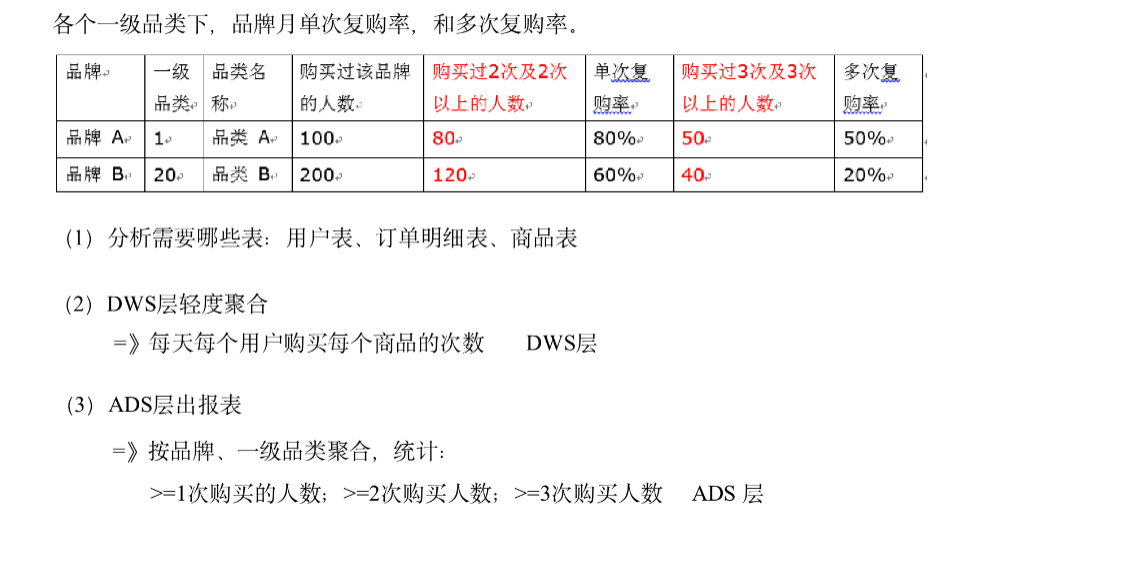
2）查询导入数据

hive (gmall)> select \* from ads\_user\_action\_convert\_day;

# 第6章 需求三：品牌复购率

需求：以月为单位统计，购买2次以上商品的用户

## 6.1 复购率计算分析



## 6.2 DWS层

### 6.2.1 用户购买商品明细表（宽表）

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 string comment '当日下单单数',

order\_amount string comment '当日下单金额'

) COMMENT '用户购买商品明细表'

PARTITIONED BY (`dt` string)

stored as parquet

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

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

### 6.2.2 数据导入

hive (gmall)>

with

tmp\_detail as

(

select

user\_id,

sku\_id,

sum(sku\_num) sku\_num,

count(\*) order\_count,

sum(od.order\_price\*sku\_num) order\_amount

from dwd\_order\_detail od

where od.dt='2019-02-10'

group by user\_id, sku\_id

)

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

select

tmp\_detail.user\_id,

tmp\_detail.sku\_id,

u.gender,

months\_between('2019-02-10', u.birthday)/12 age,

u.user\_level,

price,

sku\_name,

tm\_id,

category3\_id,

category2\_id,

category1\_id,

category3\_name,

category2\_name,

category1\_name,

spu\_id,

tmp\_detail.sku\_num,

tmp\_detail.order\_count,

tmp\_detail.order\_amount

from tmp\_detail

left join dwd\_user\_info u on tmp\_detail.user\_id =u.id and u.dt='2019-02-10'

left join dwd\_sku\_info s on tmp\_detail.sku\_id =s.id and s.dt='2019-02-10'

;

### 6.2.3 数据导入脚本

1）在/home/bawei/bin目录下创建脚本dws\_sale.sh

[bawei@hadoop102 bin]$ vim dws\_sale.sh

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;

with

tmp\_detail as

(

select

user\_id,

sku\_id,

sum(sku\_num) sku\_num,

count(\*) order\_count,

sum(od.order\_price\*sku\_num) order\_amount

from "$APP".dwd\_order\_detail od

where od.dt='$do\_date'

group by user\_id, sku\_id

)

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

select

tmp\_detail.user\_id,

tmp\_detail.sku\_id,

u.gender,

months\_between('$do\_date', u.birthday)/12 age,

u.user\_level,

price,

sku\_name,

tm\_id,

category3\_id,

category2\_id,

category1\_id,

category3\_name,

category2\_name,

category1\_name,

spu\_id,

tmp\_detail.sku\_num,

tmp\_detail.order\_count,

tmp\_detail.order\_amount

from tmp\_detail

left join "$APP".dwd\_user\_info u

on tmp\_detail.user\_id=u.id and u.dt='$do\_date'

left join "$APP".dwd\_sku\_info s on tmp\_detail.sku\_id =s.id and s.dt='$do\_date';

"

$hive -e "$sql"

2）增加脚本执行权限

[bawei@hadoop102 bin]$ chmod 777 dws\_sale.sh

3）执行脚本导入数据

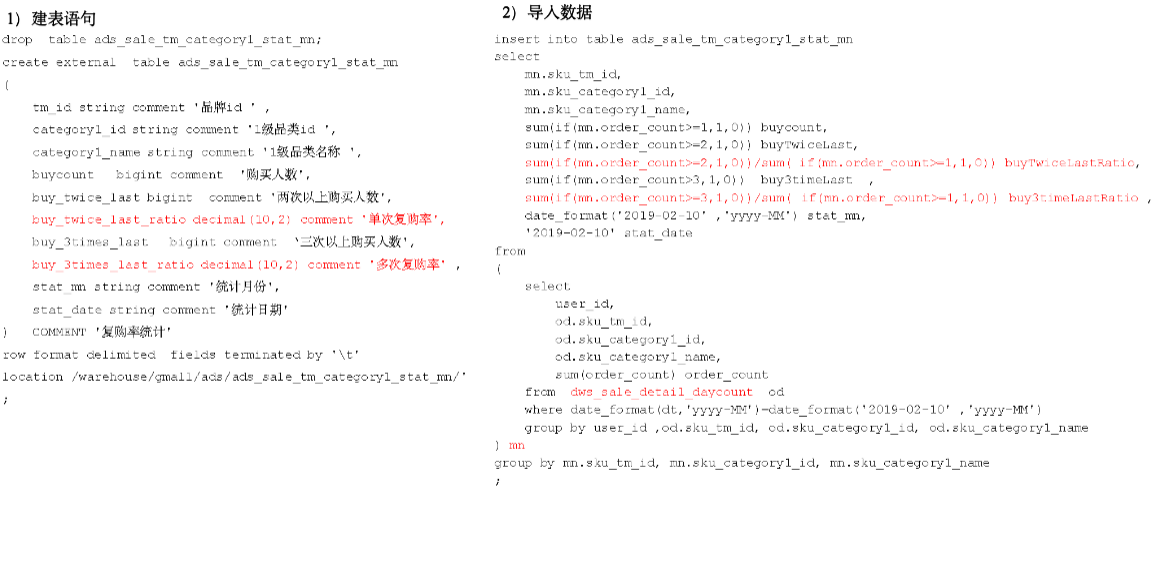
[bawei@hadoop102 bin]$ dws\_sale.sh 2019-02-11

4）查看导入数据

hive (gmall)>

select \* from dws\_sale\_detail\_daycount where dt='2019-02-11' limit 2;

## 6.3 ADS层品牌复购率



### 6.3.1 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/'

;

### 6.3.2 数据导入

1）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

;

2）查询导入数据

hive (gmall)> select \* from ads\_sale\_tm\_category1\_stat\_mn;

### 6.3.3 数据导入脚本

1）在/home/bawei/bin目录下创建脚本ads\_sale.sh

[bawei@hadoop102 bin]$ vim ads\_sale.sh

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"

2）增加脚本执行权限

[bawei@hadoop102 bin]$ chmod 777 ads\_sale.sh

3）执行脚本导入数据

[bawei@hadoop102 bin]$ ads\_sale.sh 2019-02-11

4）查看导入数据

hive (gmall)>

select \* from ads\_sale\_tm\_category1\_stat\_mn limit 2;

## 6.4 练习：求每个等级的用户对应的复购率前十的商品排行

1）每个等级，每种商品，买一次的用户数，买两次的用户数=》得出复购率

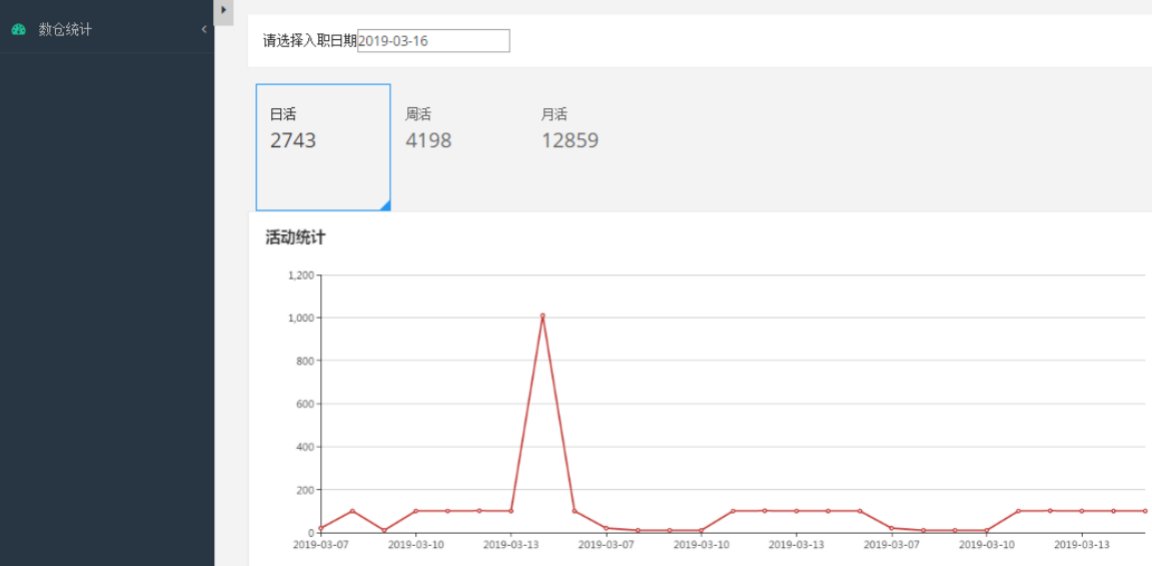
2）利用开窗函数，取每个等级的前十

3）形成脚本

# 第7章 数据可视化

## 7.1 在MySQL中创建表

### 7.1.1 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) DEFAULT NULL 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 ('2019-03-01 14:10:04', 20, 30, 100, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-02 14:12:48', 35, 50, 100, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-03 14:14:07', 25, 640, 3300, 'Y', 'Y');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-04 14:14:14', 10, 23, 123, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-05 14:14:21', 80, 121, 131, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-06 14:14:38', 30, 53, 453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-07 14:33:27', 20, 31, 453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-08 14:33:39', 10, 53, 453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-09 14:33:47', 10, 34, 453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-10 14:33:54', 10, 653, 8453, 'Y', 'Y');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-11 14:34:04', 100, 453, 1453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-12 14:34:10', 101, 153, 134, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-13 14:34:16', 100, 286, 313, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-14 14:34:22', 100, 45, 453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-15 14:34:29', 100, 345, 3453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-16 14:34:35', 101, 453, 453, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-17 14:34:41', 100, 678, 9812, 'Y', 'Y');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-18 14:34:46', 100, 186, 193, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-19 14:34:53', 453, 686, 712, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-20 14:34:57', 452, 786, 823, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-21 14:35:02', 214, 58, 213, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-22 14:35:08', 76, 78, 95, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-23 14:35:13', 76, 658, 745, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-24 14:35:19', 76, 687, 9300, 'Y', 'Y');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-25 14:35:25', 76, 876, 923, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-26 14:35:30', 76, 456, 511, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-27 14:35:35', 76, 456, 623, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-28 14:35:41', 43, 753, 4000, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-29 14:35:47', 76, 876, 4545, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-30 14:35:57', 76, 354, 523, 'N', 'N');

INSERT INTO `ads\_uv\_count` VALUES ('2019-03-31 14:36:02', 43, 634, 6213, 'Y', 'Y');

### 7.1.2 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) DEFAULT NULL 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 ('2019-03-09','2019-03-08', 1,88, 99, 0.78);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-10','2019-03-08', 2,77, 88, 0.68);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-11','2019-03-08', 3,66, 77, 0.58);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-12','2019-03-08', 4,55, 66, 0.48);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-13','2019-03-08', 5,44, 55, 0.38);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-14','2019-03-08', 6,33, 44, 0.28);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-10','2019-03-09', 1,77, 88, 0.56);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-11','2019-03-09', 2,66, 77, 0.46);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-12','2019-03-09', 3,55, 66, 0.36);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-13','2019-03-09', 4,44, 55, 0.26);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-14','2019-03-09', 5,33, 44, 0.16);

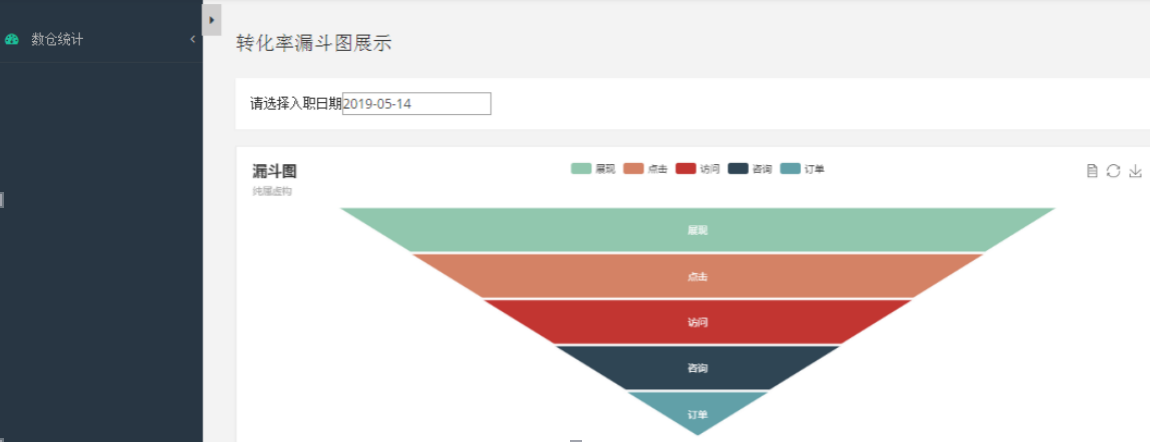
INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-11','2019-03-10', 1,66, 77, 0.55);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-12','2019-03-10', 2,55, 66, 0.45);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-13','2019-03-10', 3,44, 55, 0.35);

INSERT INTO `ads\_user\_retention\_day\_rate` VALUES ('2019-03-14','2019-03-10', 4,33, 44, 0.25);

### 7.1.3 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) DEFAULT NULL 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 ('2019-04-28 19:36:18', 10000, 3000, 0.25, 2000, 0.15);

### 7.1.4 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) DEFAULT NULL 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

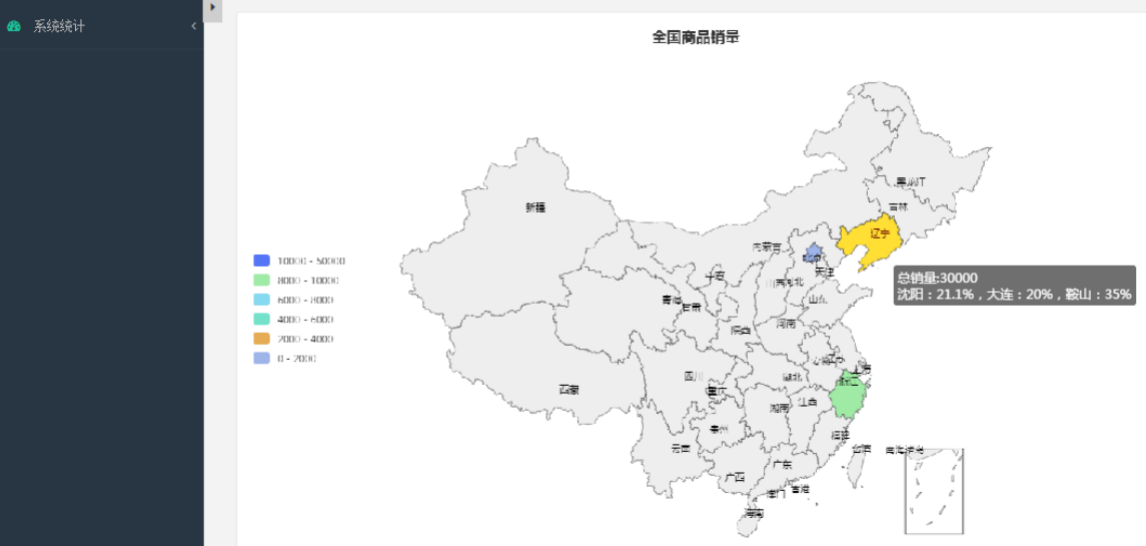
INSERT INTO `ads\_gmv\_sum\_day` VALUES ('2019-03-01 22:51:37', 1000, 210000.00, 2000.00);

INSERT INTO `ads\_gmv\_sum\_day` VALUES ('2019-05-08 22:52:32', 3434, 12413.00, 1.00);

INSERT INTO `ads\_gmv\_sum\_day` VALUES ('2019-07-13 22:52:51', 1222, 324345.00, 1.00);

INSERT INTO `ads\_gmv\_sum\_day` VALUES ('2019-09-13 22:53:08', 2344, 12312.00, 1.00);

### 7.1.5 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%');

## 7.2 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

## 7.3 Sqoop导出脚本

1）编写Sqoop导出脚本

在/home/bawei/bin目录下创建脚本sqoop\_export.sh

[bawei@hadoop102 bin]$ vim sqoop\_export.sh

在脚本中填写如下内容

#!/bin/bash

db\_name=gmall

export\_data() {

/opt/module/sqoop/bin/sqoop export \

--connect "jdbc:mysql://hadoop102:3306/${db\_name}?useUnicode=true&characterEncoding=utf-8" \

--username root \

--password 000000 \

--table $1 \

--num-mappers 1 \

--export-dir /warehouse/$db\_name/ads/$1 \

--input-fields-terminated-by "\t" \

--update-mode allowinsert \

--update-key "tm\_id,category1\_id,stat\_mn,stat\_date" \

--input-null-string '\\N' \

--input-null-non-string '\\N'

}

case $1 in

"ads\_uv\_count")

export\_data "ads\_uv\_count"

;;

"ads\_user\_action\_convert\_day")

export\_data "ads\_user\_action\_convert\_day"

;;

"ads\_gmv\_sum\_day")

export\_data "ads\_gmv\_sum\_day"

;;

"all")

export\_data "ads\_uv\_count"

export\_data "ads\_user\_action\_convert\_day"

export\_data "ads\_gmv\_sum\_day"

;;

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

[bawei@hadoop102 bin]$ chmod 777 sqoop\_export.sh

[bawei@hadoop102 bin]$ sqoop\_export.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8章 Azkaban调度器

## 8.1 Azkaban安装

详见：大数据技术之Azkaban



## 8.2 GMV指标获取的全调度流程

1）生成数据

CALL init\_data('2019-02-12',300,200,300,FALSE);

2）编写Azkaban程序运行job

（1）import.job文件

type=command

do\_date=${dt}

command=/home/bawei/bin/sqoop\_import.sh all ${do\_date}

（2）ods.job文件

type=command

do\_date=${dt}

dependencies=import

command=/home/bawei/bin/ods\_db.sh ${do\_date}

（3）dwd.job文件

type=command

do\_date=${dt}

dependencies=ods

command=/home/bawei/bin/dwd\_db.sh ${do\_date}

（4）dws.job文件

type=command

do\_date=${dt}

dependencies=dwd

command=/home/bawei/bin/dws\_db\_wide.sh ${do\_date}

（5）ads.job文件

type=command

do\_date=${dt}

dependencies=dws

command=/home/bawei/bin/ads\_db\_gmv.sh ${do\_date}

（6）export.job文件

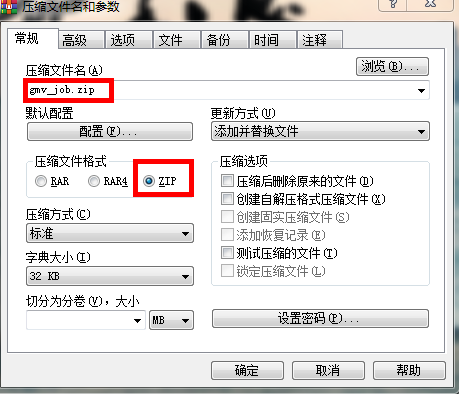
type=command

dependencies=ads

command=/home/bawei/bin/sqoop\_export.sh ads\_gmv\_sum\_day

（7）将以上6个文件压缩成gmv-job.zip文件

3）创建Azkaban工程，并上传gmv-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\_gmv\_sum\_day;

# 第9章 订单表拉链表

## 9.1 什么是拉链表

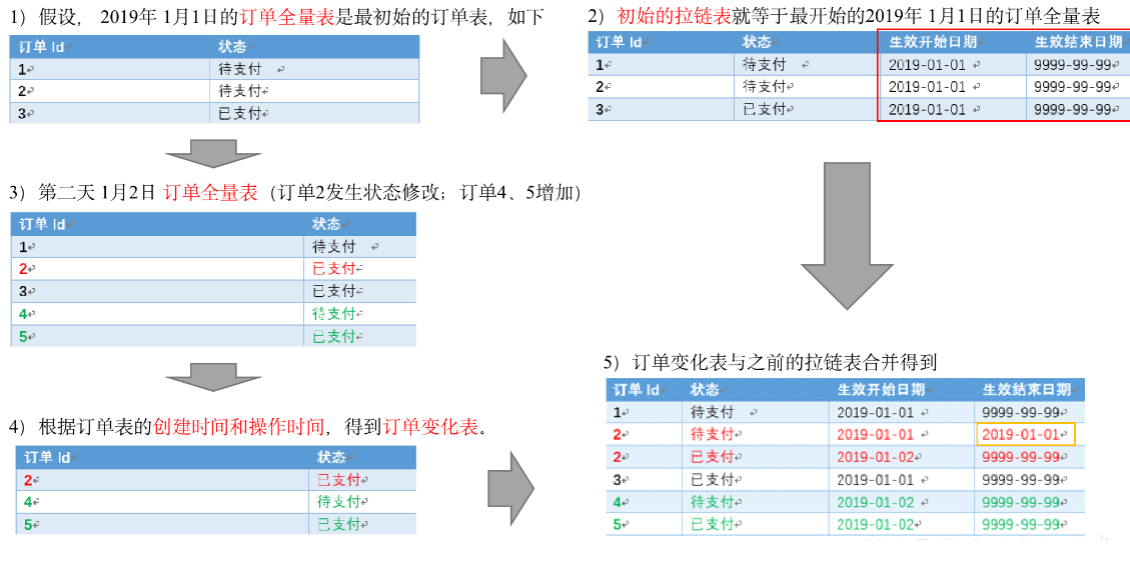


## 9.2 为什么要做拉链表

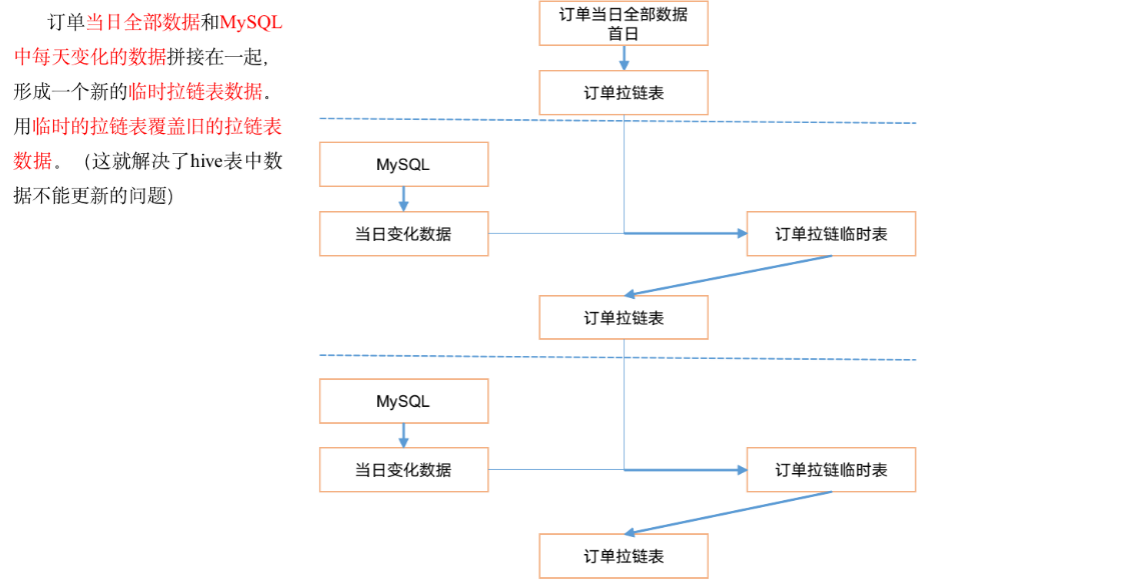




## 9.3 拉链表形成过程

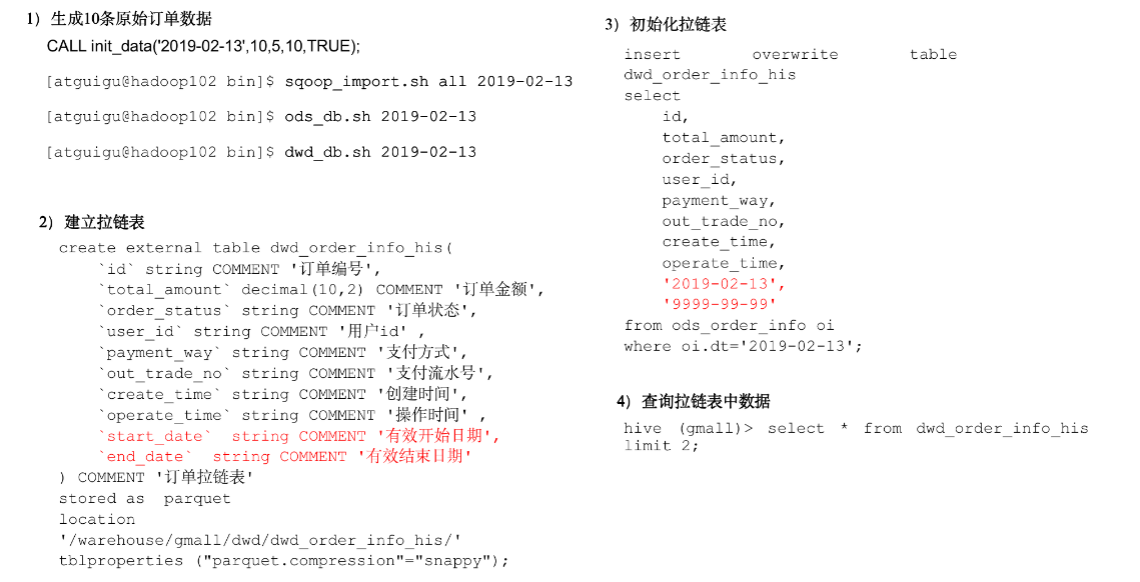


## 9.4 拉链表制作过程图



## 9.5 拉链表制作过程

### 9.5.1 步骤0：初始化拉链表（首次独立执行）



1）生成10条原始订单数据

CALL init\_data('2019-02-13',10,5,10,TRUE);

[bawei@hadoop102 bin]$ sqoop\_import.sh all 2019-02-13

[bawei@hadoop102 bin]$ ods\_db.sh 2019-02-13

[bawei@hadoop102 bin]$ dwd\_db.sh 2019-02-13

2）建立拉链表

hive (gmall)>

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

create external table dwd\_order\_info\_his(

  `id` string COMMENT '订单编号',

  `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' ,

    `payment\_way` string COMMENT '支付方式',

    `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',

    `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\_order\_info\_his/'

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

3）初始化拉链表

hive (gmall)>

insert overwrite table dwd\_order\_info\_his

select

id,

total\_amount,

order\_status,

user\_id,

payment\_way,

out\_trade\_no,

create\_time,

operate\_time,

'2019-02-13',

'9999-99-99'

from ods\_order\_info oi

where oi.dt='2019-02-13';

4）查询拉链表中数据

hive (gmall)> select \* from dwd\_order\_info\_his limit 2;

### 9.5.2 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

（1）最好表内有创建时间和变动时间（Lucky!）

（2）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)。

（3）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)

（4）要求业务数据库提供变动流水（人品，颜值）

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

（1）2019-02-14日新增2条订单数据

CALL init\_data('2019-02-14',2,5,10,TRUE);

（2）通过Sqoop把2019-02-14日所有数据导入

sqoop\_import.sh all 2019-02-14

（3）ODS层数据导入

ods\_db.sh 2019-02-14

（4）DWD层数据导入

dwd\_db.sh 2019-02-14

### 9.5.3 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\_order\_info\_his\_tmp;

create table dwd\_order\_info\_his\_tmp(

  `id` string COMMENT '订单编号',

  `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' ,

    `payment\_way` string COMMENT '支付方式',

    `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',

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

    `end\_date` string COMMENT '有效结束日期'

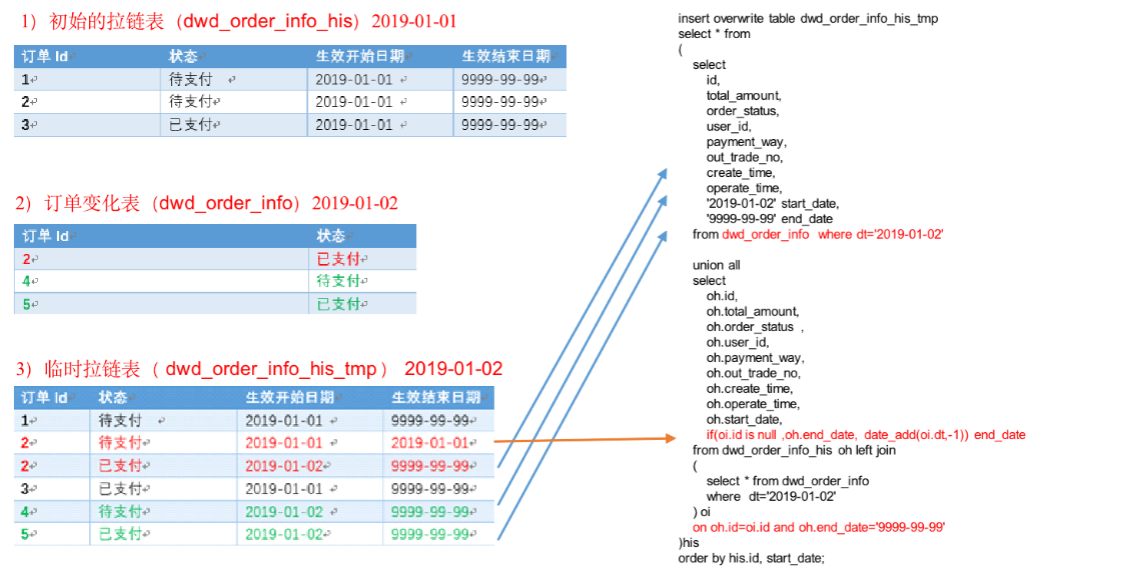
) COMMENT '订单拉链临时表'

stored as parquet

location '/warehouse/gmall/dwd/dwd\_order\_info\_his\_tmp/'

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

2）导入脚本



hive (gmall)>

insert overwrite table dwd\_order\_info\_his\_tmp

select \* from

(

select

id,

   total\_amount,

   order\_status,

    user\_id,

    payment\_way,

    out\_trade\_no,

    create\_time,

    operate\_time,

    '2019-02-14' start\_date,

   '9999-99-99' end\_date

from dwd\_order\_info where dt='2019-02-14'

union all

select oh.id,

    oh.total\_amount,

    oh.order\_status,

    oh.user\_id,

    oh.payment\_way,

    oh.out\_trade\_no,

    oh.create\_time,

    oh.operate\_time,

    oh.start\_date,

    if(oi.id is null, oh.end\_date, date\_add(oi.dt,-1)) end\_date

from dwd\_order\_info\_his oh left join

     (

select

\*

from dwd\_order\_info

where dt='2019-02-14'

) oi

     on oh.id=oi.id and oh.end\_date='9999-99-99'

)his

order by his.id, start\_date;

### 9.5.4 步骤3：把临时表覆盖给拉链表

1）导入数据

hive (gmall)>

insert overwrite table dwd\_order\_info\_his

select \* from dwd\_order\_info\_his\_tmp;

2）查询导入数据

hive (gmall)> select \* from dwd\_order\_info\_his;

### 9.5.5 整理为每日脚本

# 第10章 项目总结

## 10.1 熟悉8张表的业务字段，每张表记住3-5个字段

## 10.2 数仓理论

1）表的分类：实体表、维度表、事务型事实表、周期型事实表

2）表的同步策略：

实体表（全量）

维度表（全量）

事务型事实表（增量）

周期型事实表（新增和变化、拉链表）

3）范式理论：

一范式原则：属性不可切割；

二范式原则：不能存在部分函数依赖；

三范式原则：不能存在传递函数依赖；

4）数仓维度建模模型

星型模型，维度一层；

雪花模型，维度多层；

星座模型，多个事实表；

性能优先选择星型模型，灵活优先选择雪花模型。企业中星型模型多一些。

## 10.3 Sqoop参数

/opt/module/sqoop/bin/sqoop import \

--connect \

--username \

--password \

--target-dir \

--delete-target-dir \

--num-mappers \

--fields-terminated-by   \

--query   "$2" ' and $CONDITIONS;'

### 10.3.1 Sqoop导入导出Null存储一致性问题

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。

### 10.3.2 Sqoop数据导出一致性问题

1）场景1：如Sqoop在导出到Mysql时，使用4个Map任务，过程中有2个任务失败，那此时MySQL中存储了另外两个Map任务导入的数据，此时老板正好看到了这个报表数据。而开发工程师发现任务失败后，会调试问题并最终将全部数据正确的导入MySQL，那后面老板再次看报表数据，发现本次看到的数据与之前的不一致，这在生产环境是不允许的。

官网：http://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html

Since Sqoop breaks down export process into multiple transactions, it is possible that a failed export job may result in partial data being committed to the database. This can further lead to subsequent jobs failing due to insert collisions in some cases, or lead to duplicated data in others. You can overcome this problem by specifying a staging table via the --staging-table option which acts as an auxiliary table that is used to stage exported data. The staged data is finally moved to the destination table in a single transaction.

–staging-table方式

sqoop export --connect jdbc:mysql://192.168.137.10:3306/user\_behavior --username root --password 123456 --table app\_cource\_study\_report --columns watch\_video\_cnt,complete\_video\_cnt,dt --fields-terminated-by "\t" --export-dir "/user/hive/warehouse/tmp.db/app\_cource\_study\_analysis\_${day}" --staging-table app\_cource\_study\_report\_tmp --clear-staging-table --input-null-string '\N'

2）场景2：设置map数量为1个（不推荐，面试官想要的答案不只这个）

多个Map任务时，采用–staging-table方式，仍然可以解决数据一致性问题。

### 10.3.3 Sqoop底层运行的任务是什么

只有Map阶段，没有Reduce阶段的任务。

### 10.3.4 Sqoop数据导出的时候一次执行多长时间

Sqoop任务5分钟-2个小时的都有。取决于数据量。

## 10.4 需求指标分析

1）GMV：一段时间内的网站成交金额（包括付款和未付款）

计算：基于用户行为宽表，对订单总金额进行sum。

2）转化率：（先分别计算分子和分母，再相除）

（1）新增用户占活跃用户的比率；

cast(sum( uc.nmc)/sum( uc.dc)\*100 as decimal(10,2)) new\_m\_ratio

（2）下单人数占活跃用户的比率；

sum(if(order\_count>0,1,0)) order\_count

cast(ua.order\_count/uv.day\_count\*100 as decimal(10,2)) visitor2order\_convert\_ratio

（3）支付人数占下单人数的比率；

sum(if(payment\_count>0,1,0)) payment\_count

cast(ua.payment\_count/ua.order\_count\*100 as decimal(10,2)) order2payment\_convert\_ratio

3）复购率：（先分别计算分子和分母，再相除）

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 ,

## 10.5 拉链表

1）通过关系型数据库的create time和operation time获取数据的新增和变化。

2）用临时拉链表解决Hive了中数据不能更新的问题。

## 10.6 Azkaban

1）每天集群运行多少job?

2）多个指标（200）\*6=1200（1000-2000个job）

3）每天集群运行多少个task? 1000\*（5-8）=5000多个

4）任务挂了怎么办？运行成功或者失败都会发邮件

## 10.7 项目中表关系

