# 第10章 数仓开发之DWS层

设计要点：

（1）DWS层的设计参考指标体系。

（2）DWS层的数据存储格式为ORC列式存储 + snappy压缩。

（3）DWS层表名的命名规范为dws\_数据域\_统计粒度\_业务过程\_统计周期（1d/nd/td）

注：1d表示最近1日，nd表示最近n日，td表示历史至今。

## 10.1 最近1日汇总表

### 10.1.1 交易域用户商品粒度订单最近1日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_sku\_order\_1d;

CREATE EXTERNAL TABLE dws\_trade\_user\_sku\_order\_1d

(

`user\_id` STRING COMMENT '用户id',

`sku\_id` STRING COMMENT 'sku\_id',

`sku\_name` STRING COMMENT 'sku名称',

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

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

`category2\_id` STRING COMMENT '一级分类id',

`category2\_name` STRING COMMENT '一级分类名称',

`category3\_id` STRING COMMENT '一级分类id',

`category3\_name` STRING COMMENT '一级分类名称',

`tm\_id` STRING COMMENT '品牌id',

`tm\_name` STRING COMMENT '品牌名称',

`order\_count\_1d` BIGINT COMMENT '最近1日下单次数',

`order\_num\_1d` BIGINT COMMENT '最近1日下单件数',

`order\_original\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日下单原始金额',

`activity\_reduce\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日活动优惠金额',

`coupon\_reduce\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日优惠券优惠金额',

`order\_total\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日下单最终金额'

) COMMENT '交易域用户商品粒度订单最近1日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_sku\_order\_1d'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

**（1）首日装载**

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

insert overwrite table dws\_trade\_user\_sku\_order\_1d partition(dt)

select

user\_id,

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name,

order\_count\_1d,

order\_num\_1d,

order\_original\_amount\_1d,

activity\_reduce\_amount\_1d,

coupon\_reduce\_amount\_1d,

order\_total\_amount\_1d,

dt

from

(

select

dt,

user\_id,

sku\_id,

count(\*) order\_count\_1d,

sum(sku\_num) order\_num\_1d,

sum(split\_original\_amount) order\_original\_amount\_1d,

sum(nvl(split\_activity\_amount,0.0)) activity\_reduce\_amount\_1d,

sum(nvl(split\_coupon\_amount,0.0)) coupon\_reduce\_amount\_1d,

sum(split\_total\_amount) order\_total\_amount\_1d

from dwd\_trade\_order\_detail\_inc

group by dt,user\_id,sku\_id

)od

left join

(

select

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name

from dim\_sku\_full

where dt='2020-06-14'

)sku

on od.sku\_id=sku.id;

**（2）每日装载**

insert overwrite table dws\_trade\_user\_sku\_order\_1d partition(dt='2020-06-15')

select

user\_id,

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name,

order\_count,

order\_num,

order\_original\_amount,

activity\_reduce\_amount,

coupon\_reduce\_amount,

order\_total\_amount

from

(

select

user\_id,

sku\_id,

count(\*) order\_count,

sum(sku\_num) order\_num,

sum(split\_original\_amount) order\_original\_amount,

sum(nvl(split\_activity\_amount,0)) activity\_reduce\_amount,

sum(nvl(split\_coupon\_amount,0)) coupon\_reduce\_amount,

sum(split\_total\_amount) order\_total\_amount

from dwd\_trade\_order\_detail\_inc

where dt='2020-06-15'

group by user\_id,sku\_id

)od

left join

(

select

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name

from dim\_sku\_full

where dt='2020-06-15'

)sku

on od.sku\_id=sku.id;

### 10.1.2 交易域用户商品粒度退单最近1日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_sku\_order\_refund\_1d;

CREATE EXTERNAL TABLE dws\_trade\_user\_sku\_order\_refund\_1d

(

`user\_id` STRING COMMENT '用户id',

`sku\_id` STRING COMMENT 'sku\_id',

`sku\_name` STRING COMMENT 'sku名称',

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

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

`category2\_id` STRING COMMENT '一级分类id',

`category2\_name` STRING COMMENT '一级分类名称',

`category3\_id` STRING COMMENT '一级分类id',

`category3\_name` STRING COMMENT '一级分类名称',

`tm\_id` STRING COMMENT '品牌id',

`tm\_name` STRING COMMENT '品牌名称',

`order\_refund\_count\_1d` BIGINT COMMENT '最近1日退单次数',

`order\_refund\_num\_1d` BIGINT COMMENT '最近1日退单件数',

`order\_refund\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日退单金额'

) COMMENT '交易域用户商品粒度退单最近1日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_sku\_order\_refund\_1d'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

**（1）首日装载**

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

insert overwrite table dws\_trade\_user\_sku\_order\_refund\_1d partition(dt)

select

user\_id,

sku\_id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name,

order\_refund\_count,

order\_refund\_num,

order\_refund\_amount,

dt

from

(

select

dt,

user\_id,

sku\_id,

count(\*) order\_refund\_count,

sum(refund\_num) order\_refund\_num,

sum(refund\_amount) order\_refund\_amount

from dwd\_trade\_order\_refund\_inc

group by dt,user\_id,sku\_id

)od

left join

(

select

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name

from dim\_sku\_full

where dt='2020-06-14'

)sku

on od.sku\_id=sku.id;

**（2）每日装载**

insert overwrite table dws\_trade\_user\_sku\_order\_refund\_1d partition(dt='2020-06-15')

select

user\_id,

sku\_id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name,

order\_refund\_count,

order\_refund\_num,

order\_refund\_amount

from

(

select

user\_id,

sku\_id,

count(\*) order\_refund\_count,

sum(refund\_num) order\_refund\_num,

sum(refund\_amount) order\_refund\_amount

from dwd\_trade\_order\_refund\_inc

where dt='2020-06-15'

group by user\_id,sku\_id

)od

left join

(

select

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name

from dim\_sku\_full

where dt='2020-06-15'

)sku

on od.sku\_id=sku.id;

### 10.1.3 交易域用户粒度订单最近1日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_order\_1d;

CREATE EXTERNAL TABLE dws\_trade\_user\_order\_1d

(

`user\_id` STRING COMMENT '用户id',

`order\_count\_1d` BIGINT COMMENT '最近1日下单次数',

`order\_num\_1d` BIGINT COMMENT '最近1日下单商品件数',

`order\_original\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日最近1日下单原始金额',

`activity\_reduce\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日下单活动优惠金额',

`coupon\_reduce\_amount\_1d` DECIMAL(16, 2) COMMENT '下单优惠券优惠金额',

`order\_total\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日下单最终金额'

) COMMENT '交易域用户粒度订单最近1日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_order\_1d'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

**（1）首日装载**

insert overwrite table dws\_trade\_user\_order\_1d partition(dt)

select

user\_id,

count(distinct(order\_id)),

sum(sku\_num),

sum(split\_original\_amount),

sum(nvl(split\_activity\_amount,0)),

sum(nvl(split\_coupon\_amount,0)),

sum(split\_total\_amount),

dt

from dwd\_trade\_order\_detail\_inc

group by user\_id,dt;

**（2）每日装载**

insert overwrite table dws\_trade\_user\_order\_1d partition(dt='2020-06-15')

select

user\_id,

count(distinct(order\_id)),

sum(sku\_num),

sum(split\_original\_amount),

sum(nvl(split\_activity\_amount,0)),

sum(nvl(split\_coupon\_amount,0)),

sum(split\_total\_amount)

from dwd\_trade\_order\_detail\_inc

where dt='2020-06-15'

group by user\_id;

### 10.1.4 交易域用户粒度加购最近1日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_cart\_add\_1d;

CREATE EXTERNAL TABLE dws\_trade\_user\_cart\_add\_1d

(

`user\_id` STRING COMMENT '用户id',

`cart\_add\_count\_1d` BIGINT COMMENT '最近1日加购次数',

`cart\_add\_num\_1d` BIGINT COMMENT '最近1日加购商品件数'

) COMMENT '交易域用户粒度加购最近1日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_cart\_add\_1d'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

**（1）首日装载**

insert overwrite table dws\_trade\_user\_cart\_add\_1d partition(dt)

select

user\_id,

count(\*),

sum(sku\_num),

dt

from dwd\_trade\_cart\_add\_inc

group by user\_id,dt;

**（2）每日装载**

insert overwrite table dws\_trade\_user\_cart\_add\_1d partition(dt='2020-06-15')

select

user\_id,

count(\*),

sum(sku\_num)

from dwd\_trade\_cart\_add\_inc

where dt='2020-06-15'

group by user\_id;

### 10.1.5 交易域用户粒度支付最近1日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_payment\_1d;

CREATE EXTERNAL TABLE dws\_trade\_user\_payment\_1d

(

`user\_id` STRING COMMENT '用户id',

`payment\_count\_1d` BIGINT COMMENT '最近1日支付次数',

`payment\_num\_1d` BIGINT COMMENT '最近1日支付商品件数',

`payment\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日支付金额'

) COMMENT '交易域用户粒度支付最近1日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_payment\_1d'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

**（1）首日装载**

insert overwrite table dws\_trade\_user\_payment\_1d partition(dt)

select

user\_id,

count(distinct(order\_id)),

sum(sku\_num),

sum(split\_payment\_amount),

dt

from dwd\_trade\_pay\_detail\_suc\_inc

group by user\_id,dt;

**（2）每日装载**

insert overwrite table dws\_trade\_user\_payment\_1d partition(dt='2020-06-15')

select

user\_id,

count(distinct(order\_id)),

sum(sku\_num),

sum(split\_payment\_amount)

from dwd\_trade\_pay\_detail\_suc\_inc

where dt='2020-06-15'

group by user\_id;

### 10.1.6 交易域省份粒度订单最近1日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_province\_order\_1d;

CREATE EXTERNAL TABLE dws\_trade\_province\_order\_1d

(

`province\_id` STRING COMMENT '用户id',

`province\_name` STRING COMMENT '省份名称',

`area\_code` STRING COMMENT '地区编码',

`iso\_code` STRING COMMENT '旧版ISO-3166-2编码',

`iso\_3166\_2` STRING COMMENT '新版版ISO-3166-2编码',

`order\_count\_1d` BIGINT COMMENT '最近1日下单次数',

`order\_original\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日下单原始金额',

`activity\_reduce\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日下单活动优惠金额',

`coupon\_reduce\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日下单优惠券优惠金额',

`order\_total\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日下单最终金额'

) COMMENT '交易域省份粒度订单最近1日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_province\_order\_1d'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

**（1）首日装载**

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

insert overwrite table dws\_trade\_province\_order\_1d partition(dt)

select

province\_id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2,

order\_count\_1d,

order\_original\_amount\_1d,

activity\_reduce\_amount\_1d,

coupon\_reduce\_amount\_1d,

order\_total\_amount\_1d,

dt

from

(

select

province\_id,

count(distinct(order\_id)) order\_count\_1d,

sum(split\_original\_amount) order\_original\_amount\_1d,

sum(nvl(split\_activity\_amount,0)) activity\_reduce\_amount\_1d,

sum(nvl(split\_coupon\_amount,0)) coupon\_reduce\_amount\_1d,

sum(split\_total\_amount) order\_total\_amount\_1d,

dt

from dwd\_trade\_order\_detail\_inc

group by province\_id,dt

)o

left join

(

select

id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2

from dim\_province\_full

where dt='2020-06-14'

)p

on o.province\_id=p.id;

**（2）每日装载**

insert overwrite table dws\_trade\_province\_order\_1d partition(dt='2020-06-15')

select

province\_id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2,

order\_count\_1d,

order\_original\_amount\_1d,

activity\_reduce\_amount\_1d,

coupon\_reduce\_amount\_1d,

order\_total\_amount\_1d

from

(

select

province\_id,

count(distinct(order\_id)) order\_count\_1d,

sum(split\_original\_amount) order\_original\_amount\_1d,

sum(nvl(split\_activity\_amount,0)) activity\_reduce\_amount\_1d,

sum(nvl(split\_coupon\_amount,0)) coupon\_reduce\_amount\_1d,

sum(split\_total\_amount) order\_total\_amount\_1d

from dwd\_trade\_order\_detail\_inc

where dt='2020-06-15'

group by province\_id

)o

left join

(

select

id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2

from dim\_province\_full

where dt='2020-06-15'

)p

on o.province\_id=p.id;

### 10.1.7 交易域用户粒度退单最近1日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_order\_refund\_1d;

CREATE EXTERNAL TABLE dws\_trade\_user\_order\_refund\_1d

(

`user\_id` STRING COMMENT '用户id',

`order\_refund\_count\_1d` BIGINT COMMENT '最近1日退单次数',

`order\_refund\_num\_1d` BIGINT COMMENT '最近1日退单商品件数',

`order\_refund\_amount\_1d` DECIMAL(16, 2) COMMENT '最近1日退单金额'

) COMMENT '交易域用户粒度退单最近1日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_order\_refund\_1d'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

**（1）首日装载**

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

insert overwrite table dws\_trade\_user\_order\_refund\_1d partition(dt)

select

user\_id,

count(\*) order\_refund\_count,

sum(refund\_num) order\_refund\_num,

sum(refund\_amount) order\_refund\_amount,

dt

from dwd\_trade\_order\_refund\_inc

group by user\_id,dt;

**（2）每日装载**

insert overwrite table dws\_trade\_user\_order\_refund\_1d partition(dt='2020-06-15')

select

user\_id,

count(\*),

sum(refund\_num),

sum(refund\_amount)

from dwd\_trade\_order\_refund\_inc

where dt='2020-06-15'

group by user\_id;

### 10.1.8 流量域会话粒度页面浏览最近1日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_traffic\_session\_page\_view\_1d;

CREATE EXTERNAL TABLE dws\_traffic\_session\_page\_view\_1d

(

`session\_id` STRING COMMENT '会话id',

`mid\_id` string comment '设备id',

`brand` string comment '手机品牌',

`model` string comment '手机型号',

`operate\_system` string comment '操作系统',

`version\_code` string comment 'app版本号',

`channel` string comment '渠道',

`during\_time\_1d` BIGINT COMMENT '最近1日访问时长',

`page\_count\_1d` BIGINT COMMENT '最近1日访问页面数'

) COMMENT '流量域会话粒度页面浏览最近1日汇总表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_traffic\_session\_page\_view\_1d'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

insert overwrite table dws\_traffic\_session\_page\_view\_1d partition(dt='2020-06-14')

select

session\_id,

mid\_id,

brand,

model,

operate\_system,

version\_code,

channel,

sum(during\_time),

count(\*)

from dwd\_traffic\_page\_view\_inc

where dt='2020-06-14'

group by session\_id,mid\_id,brand,model,operate\_system,version\_code,channel;

### 10.1.9 流量域访客页面粒度页面浏览最近1日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_traffic\_page\_visitor\_page\_view\_1d;

CREATE EXTERNAL TABLE dws\_traffic\_page\_visitor\_page\_view\_1d

(

`mid\_id` STRING COMMENT '访客id',

`brand` string comment '手机品牌',

`model` string comment '手机型号',

`operate\_system` string comment '操作系统',

`page\_id` STRING COMMENT '页面id',

`during\_time\_1d` BIGINT COMMENT '最近1日浏览时长',

`view\_count\_1d` BIGINT COMMENT '最近1日访问次数'

) COMMENT '流量域访客页面粒度页面浏览最近1日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_traffic\_page\_visitor\_page\_view\_1d'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

insert overwrite table dws\_traffic\_page\_visitor\_page\_view\_1d partition(dt='2020-06-14')

select

mid\_id,

brand,

model,

operate\_system,

page\_id,

sum(during\_time),

count(\*)

from dwd\_traffic\_page\_view\_inc

where dt='2020-06-14'

group by mid\_id,brand,model,operate\_system,page\_id;

### 10.1.10 数据装载脚本

**1）首日数据装载脚本**

（1）在hadoop102的/home/atguigu/bin目录下创建dwd\_to\_dws\_1d\_init.sh

[atguigu@hadoop102 bin]$ vim dwd\_to\_dws\_1d\_init.sh

（2）编写如下内容

#!/bin/bash

APP=gmall

if [ -n "$2" ] ;then

do\_date=$2

else

echo "请传入日期参数"

exit

fi

dws\_trade\_province\_order\_1d="

insert overwrite table ${APP}.dws\_trade\_province\_order\_1d partition(dt)

select

province\_id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2,

order\_count\_1d,

order\_original\_amount\_1d,

activity\_reduce\_amount\_1d,

coupon\_reduce\_amount\_1d,

order\_total\_amount\_1d,

dt

from

(

select

province\_id,

count(distinct(order\_id)) order\_count\_1d,

sum(split\_original\_amount) order\_original\_amount\_1d,

sum(nvl(split\_activity\_amount,0)) activity\_reduce\_amount\_1d,

sum(nvl(split\_coupon\_amount,0)) coupon\_reduce\_amount\_1d,

sum(split\_total\_amount) order\_total\_amount\_1d,

dt

from ${APP}.dwd\_trade\_order\_detail\_inc

group by province\_id,dt

)o

left join

(

select

id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2

from ${APP}.dim\_province\_full

where dt='$do\_date'

)p

on o.province\_id=p.id;

"

dws\_trade\_user\_cart\_add\_1d="

insert overwrite table ${APP}.dws\_trade\_user\_cart\_add\_1d partition(dt)

select

user\_id,

count(\*),

sum(sku\_num),

dt

from ${APP}.dwd\_trade\_cart\_add\_inc

group by user\_id,dt;

"

dws\_trade\_user\_order\_1d="

insert overwrite table ${APP}.dws\_trade\_user\_order\_1d partition(dt)

select

user\_id,

count(distinct(order\_id)),

sum(sku\_num),

sum(split\_original\_amount),

sum(nvl(split\_activity\_amount,0)),

sum(nvl(split\_coupon\_amount,0)),

sum(split\_total\_amount),

dt

from ${APP}.dwd\_trade\_order\_detail\_inc

group by user\_id,dt;

"

dws\_trade\_user\_order\_refund\_1d="

insert overwrite table ${APP}.dws\_trade\_user\_order\_refund\_1d partition(dt)

select

user\_id,

count(\*) order\_refund\_count,

sum(refund\_num) order\_refund\_num,

sum(refund\_amount) order\_refund\_amount,

dt

from ${APP}.dwd\_trade\_order\_refund\_inc

group by user\_id,dt;

"

dws\_trade\_user\_payment\_1d="

insert overwrite table ${APP}.dws\_trade\_user\_payment\_1d partition(dt)

select

user\_id,

count(distinct(order\_id)),

sum(sku\_num),

sum(split\_payment\_amount),

dt

from ${APP}.dwd\_trade\_pay\_detail\_suc\_inc

group by user\_id,dt;

"

dws\_trade\_user\_sku\_order\_1d="

insert overwrite table ${APP}.dws\_trade\_user\_sku\_order\_1d partition(dt)

select

user\_id,

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name,

order\_count\_1d,

order\_num\_1d,

order\_original\_amount\_1d,

activity\_reduce\_amount\_1d,

coupon\_reduce\_amount\_1d,

order\_total\_amount\_1d,

dt

from

(

select

dt,

user\_id,

sku\_id,

count(\*) order\_count\_1d,

sum(sku\_num) order\_num\_1d,

sum(split\_original\_amount) order\_original\_amount\_1d,

sum(nvl(split\_activity\_amount,0.0)) activity\_reduce\_amount\_1d,

sum(nvl(split\_coupon\_amount,0.0)) coupon\_reduce\_amount\_1d,

sum(split\_total\_amount) order\_total\_amount\_1d

from ${APP}.dwd\_trade\_order\_detail\_inc

group by dt,user\_id,sku\_id

)od

left join

(

select

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name

from ${APP}.dim\_sku\_full

where dt='$do\_date'

)sku

on od.sku\_id=sku.id;

"

dws\_trade\_user\_sku\_order\_refund\_1d="

insert overwrite table ${APP}.dws\_trade\_user\_sku\_order\_refund\_1d partition(dt)

select

user\_id,

sku\_id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name,

order\_refund\_count,

order\_refund\_num,

order\_refund\_amount,

dt

from

(

select

dt,

user\_id,

sku\_id,

count(\*) order\_refund\_count,

sum(refund\_num) order\_refund\_num,

sum(refund\_amount) order\_refund\_amount

from ${APP}.dwd\_trade\_order\_refund\_inc

group by dt,user\_id,sku\_id

)od

left join

(

select

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name

from ${APP}.dim\_sku\_full

where dt='$do\_date'

)sku

on od.sku\_id=sku.id;

"

dws\_traffic\_page\_visitor\_page\_view\_1d="

insert overwrite table ${APP}.dws\_traffic\_page\_visitor\_page\_view\_1d partition(dt='$do\_date')

select

mid\_id,

brand,

model,

operate\_system,

page\_id,

sum(during\_time),

count(\*)

from ${APP}.dwd\_traffic\_page\_view\_inc

where dt='$do\_date'

group by mid\_id,brand,model,operate\_system,page\_id;

"

dws\_traffic\_session\_page\_view\_1d="

insert overwrite table ${APP}.dws\_traffic\_session\_page\_view\_1d partition(dt='$do\_date')

select

session\_id,

mid\_id,

brand,

model,

operate\_system,

version\_code,

channel,

sum(during\_time),

count(\*)

from ${APP}.dwd\_traffic\_page\_view\_inc

where dt='$do\_date'

group by session\_id,mid\_id,brand,model,operate\_system,version\_code,channel;

"

case $1 in

"dws\_trade\_province\_order\_1d" )

hive -e "$dws\_trade\_province\_order\_1d"

;;

"dws\_trade\_user\_cart\_add\_1d" )

hive -e "$dws\_trade\_user\_cart\_add\_1d"

;;

"dws\_trade\_user\_order\_1d" )

hive -e "$dws\_trade\_user\_order\_1d"

;;

"dws\_trade\_user\_order\_refund\_1d" )

hive -e "$dws\_trade\_user\_order\_refund\_1d"

;;

"dws\_trade\_user\_payment\_1d" )

hive -e "$dws\_trade\_user\_payment\_1d"

;;

"dws\_trade\_user\_sku\_order\_1d" )

hive -e "$dws\_trade\_user\_sku\_order\_1d"

;;

"dws\_trade\_user\_sku\_order\_refund\_1d" )

hive -e "$dws\_trade\_user\_sku\_order\_refund\_1d"

;;

"dws\_traffic\_page\_visitor\_page\_view\_1d" )

hive -e "$dws\_traffic\_page\_visitor\_page\_view\_1d"

;;

"dws\_traffic\_session\_page\_view\_1d" )

hive -e "$dws\_traffic\_session\_page\_view\_1d"

;;

"all" )

hive -e "$dws\_trade\_province\_order\_1d$dws\_trade\_user\_cart\_add\_1d$dws\_trade\_user\_order\_1d$dws\_trade\_user\_order\_refund\_1d$dws\_trade\_user\_payment\_1d$dws\_trade\_user\_sku\_order\_1d$dws\_trade\_user\_sku\_order\_refund\_1d$dws\_traffic\_page\_visitor\_page\_view\_1d$dws\_traffic\_session\_page\_view\_1d"

;;

esac

（3）增加脚本执行权限

[atguigu@hadoop102 bin]$ chmod +x dwd\_to\_dws\_1d\_init.sh

（4）脚本用法

[atguigu@hadoop102 bin]$ dwd\_to\_dws\_1d\_init.sh all 2020-06-14

**2）每日数据装载脚本**

（1）在hadoop102的/home/atguigu/bin目录下创建dwd\_to\_dws\_1d.sh

[atguigu@hadoop102 bin]$ vim dwd\_to\_dws\_1d.sh

（2）编写如下内容

#!/bin/bash

APP=gmall

# 如果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

dws\_trade\_province\_order\_1d="

insert overwrite table ${APP}.dws\_trade\_province\_order\_1d partition(dt='$do\_date')

select

province\_id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2,

order\_count\_1d,

order\_original\_amount\_1d,

activity\_reduce\_amount\_1d,

coupon\_reduce\_amount\_1d,

order\_total\_amount\_1d

from

(

select

province\_id,

count(distinct(order\_id)) order\_count\_1d,

sum(split\_original\_amount) order\_original\_amount\_1d,

sum(nvl(split\_activity\_amount,0)) activity\_reduce\_amount\_1d,

sum(nvl(split\_coupon\_amount,0)) coupon\_reduce\_amount\_1d,

sum(split\_total\_amount) order\_total\_amount\_1d

from ${APP}.dwd\_trade\_order\_detail\_inc

where dt='$do\_date'

group by province\_id

)o

left join

(

select

id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2

from ${APP}.dim\_province\_full

where dt='$do\_date'

)p

on o.province\_id=p.id;

"

dws\_trade\_user\_cart\_add\_1d="

insert overwrite table ${APP}.dws\_trade\_user\_cart\_add\_1d partition(dt='$do\_date')

select

user\_id,

count(\*),

sum(sku\_num)

from ${APP}.dwd\_trade\_cart\_add\_inc

where dt='$do\_date'

group by user\_id;

"

dws\_trade\_user\_order\_1d="

insert overwrite table ${APP}.dws\_trade\_user\_order\_1d partition(dt='$do\_date')

select

user\_id,

count(distinct(order\_id)),

sum(sku\_num),

sum(split\_original\_amount),

sum(nvl(split\_activity\_amount,0)),

sum(nvl(split\_coupon\_amount,0)),

sum(split\_total\_amount)

from ${APP}.dwd\_trade\_order\_detail\_inc

where dt='$do\_date'

group by user\_id;

"

dws\_trade\_user\_order\_refund\_1d="

insert overwrite table ${APP}.dws\_trade\_user\_order\_refund\_1d partition(dt='$do\_date')

select

user\_id,

count(\*),

sum(refund\_num),

sum(refund\_amount)

from ${APP}.dwd\_trade\_order\_refund\_inc

where dt='$do\_date'

group by user\_id;

"

dws\_trade\_user\_payment\_1d="

insert overwrite table ${APP}.dws\_trade\_user\_payment\_1d partition(dt='$do\_date')

select

user\_id,

count(distinct(order\_id)),

sum(sku\_num),

sum(split\_payment\_amount)

from ${APP}.dwd\_trade\_pay\_detail\_suc\_inc

where dt='$do\_date'

group by user\_id;

"

dws\_trade\_user\_sku\_order\_1d="

insert overwrite table ${APP}.dws\_trade\_user\_sku\_order\_1d partition(dt='$do\_date')

select

user\_id,

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name,

order\_count,

order\_num,

order\_original\_amount,

activity\_reduce\_amount,

coupon\_reduce\_amount,

order\_total\_amount

from

(

select

user\_id,

sku\_id,

count(\*) order\_count,

sum(sku\_num) order\_num,

sum(split\_original\_amount) order\_original\_amount,

sum(nvl(split\_activity\_amount,0)) activity\_reduce\_amount,

sum(nvl(split\_coupon\_amount,0)) coupon\_reduce\_amount,

sum(split\_total\_amount) order\_total\_amount

from ${APP}.dwd\_trade\_order\_detail\_inc

where dt='$do\_date'

group by user\_id,sku\_id

)od

left join

(

select

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name

from ${APP}.dim\_sku\_full

where dt='$do\_date'

)sku

on od.sku\_id=sku.id;

"

dws\_trade\_user\_sku\_order\_refund\_1d="

insert overwrite table ${APP}.dws\_trade\_user\_sku\_order\_refund\_1d partition(dt='$do\_date')

select

user\_id,

sku\_id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name,

order\_refund\_count,

order\_refund\_num,

order\_refund\_amount

from

(

select

user\_id,

sku\_id,

count(\*) order\_refund\_count,

sum(refund\_num) order\_refund\_num,

sum(refund\_amount) order\_refund\_amount

from ${APP}.dwd\_trade\_order\_refund\_inc

where dt='$do\_date'

group by user\_id,sku\_id

)od

left join

(

select

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name

from ${APP}.dim\_sku\_full

where dt='$do\_date'

)sku

on od.sku\_id=sku.id;

"

dws\_traffic\_page\_visitor\_page\_view\_1d="

insert overwrite table ${APP}.dws\_traffic\_page\_visitor\_page\_view\_1d partition(dt='$do\_date')

select

mid\_id,

brand,

model,

operate\_system,

page\_id,

sum(during\_time),

count(\*)

from ${APP}.dwd\_traffic\_page\_view\_inc

where dt='$do\_date'

group by mid\_id,brand,model,operate\_system,page\_id;

"

dws\_traffic\_session\_page\_view\_1d="

insert overwrite table ${APP}.dws\_traffic\_session\_page\_view\_1d partition(dt='$do\_date')

select

session\_id,

mid\_id,

brand,

model,

operate\_system,

version\_code,

channel,

sum(during\_time),

count(\*)

from ${APP}.dwd\_traffic\_page\_view\_inc

where dt='$do\_date'

group by session\_id,mid\_id,brand,model,operate\_system,version\_code,channel;

"

case $1 in

"dws\_trade\_province\_order\_1d" )

hive -e "$dws\_trade\_province\_order\_1d"

;;

"dws\_trade\_user\_cart\_add\_1d" )

hive -e "$dws\_trade\_user\_cart\_add\_1d"

;;

"dws\_trade\_user\_order\_1d" )

hive -e "$dws\_trade\_user\_order\_1d"

;;

"dws\_trade\_user\_order\_refund\_1d" )

hive -e "$dws\_trade\_user\_order\_refund\_1d"

;;

"dws\_trade\_user\_payment\_1d" )

hive -e "$dws\_trade\_user\_payment\_1d"

;;

"dws\_trade\_user\_sku\_order\_1d" )

hive -e "$dws\_trade\_user\_sku\_order\_1d"

;;

"dws\_trade\_user\_sku\_order\_refund\_1d" )

hive -e "$dws\_trade\_user\_sku\_order\_refund\_1d"

;;

"dws\_traffic\_page\_visitor\_page\_view\_1d" )

hive -e "$dws\_traffic\_page\_visitor\_page\_view\_1d"

;;

"dws\_traffic\_session\_page\_view\_1d" )

hive -e "$dws\_traffic\_session\_page\_view\_1d"

;;

"all" )

hive -e "$dws\_trade\_province\_order\_1d$dws\_trade\_user\_cart\_add\_1d$dws\_trade\_user\_order\_1d$dws\_trade\_user\_order\_refund\_1d$dws\_trade\_user\_payment\_1d$dws\_trade\_user\_sku\_order\_1d$dws\_trade\_user\_sku\_order\_refund\_1d$dws\_traffic\_page\_visitor\_page\_view\_1d$dws\_traffic\_session\_page\_view\_1d"

;;

esac

（3）增加脚本执行权限

[atguigu@hadoop102 bin]$ chmod +x dwd\_to\_dws\_1d.sh

（4）脚本用法

[atguigu@hadoop102 bin]$ dwd\_to\_dws\_1d.sh all 2020-06-14

## 10.2 最近n日汇总表

### 10.2.1 交易域用户商品粒度订单最近n日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_sku\_order\_nd;

CREATE EXTERNAL TABLE dws\_trade\_user\_sku\_order\_nd

(

`user\_id` STRING COMMENT '用户id',

`sku\_id` STRING COMMENT 'sku\_id',

`sku\_name` STRING COMMENT 'sku名称',

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

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

`category2\_id` STRING COMMENT '一级分类id',

`category2\_name` STRING COMMENT '一级分类名称',

`category3\_id` STRING COMMENT '一级分类id',

`category3\_name` STRING COMMENT '一级分类名称',

`tm\_id` STRING COMMENT '品牌id',

`tm\_name` STRING COMMENT '品牌名称',

`order\_count\_7d` STRING COMMENT '最近7日下单次数',

`order\_num\_7d` BIGINT COMMENT '最近7日下单件数',

`order\_original\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日下单原始金额',

`activity\_reduce\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日活动优惠金额',

`coupon\_reduce\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日优惠券优惠金额',

`order\_total\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日下单最终金额',

`order\_count\_30d` BIGINT COMMENT '最近30日下单次数',

`order\_num\_30d` BIGINT COMMENT '最近30日下单件数',

`order\_original\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日下单原始金额',

`activity\_reduce\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日活动优惠金额',

`coupon\_reduce\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日优惠券优惠金额',

`order\_total\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日下单最终金额'

) COMMENT '交易域用户商品粒度订单最近n日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_sku\_order\_nd'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

insert overwrite table dws\_trade\_user\_sku\_order\_nd partition(dt='2020-06-14')

select

user\_id,

sku\_id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name,

sum(if(dt>=date\_add('2020-06-14',-6),order\_count\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),order\_num\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),order\_original\_amount\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),activity\_reduce\_amount\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),coupon\_reduce\_amount\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),order\_total\_amount\_1d,0)),

sum(order\_count\_1d),

sum(order\_num\_1d),

sum(order\_original\_amount\_1d),

sum(activity\_reduce\_amount\_1d),

sum(coupon\_reduce\_amount\_1d),

sum(order\_total\_amount\_1d)

from dws\_trade\_user\_sku\_order\_1d

where dt>=date\_add('2020-06-14',-29)

group by user\_id,sku\_id,sku\_name,category1\_id,category1\_name,category2\_id,category2\_name,category3\_id,category3\_name,tm\_id,tm\_name;

### 10.2.2 交易域用户商品粒度退单最近n日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_sku\_order\_refund\_nd;

CREATE EXTERNAL TABLE dws\_trade\_user\_sku\_order\_refund\_nd

(

`user\_id` STRING COMMENT '用户id',

`sku\_id` STRING COMMENT 'sku\_id',

`sku\_name` STRING COMMENT 'sku名称',

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

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

`category2\_id` STRING COMMENT '一级分类id',

`category2\_name` STRING COMMENT '一级分类名称',

`category3\_id` STRING COMMENT '一级分类id',

`category3\_name` STRING COMMENT '一级分类名称',

`tm\_id` STRING COMMENT '品牌id',

`tm\_name` STRING COMMENT '品牌名称',

`order\_refund\_count\_7d` BIGINT COMMENT '最近7日退单次数',

`order\_refund\_num\_7d` BIGINT COMMENT '最近7日退单件数',

`order\_refund\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日退单金额',

`order\_refund\_count\_30d` BIGINT COMMENT '最近30日退单次数',

`order\_refund\_num\_30d` BIGINT COMMENT '最近30日退单件数',

`order\_refund\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日退单金额'

) COMMENT '交易域用户商品粒度退单最近n日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_sku\_order\_refund\_nd'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

insert overwrite table dws\_trade\_user\_sku\_order\_refund\_nd partition(dt='2020-06-14')

select

user\_id,

sku\_id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name,

sum(if(dt>=date\_add('2020-06-14',-6),order\_refund\_count\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),order\_refund\_num\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),order\_refund\_amount\_1d,0)),

sum(order\_refund\_count\_1d),

sum(order\_refund\_num\_1d),

sum(order\_refund\_amount\_1d)

from dws\_trade\_user\_sku\_order\_refund\_1d

where dt>=date\_add('2020-06-14',-29)

and dt<='2020-06-14'

group by user\_id,sku\_id,sku\_name,category1\_id,category1\_name,category2\_id,category2\_name,category3\_id,category3\_name,tm\_id,tm\_name;

### 10.2.3 交易域用户粒度订单最近n日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_order\_nd;

CREATE EXTERNAL TABLE dws\_trade\_user\_order\_nd

(

`user\_id` STRING COMMENT '用户id',

`order\_count\_7d` BIGINT COMMENT '最近7日下单次数',

`order\_num\_7d` BIGINT COMMENT '最近7日下单商品件数',

`order\_original\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日下单原始金额',

`activity\_reduce\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日下单活动优惠金额',

`coupon\_reduce\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日下单优惠券优惠金额',

`order\_total\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日下单最终金额',

`order\_count\_30d` BIGINT COMMENT '最近30日下单次数',

`order\_num\_30d` BIGINT COMMENT '最近30日下单商品件数',

`order\_original\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日下单原始金额',

`activity\_reduce\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日下单活动优惠金额',

`coupon\_reduce\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日下单优惠券优惠金额',

`order\_total\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日下单最终金额'

) COMMENT '交易域用户粒度订单最近n日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_order\_nd'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

insert overwrite table dws\_trade\_user\_order\_nd partition(dt='2020-06-14')

select

user\_id,

sum(if(dt>=date\_add('2020-06-14',-6),order\_count\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),order\_num\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),order\_original\_amount\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),activity\_reduce\_amount\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),coupon\_reduce\_amount\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),order\_total\_amount\_1d,0)),

sum(order\_count\_1d),

sum(order\_num\_1d),

sum(order\_original\_amount\_1d),

sum(activity\_reduce\_amount\_1d),

sum(coupon\_reduce\_amount\_1d),

sum(order\_total\_amount\_1d)

from dws\_trade\_user\_order\_1d

where dt>=date\_add('2020-06-14',-29)

and dt<='2020-06-14'

group by user\_id;

### 10.2.4 交易域用户粒度加购最近n日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_cart\_add\_nd;

CREATE EXTERNAL TABLE dws\_trade\_user\_cart\_add\_nd

(

`user\_id` STRING COMMENT '用户id',

`cart\_add\_count\_7d` BIGINT COMMENT '最近7日加购次数',

`cart\_add\_num\_7d` BIGINT COMMENT '最近7日加购商品件数',

`cart\_add\_count\_30d` BIGINT COMMENT '最近30日加购次数',

`cart\_add\_num\_30d` BIGINT COMMENT '最近30日加购商品件数'

) COMMENT '交易域用户粒度加购最近n日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_cart\_add\_nd'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

insert overwrite table dws\_trade\_user\_cart\_add\_nd partition(dt='2020-06-14')

select

user\_id,

sum(if(dt>=date\_add('2020-06-14',-6),cart\_add\_count\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),cart\_add\_num\_1d,0)),

sum(cart\_add\_count\_1d),

sum(cart\_add\_num\_1d)

from dws\_trade\_user\_cart\_add\_1d

where dt>=date\_add('2020-06-14',-29)

and dt<='2020-06-14'

group by user\_id;

### 10.2.5 交易域用户粒度支付最近n日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_payment\_nd;

CREATE EXTERNAL TABLE dws\_trade\_user\_payment\_nd

(

`user\_id` STRING COMMENT '用户id',

`payment\_count\_7d` BIGINT COMMENT '最近7日支付次数',

`payment\_num\_7d` BIGINT COMMENT '最近7日支付商品件数',

`payment\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日支付金额',

`payment\_count\_30d` BIGINT COMMENT '最近30日支付次数',

`payment\_num\_30d` BIGINT COMMENT '最近30日支付商品件数',

`payment\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日支付金额'

) COMMENT '交易域用户粒度支付最近n日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_payment\_nd'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

insert overwrite table dws\_trade\_user\_payment\_nd partition (dt = '2020-06-14')

select user\_id,

sum(if(dt >= date\_add('2020-06-14', -6), payment\_count\_1d, 0)),

sum(if(dt >= date\_add('2020-06-14', -6), payment\_num\_1d, 0)),

sum(if(dt >= date\_add('2020-06-14', -6), payment\_amount\_1d, 0)),

sum(payment\_count\_1d),

sum(payment\_num\_1d),

sum(payment\_amount\_1d)

from dws\_trade\_user\_payment\_1d

where dt >= date\_add('2020-06-14', -29)

and dt <= '2020-06-14'

group by user\_id;

### 10.2.6 交易域省份粒度订单最近n日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_province\_order\_nd;

CREATE EXTERNAL TABLE dws\_trade\_province\_order\_nd

(

`province\_id` STRING COMMENT '用户id',

`province\_name` STRING COMMENT '省份名称',

`area\_code` STRING COMMENT '地区编码',

`iso\_code` STRING COMMENT '旧版ISO-3166-2编码',

`iso\_3166\_2` STRING COMMENT '新版版ISO-3166-2编码',

`order\_count\_7d` BIGINT COMMENT '最近7日下单次数',

`order\_original\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日下单原始金额',

`activity\_reduce\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日下单活动优惠金额',

`coupon\_reduce\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日下单优惠券优惠金额',

`order\_total\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日下单最终金额',

`order\_count\_30d` BIGINT COMMENT '最近30日下单次数',

`order\_original\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日下单原始金额',

`activity\_reduce\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日下单活动优惠金额',

`coupon\_reduce\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日下单优惠券优惠金额',

`order\_total\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日下单最终金额'

) COMMENT '交易域省份粒度订单最近n日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_province\_order\_nd'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

insert overwrite table dws\_trade\_province\_order\_nd partition(dt='2020-06-14')

select

province\_id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2,

sum(if(dt>=date\_add('2020-06-14',-6),order\_count\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),order\_original\_amount\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),activity\_reduce\_amount\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),coupon\_reduce\_amount\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),order\_total\_amount\_1d,0)),

sum(order\_count\_1d),

sum(order\_original\_amount\_1d),

sum(activity\_reduce\_amount\_1d),

sum(coupon\_reduce\_amount\_1d),

sum(order\_total\_amount\_1d)

from dws\_trade\_province\_order\_1d

where dt>=date\_add('2020-06-14',-29)

and dt<='2020-06-14'

group by province\_id,province\_name,area\_code,iso\_code,iso\_3166\_2;

### 10.2.7 交易域优惠券粒度订单最近n日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_coupon\_order\_nd;

CREATE EXTERNAL TABLE dws\_trade\_coupon\_order\_nd

(

`coupon\_id` STRING COMMENT '优惠券id',

`coupon\_name` STRING COMMENT '优惠券名称',

`coupon\_type\_code` STRING COMMENT '优惠券类型id',

`coupon\_type\_name` STRING COMMENT '优惠券类型名称',

`coupon\_rule` STRING COMMENT '优惠券规则',

`start\_date` STRING COMMENT '发布日期',

`original\_amount\_30d` DECIMAL(16, 2) COMMENT '使用下单原始金额',

`coupon\_reduce\_amount\_30d` DECIMAL(16, 2) COMMENT '使用下单优惠金额'

) COMMENT '交易域优惠券粒度订单最近n日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_coupon\_order\_nd'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

insert overwrite table dws\_trade\_coupon\_order\_nd partition(dt='2020-06-14')

select

id,

coupon\_name,

coupon\_type\_code,

coupon\_type\_name,

benefit\_rule,

start\_date,

sum(split\_original\_amount),

sum(split\_coupon\_amount)

from

(

select

id,

coupon\_name,

coupon\_type\_code,

coupon\_type\_name,

benefit\_rule,

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

from dim\_coupon\_full

where dt='2020-06-14'

and date\_format(start\_time,'yyyy-MM-dd')>=date\_add('2020-06-14',-29)

)cou

left join

(

select

coupon\_id,

order\_id,

split\_original\_amount,

split\_coupon\_amount

from dwd\_trade\_order\_detail\_inc

where dt>=date\_add('2020-06-14',-29)

and dt<='2020-06-14'

and coupon\_id is not null

)od

on cou.id=od.coupon\_id

group by id,coupon\_name,coupon\_type\_code,coupon\_type\_name,benefit\_rule,start\_date;

### 10.2.8 交易域活动粒度订单最近n日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_activity\_order\_nd;

CREATE EXTERNAL TABLE dws\_trade\_activity\_order\_nd

(

`activity\_id` STRING COMMENT '活动id',

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

`activity\_type\_code` STRING COMMENT '活动类型编码',

`activity\_type\_name` STRING COMMENT '活动类型名称',

`start\_date` STRING COMMENT '发布日期',

`original\_amount\_30d` DECIMAL(16, 2) COMMENT '参与活动订单原始金额',

`activity\_reduce\_amount\_30d` DECIMAL(16, 2) COMMENT '参与活动订单优惠金额'

) COMMENT '交易域活动粒度订单最近n日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_activity\_order\_nd'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

insert overwrite table dws\_trade\_activity\_order\_nd partition(dt='2020-06-14')

select

act.activity\_id,

activity\_name,

activity\_type\_code,

activity\_type\_name,

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

sum(split\_original\_amount),

sum(split\_activity\_amount)

from

(

select

activity\_id,

activity\_name,

activity\_type\_code,

activity\_type\_name,

start\_time

from dim\_activity\_full

where dt='2020-06-14'

and date\_format(start\_time,'yyyy-MM-dd')>=date\_add('2020-06-14',-29)

group by activity\_id, activity\_name, activity\_type\_code, activity\_type\_name,start\_time

)act

left join

(

select

activity\_id,

order\_id,

split\_original\_amount,

split\_activity\_amount

from dwd\_trade\_order\_detail\_inc

where dt>=date\_add('2020-06-14',-29)

and dt<='2020-06-14'

and activity\_id is not null

)od

on act.activity\_id=od.activity\_id

group by act.activity\_id,activity\_name,activity\_type\_code,activity\_type\_name,start\_time;

### 10.2.9 交易域用户粒度退单最近n日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_order\_refund\_nd;

CREATE EXTERNAL TABLE dws\_trade\_user\_order\_refund\_nd

(

`user\_id` STRING COMMENT '用户id',

`order\_refund\_count\_7d` BIGINT COMMENT '最近7日退单次数',

`order\_refund\_num\_7d` BIGINT COMMENT '最近7日退单商品件数',

`order\_refund\_amount\_7d` DECIMAL(16, 2) COMMENT '最近7日退单金额',

`order\_refund\_count\_30d` BIGINT COMMENT '最近30日退单次数',

`order\_refund\_num\_30d` BIGINT COMMENT '最近30日退单商品件数',

`order\_refund\_amount\_30d` DECIMAL(16, 2) COMMENT '最近30日退单金额'

) COMMENT '交易域用户粒度退单最近n日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_order\_refund\_nd'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

insert overwrite table dws\_trade\_user\_order\_refund\_nd partition(dt='2020-06-14')

select

user\_id,

sum(if(dt>=date\_add('2020-06-14',-6),order\_refund\_count\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),order\_refund\_num\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),order\_refund\_amount\_1d,0)),

sum(order\_refund\_count\_1d),

sum(order\_refund\_num\_1d),

sum(order\_refund\_amount\_1d)

from dws\_trade\_user\_order\_refund\_1d

where dt>=date\_add('2020-06-14',-29)

and dt<='2020-06-14'

group by user\_id;

### 10.2.10 流量域访客页面粒度页面浏览最近n日汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_traffic\_page\_visitor\_page\_view\_nd;

CREATE EXTERNAL TABLE dws\_traffic\_page\_visitor\_page\_view\_nd

(

`mid\_id` STRING COMMENT '访客id',

`brand` string comment '手机品牌',

`model` string comment '手机型号',

`operate\_system` string comment '操作系统',

`page\_id` STRING COMMENT '页面id',

`during\_time\_7d` BIGINT COMMENT '最近7日浏览时长',

`view\_count\_7d` BIGINT COMMENT '最近7日访问次数',

`during\_time\_30d` BIGINT COMMENT '最近30日浏览时长',

`view\_count\_30d` BIGINT COMMENT '最近30日访问次数'

) COMMENT '流量域访客页面粒度页面浏览最近n日汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_traffic\_page\_visitor\_page\_view\_nd'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

insert overwrite table dws\_traffic\_page\_visitor\_page\_view\_nd partition(dt='2020-06-14')

select

mid\_id,

brand,

model,

operate\_system,

page\_id,

sum(if(dt>=date\_add('2020-06-14',-6),during\_time\_1d,0)),

sum(if(dt>=date\_add('2020-06-14',-6),view\_count\_1d,0)),

sum(during\_time\_1d),

sum(view\_count\_1d)

from dws\_traffic\_page\_visitor\_page\_view\_1d

where dt>=date\_add('2020-06-14',-29)

and dt<='2020-06-14'

group by mid\_id,brand,model,operate\_system,page\_id;

### 10.2.11 数据装载脚本

**1）每日数据装载脚本**

（1）在hadoop102的/home/atguigu/bin目录下创建dws\_1d\_to\_dws\_nd.sh

[atguigu@hadoop102 bin]$ vim dws\_1d\_to\_dws\_nd.sh

（2）编写如下内容

#!/bin/bash

APP=gmall

# 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

dws\_trade\_activity\_order\_nd="

insert overwrite table ${APP}.dws\_trade\_activity\_order\_nd partition(dt='$do\_date')

select

act.activity\_id,

activity\_name,

activity\_type\_code,

activity\_type\_name,

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

sum(split\_original\_amount),

sum(split\_activity\_amount)

from

(

select

activity\_id,

activity\_name,

activity\_type\_code,

activity\_type\_name,

start\_time

from ${APP}.dim\_activity\_full

where dt='$do\_date'

and date\_format(start\_time,'yyyy-MM-dd')>=date\_add('$do\_date',-29)

group by activity\_id, activity\_name, activity\_type\_code, activity\_type\_name,start\_time

)act

left join

(

select

activity\_id,

order\_id,

split\_original\_amount,

split\_activity\_amount

from ${APP}.dwd\_trade\_order\_detail\_inc

where dt>=date\_add('$do\_date',-29)

and dt<='$do\_date'

and activity\_id is not null

)od

on act.activity\_id=od.activity\_id

group by act.activity\_id,activity\_name,activity\_type\_code,activity\_type\_name,start\_time;

"

dws\_trade\_coupon\_order\_nd="

insert overwrite table ${APP}.dws\_trade\_coupon\_order\_nd partition(dt='$do\_date')

select

id,

coupon\_name,

coupon\_type\_code,

coupon\_type\_name,

benefit\_rule,

start\_date,

sum(split\_original\_amount),

sum(split\_coupon\_amount)

from

(

select

id,

coupon\_name,

coupon\_type\_code,

coupon\_type\_name,

benefit\_rule,

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

from ${APP}.dim\_coupon\_full

where dt='$do\_date'

and date\_format(start\_time,'yyyy-MM-dd')>=date\_add('$do\_date',-29)

)cou

left join

(

select

coupon\_id,

order\_id,

split\_original\_amount,

split\_coupon\_amount

from ${APP}.dwd\_trade\_order\_detail\_inc

where dt>=date\_add('$do\_date',-29)

and dt<='$do\_date'

and coupon\_id is not null

)od

on cou.id=od.coupon\_id

group by id,coupon\_name,coupon\_type\_code,coupon\_type\_name,benefit\_rule,start\_date;

"

dws\_trade\_province\_order\_nd="

insert overwrite table ${APP}.dws\_trade\_province\_order\_nd partition(dt='$do\_date')

select

province\_id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2,

sum(if(dt>=date\_add('$do\_date',-6),order\_count\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),order\_original\_amount\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),activity\_reduce\_amount\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),coupon\_reduce\_amount\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),order\_total\_amount\_1d,0)),

sum(order\_count\_1d),

sum(order\_original\_amount\_1d),

sum(activity\_reduce\_amount\_1d),

sum(coupon\_reduce\_amount\_1d),

sum(order\_total\_amount\_1d)

from ${APP}.dws\_trade\_province\_order\_1d

where dt>=date\_add('$do\_date',-29)

and dt<='$do\_date'

group by province\_id,province\_name,area\_code,iso\_code,iso\_3166\_2;

"

dws\_trade\_user\_cart\_add\_nd="

insert overwrite table ${APP}.dws\_trade\_user\_cart\_add\_nd partition(dt='$do\_date')

select

user\_id,

sum(if(dt>=date\_add('$do\_date',-6),cart\_add\_count\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),cart\_add\_num\_1d,0)),

sum(cart\_add\_count\_1d),

sum(cart\_add\_num\_1d)

from ${APP}.dws\_trade\_user\_cart\_add\_1d

where dt>=date\_add('$do\_date',-29)

and dt<='$do\_date'

group by user\_id;

"

dws\_trade\_user\_order\_nd="

insert overwrite table ${APP}.dws\_trade\_user\_order\_nd partition(dt='$do\_date')

select

user\_id,

sum(if(dt>=date\_add('$do\_date',-6),order\_count\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),order\_num\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),order\_original\_amount\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),activity\_reduce\_amount\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),coupon\_reduce\_amount\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),order\_total\_amount\_1d,0)),

sum(order\_count\_1d),

sum(order\_num\_1d),

sum(order\_original\_amount\_1d),

sum(activity\_reduce\_amount\_1d),

sum(coupon\_reduce\_amount\_1d),

sum(order\_total\_amount\_1d)

from ${APP}.dws\_trade\_user\_order\_1d

where dt>=date\_add('$do\_date',-29)

and dt<='$do\_date'

group by user\_id;

"

dws\_trade\_user\_order\_refund\_nd="

insert overwrite table ${APP}.dws\_trade\_user\_order\_refund\_nd partition(dt='$do\_date')

select

user\_id,

sum(if(dt>=date\_add('$do\_date',-6),order\_refund\_count\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),order\_refund\_num\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),order\_refund\_amount\_1d,0)),

sum(order\_refund\_count\_1d),

sum(order\_refund\_num\_1d),

sum(order\_refund\_amount\_1d)

from ${APP}.dws\_trade\_user\_order\_refund\_1d

where dt>=date\_add('$do\_date',-29)

and dt<='$do\_date'

group by user\_id;

"

dws\_trade\_user\_payment\_nd="

insert overwrite table ${APP}.dws\_trade\_user\_payment\_nd partition (dt = '$do\_date')

select user\_id,

sum(if(dt >= date\_add('$do\_date', -6), payment\_count\_1d, 0)),

sum(if(dt >= date\_add('$do\_date', -6), payment\_num\_1d, 0)),

sum(if(dt >= date\_add('$do\_date', -6), payment\_amount\_1d, 0)),

sum(payment\_count\_1d),

sum(payment\_num\_1d),

sum(payment\_amount\_1d)

from ${APP}.dws\_trade\_user\_payment\_1d

where dt >= date\_add('$do\_date', -29)

and dt <= '$do\_date'

group by user\_id;

"

dws\_trade\_user\_sku\_order\_nd="

insert overwrite table ${APP}.dws\_trade\_user\_sku\_order\_nd partition(dt='$do\_date')

select

user\_id,

sku\_id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name,

sum(if(dt>=date\_add('$do\_date',-6),order\_count\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),order\_num\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),order\_original\_amount\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),activity\_reduce\_amount\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),coupon\_reduce\_amount\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),order\_total\_amount\_1d,0)),

sum(order\_count\_1d),

sum(order\_num\_1d),

sum(order\_original\_amount\_1d),

sum(activity\_reduce\_amount\_1d),

sum(coupon\_reduce\_amount\_1d),

sum(order\_total\_amount\_1d)

from ${APP}.dws\_trade\_user\_sku\_order\_1d

where dt>=date\_add('$do\_date',-30)

group by user\_id,sku\_id,sku\_name,category1\_id,category1\_name,category2\_id,category2\_name,category3\_id,category3\_name,tm\_id,tm\_name;

"

dws\_trade\_user\_sku\_order\_refund\_nd="

insert overwrite table ${APP}.dws\_trade\_user\_sku\_order\_refund\_nd partition(dt='$do\_date')

select

user\_id,

sku\_id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

tm\_id,

tm\_name,

sum(if(dt>=date\_add('$do\_date',-6),order\_refund\_count\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),order\_refund\_num\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),order\_refund\_amount\_1d,0)),

sum(order\_refund\_count\_1d),

sum(order\_refund\_num\_1d),

sum(order\_refund\_amount\_1d)

from ${APP}.dws\_trade\_user\_sku\_order\_refund\_1d

where dt>=date\_add('$do\_date',-29)

and dt<='$do\_date'

group by user\_id,sku\_id,sku\_name,category1\_id,category1\_name,category2\_id,category2\_name,category3\_id,category3\_name,tm\_id,tm\_name;

"

dws\_traffic\_page\_visitor\_page\_view\_nd="

insert overwrite table ${APP}.dws\_traffic\_page\_visitor\_page\_view\_nd partition(dt='$do\_date')

select

mid\_id,

brand,

model,

operate\_system,

page\_id,

sum(if(dt>=date\_add('$do\_date',-6),during\_time\_1d,0)),

sum(if(dt>=date\_add('$do\_date',-6),view\_count\_1d,0)),

sum(during\_time\_1d),

sum(view\_count\_1d)

from ${APP}.dws\_traffic\_page\_visitor\_page\_view\_1d

where dt>=date\_add('$do\_date',-29)

and dt<='$do\_date'

group by mid\_id,brand,model,operate\_system,page\_id;

"

case $1 in

"dws\_trade\_activity\_order\_nd" )

hive -e "$dws\_trade\_activity\_order\_nd"

;;

"dws\_trade\_coupon\_order\_nd" )

hive -e "$dws\_trade\_coupon\_order\_nd"

;;

"dws\_trade\_province\_order\_nd" )

hive -e "$dws\_trade\_province\_order\_nd"

;;

"dws\_trade\_user\_cart\_add\_nd" )

hive -e "$dws\_trade\_user\_cart\_add\_nd"

;;

"dws\_trade\_user\_order\_nd" )

hive -e "$dws\_trade\_user\_order\_nd"

;;

"dws\_trade\_user\_order\_refund\_nd" )

hive -e "$dws\_trade\_user\_order\_refund\_nd"

;;

"dws\_trade\_user\_payment\_nd" )

hive -e "$dws\_trade\_user\_payment\_nd"

;;

"dws\_trade\_user\_sku\_order\_nd" )

hive -e "$dws\_trade\_user\_sku\_order\_nd"

;;

"dws\_trade\_user\_sku\_order\_refund\_nd" )

hive -e "$dws\_trade\_user\_sku\_order\_refund\_nd"

;;

"dws\_traffic\_page\_visitor\_page\_view\_nd" )

hive -e "$dws\_traffic\_page\_visitor\_page\_view\_nd"

;;

"all" )

hive -e "$dws\_trade\_activity\_order\_nd$dws\_trade\_coupon\_order\_nd$dws\_trade\_province\_order\_nd$dws\_trade\_user\_cart\_add\_nd$dws\_trade\_user\_order\_nd$dws\_trade\_user\_order\_refund\_nd$dws\_trade\_user\_payment\_nd$dws\_trade\_user\_sku\_order\_nd$dws\_trade\_user\_sku\_order\_refund\_nd$dws\_traffic\_page\_visitor\_page\_view\_nd"

;;

esac

（3）增加脚本执行权限

[atguigu@hadoop102 bin]$ chmod +x dws\_1d\_to\_dws\_nd.sh

（4）脚本用法

[atguigu@hadoop102 bin]$ dws\_1d\_to\_dws\_nd.sh all 2020-06-14

## 10.3 历史至今汇总表

### 10.3.1 交易域用户粒度订单历史至今汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_order\_td;

CREATE EXTERNAL TABLE dws\_trade\_user\_order\_td

(

`user\_id` STRING COMMENT '用户id',

`order\_date\_first` STRING COMMENT '首次下单日期',

`order\_date\_last` STRING COMMENT '末次下单日期',

`order\_count\_td` BIGINT COMMENT '下单次数',

`order\_num\_td` BIGINT COMMENT '购买商品件数',

`original\_amount\_td` DECIMAL(16, 2) COMMENT '原始金额',

`activity\_reduce\_amount\_td` DECIMAL(16, 2) COMMENT '活动优惠金额',

`coupon\_reduce\_amount\_td` DECIMAL(16, 2) COMMENT '优惠券优惠金额',

`total\_amount\_td` DECIMAL(16, 2) COMMENT '最终金额'

) COMMENT '交易域用户粒度订单历史至今汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_order\_td'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

**（1）首日装载**

insert overwrite table dws\_trade\_user\_order\_td partition(dt='2020-06-14')

select

user\_id,

min(dt) login\_date\_first,

max(dt) login\_date\_last,

sum(order\_count\_1d) order\_count,

sum(order\_num\_1d) order\_num,

sum(order\_original\_amount\_1d) original\_amount,

sum(activity\_reduce\_amount\_1d) activity\_reduce\_amount,

sum(coupon\_reduce\_amount\_1d) coupon\_reduce\_amount,

sum(order\_total\_amount\_1d) total\_amount

from dws\_trade\_user\_order\_1d

group by user\_id;

**（2）每日装载**

insert overwrite table dws\_trade\_user\_order\_td partition(dt='2020-06-15')

select

nvl(old.user\_id,new.user\_id),

if(new.user\_id is not null and old.user\_id is null,'2020-06-15',old.order\_date\_first),

if(new.user\_id is not null,'2020-06-15',old.order\_date\_last),

nvl(old.order\_count\_td,0)+nvl(new.order\_count\_1d,0),

nvl(old.order\_num\_td,0)+nvl(new.order\_num\_1d,0),

nvl(old.original\_amount\_td,0)+nvl(new.order\_original\_amount\_1d,0),

nvl(old.activity\_reduce\_amount\_td,0)+nvl(new.activity\_reduce\_amount\_1d,0),

nvl(old.coupon\_reduce\_amount\_td,0)+nvl(new.coupon\_reduce\_amount\_1d,0),

nvl(old.total\_amount\_td,0)+nvl(new.order\_total\_amount\_1d,0)

from

(

select

user\_id,

order\_date\_first,

order\_date\_last,

order\_count\_td,

order\_num\_td,

original\_amount\_td,

activity\_reduce\_amount\_td,

coupon\_reduce\_amount\_td,

total\_amount\_td

from dws\_trade\_user\_order\_td

where dt=date\_add('2020-06-15',-1)

)old

full outer join

(

select

user\_id,

order\_count\_1d,

order\_num\_1d,

order\_original\_amount\_1d,

activity\_reduce\_amount\_1d,

coupon\_reduce\_amount\_1d,

order\_total\_amount\_1d

from dws\_trade\_user\_order\_1d

where dt='2020-06-15'

)new

on old.user\_id=new.user\_id;

### 10.3.2 交易域用户粒度支付历史至今汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_trade\_user\_payment\_td;

CREATE EXTERNAL TABLE dws\_trade\_user\_payment\_td

(

`user\_id` STRING COMMENT '用户id',

`payment\_date\_first` STRING COMMENT '首次支付日期',

`payment\_date\_last` STRING COMMENT '末次支付日期',

`payment\_count\_td` BIGINT COMMENT '最近7日支付次数',

`payment\_num\_td` BIGINT COMMENT '最近7日支付商品件数',

`payment\_amount\_td` DECIMAL(16, 2) COMMENT '最近7日支付金额'

) COMMENT '交易域用户粒度支付历史至今汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_trade\_user\_payment\_td'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

**（1）首日装载**

insert overwrite table dws\_trade\_user\_payment\_td partition(dt='2020-06-14')

select

user\_id,

min(dt) payment\_date\_first,

max(dt) payment\_date\_last,

sum(payment\_count\_1d) payment\_count,

sum(payment\_num\_1d) payment\_num,

sum(payment\_amount\_1d) payment\_amount

from dws\_trade\_user\_payment\_1d

group by user\_id;

**（2）每日装载**

insert overwrite table dws\_trade\_user\_payment\_td partition(dt='2020-06-15')

select

nvl(old.user\_id,new.user\_id),

if(old.user\_id is null and new.user\_id is not null,'2020-06-15',old.payment\_date\_first),

if(new.user\_id is not null,'2020-06-15',old.payment\_date\_last),

nvl(old.payment\_count\_td,0)+nvl(new.payment\_count\_1d,0),

nvl(old.payment\_num\_td,0)+nvl(new.payment\_num\_1d,0),

nvl(old.payment\_amount\_td,0)+nvl(new.payment\_amount\_1d,0)

from

(

select

user\_id,

payment\_date\_first,

payment\_date\_last,

payment\_count\_td,

payment\_num\_td,

payment\_amount\_td

from dws\_trade\_user\_payment\_td

where dt=date\_add('2020-06-15',-1)

)old

full outer join

(

select

user\_id,

payment\_count\_1d,

payment\_num\_1d,

payment\_amount\_1d

from dws\_trade\_user\_payment\_1d

where dt='2020-06-15'

)new

on old.user\_id=new.user\_id;

### 10.3.3 用户域用户粒度登录历史至今汇总表

**1）建表语句**

DROP TABLE IF EXISTS dws\_user\_user\_login\_td;

CREATE EXTERNAL TABLE dws\_user\_user\_login\_td

(

`user\_id` STRING COMMENT '用户id',

`login\_date\_last` STRING COMMENT '末次登录日期',

`login\_count\_td` BIGINT COMMENT '累计登录次数'

) COMMENT '用户域用户粒度登录历史至今汇总事实表'

PARTITIONED BY (`dt` STRING)

STORED AS ORC

LOCATION '/warehouse/gmall/dws/dws\_user\_user\_login\_td'

TBLPROPERTIES ('orc.compress' = 'snappy');

**2）数据装载**

**（1）首日装载**

insert overwrite table dws\_user\_user\_login\_td partition(dt='2020-06-14')

select

u.id,

nvl(login\_date\_last,date\_format(create\_time,'yyyy-MM-dd')),

nvl(login\_count\_td,1)

from

(

select

id,

create\_time

from dim\_user\_zip

where dt='9999-12-31'

)u

left join

(

select

user\_id,

max(dt) login\_date\_last,

count(\*) login\_count\_td

from dwd\_user\_login\_inc

group by user\_id

)l

on u.id=l.user\_id;

**（2）每日装载**

insert overwrite table dws\_user\_user\_login\_td partition(dt='2020-06-15')

select

nvl(old.user\_id,new.user\_id),

if(new.user\_id is null,old.login\_date\_last,'2020-06-15'),

nvl(old.login\_count\_td,0)+nvl(new.login\_count\_1d,0)

from

(

select

user\_id,

login\_date\_last,

login\_count\_td

from dws\_user\_user\_login\_td

where dt=date\_add('2020-06-15',-1)

)old

full outer join

(

select

user\_id,

count(\*) login\_count\_1d

from dwd\_user\_login\_inc

where dt='2020-06-15'

group by user\_id

)new

on old.user\_id=new.user\_id;

### 10.3.4 数据装载脚本

**1）首日数据装载脚本**

（1）在hadoop102的/home/atguigu/bin目录下创建dws\_1d\_to\_dws\_td\_init.sh

[atguigu@hadoop102 bin]$ vim dws\_1d\_to\_dws\_td\_init.sh

（2）编写如下内容

#!/bin/bash

APP=gmall

if [ -n "$2" ] ;then

do\_date=$2

else

echo "请传入日期参数"

exit

fi

dws\_trade\_user\_order\_td="

insert overwrite table ${APP}.dws\_trade\_user\_order\_td partition(dt='$do\_date')

select

user\_id,

min(dt) login\_date\_first,

max(dt) login\_date\_last,

sum(order\_count\_1d) order\_count,

sum(order\_num\_1d) order\_num,

sum(order\_original\_amount\_1d) original\_amount,

sum(activity\_reduce\_amount\_1d) activity\_reduce\_amount,

sum(coupon\_reduce\_amount\_1d) coupon\_reduce\_amount,

sum(order\_total\_amount\_1d) total\_amount

from ${APP}.dws\_trade\_user\_order\_1d

group by user\_id;

"

dws\_trade\_user\_payment\_td="

insert overwrite table ${APP}.dws\_trade\_user\_payment\_td partition(dt='$do\_date')

select

user\_id,

min(dt) payment\_date\_first,

max(dt) payment\_date\_last,

sum(payment\_count\_1d) payment\_count,

sum(payment\_num\_1d) payment\_num,

sum(payment\_amount\_1d) payment\_amount

from ${APP}.dws\_trade\_user\_payment\_1d

group by user\_id;

"

dws\_user\_user\_login\_td="

insert overwrite table ${APP}.dws\_user\_user\_login\_td partition(dt='$do\_date')

select

u.id,

nvl(login\_date\_last,date\_format(create\_time,'yyyy-MM-dd')),

nvl(login\_count\_td,1)

from

(

select

id,

create\_time

from ${APP}.dim\_user\_zip

where dt='9999-12-31'

)u

left join

(

select

user\_id,

max(dt) login\_date\_last,

count(\*) login\_count\_td

from ${APP}.dwd\_user\_login\_inc

group by user\_id

)l

on u.id=l.user\_id;

"

case $1 in

"dws\_trade\_user\_order\_td" )

hive -e "$dws\_trade\_user\_order\_td"

;;

"dws\_trade\_user\_payment\_td" )

hive -e "$dws\_trade\_user\_payment\_td"

;;

"dws\_user\_user\_login\_td" )

hive -e "$dws\_user\_user\_login\_td"

;;

"all" )

hive -e "$dws\_trade\_user\_order\_td$dws\_trade\_user\_payment\_td$dws\_user\_user\_login\_td"

;;

esac

（3）增加脚本执行权限

[atguigu@hadoop102 bin]$ chmod +x dws\_1d\_to\_dws\_td\_init.sh

（4）脚本用法

[atguigu@hadoop102 bin]$ dws\_1d\_to\_dws\_td\_init.sh all 2020-06-14

**2）每日数据装载脚本**

（1）在hadoop102的/home/atguigu/bin目录下创建dws\_1d\_to\_dws\_td.sh

[atguigu@hadoop102 bin]$ vim dws\_1d\_to\_dws\_td.sh

（2）编写如下内容

#!/bin/bash

APP=gmall

# 如果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

dws\_trade\_user\_order\_td="

insert overwrite table ${APP}.dws\_trade\_user\_order\_td partition(dt='$do\_date')

select

nvl(old.user\_id,new.user\_id),

if(new.user\_id is not null and old.user\_id is null,'$do\_date',old.order\_date\_first),

if(new.user\_id is not null,'$do\_date',old.order\_date\_last),

nvl(old.order\_count\_td,0)+nvl(new.order\_count\_1d,0),

nvl(old.order\_num\_td,0)+nvl(new.order\_num\_1d,0),

nvl(old.original\_amount\_td,0)+nvl(new.order\_original\_amount\_1d,0),

nvl(old.activity\_reduce\_amount\_td,0)+nvl(new.activity\_reduce\_amount\_1d,0),

nvl(old.coupon\_reduce\_amount\_td,0)+nvl(new.coupon\_reduce\_amount\_1d,0),

nvl(old.total\_amount\_td,0)+nvl(new.order\_total\_amount\_1d,0)

from

(

select

user\_id,

order\_date\_first,

order\_date\_last,

order\_count\_td,

order\_num\_td,

original\_amount\_td,

activity\_reduce\_amount\_td,

coupon\_reduce\_amount\_td,

total\_amount\_td

from ${APP}.dws\_trade\_user\_order\_td

where dt=date\_add('$do\_date',-1)

)old

full outer join

(

select

user\_id,

order\_count\_1d,

order\_num\_1d,

order\_original\_amount\_1d,

activity\_reduce\_amount\_1d,

coupon\_reduce\_amount\_1d,

order\_total\_amount\_1d

from ${APP}.dws\_trade\_user\_order\_1d

where dt='$do\_date'

)new

on old.user\_id=new.user\_id;

"

dws\_trade\_user\_payment\_td="

insert overwrite table ${APP}.dws\_trade\_user\_payment\_td partition(dt='$do\_date')

select

nvl(old.user\_id,new.user\_id),

if(old.user\_id is null and new.user\_id is not null,'$do\_date',old.payment\_date\_first),

if(new.user\_id is not null,'$do\_date',old.payment\_date\_last),

nvl(old.payment\_count\_td,0)+nvl(new.payment\_count\_1d,0),

nvl(old.payment\_num\_td,0)+nvl(new.payment\_num\_1d,0),

nvl(old.payment\_amount\_td,0)+nvl(new.payment\_amount\_1d,0)

from

(

select

user\_id,

payment\_date\_first,

payment\_date\_last,

payment\_count\_td,

payment\_num\_td,

payment\_amount\_td

from ${APP}.dws\_trade\_user\_payment\_td

where dt=date\_add('$do\_date',-1)

)old

full outer join

(

select

user\_id,

payment\_count\_1d,

payment\_num\_1d,

payment\_amount\_1d

from ${APP}.dws\_trade\_user\_payment\_1d

where dt='$do\_date'

)new

on old.user\_id=new.user\_id;

"

dws\_user\_user\_login\_td="

insert overwrite table ${APP}.dws\_user\_user\_login\_td partition(dt='$do\_date')

select

nvl(old.user\_id,new.user\_id),

if(new.user\_id is null,old.login\_date\_last,'$do\_date'),

nvl(old.login\_count\_td,0)+nvl(new.login\_count\_1d,0)

from

(

select

user\_id,

login\_date\_last,

login\_count\_td

from ${APP}.dws\_user\_user\_login\_td

where dt=date\_add('$do\_date',-1)

)old

full outer join

(

select

user\_id,

count(\*) login\_count\_1d

from ${APP}.dwd\_user\_login\_inc

where dt='$do\_date'

group by user\_id

)new

on old.user\_id=new.user\_id;

"

case $1 in

"dws\_trade\_user\_order\_td" )

hive -e "$dws\_trade\_user\_order\_td"

;;

"dws\_trade\_user\_payment\_td" )

hive -e "$dws\_trade\_user\_payment\_td"

;;

"dws\_user\_user\_login\_td" )

hive -e "$dws\_user\_user\_login\_td"

;;

"all" )

hive -e "$dws\_trade\_user\_order\_td$dws\_trade\_user\_payment\_td$dws\_user\_user\_login\_td"

;;

esac

（3）增加脚本执行权限

[atguigu@hadoop102 bin]$ chmod +x dws\_1d\_to\_dws\_td.sh

（4）脚本用法

[atguigu@hadoop102 bin]$ dws\_1d\_to\_dws\_td.sh all 2020-06-14

# 第11章 数仓开发之ADS层

## 11.1 流量主题

### 11.1.1 各渠道流量统计

需求说明如下。

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | 指标 | 说明 |
| 最近1/7/30日 | 渠道 | 访客数 | 统计访问人数 |
| 最近1/7/30日 | 渠道 | 会话平均停留时长 | 统计每个会话平均停留时长 |
| 最近1/7/30日 | 渠道 | 会话平均浏览页面数 | 统计每个会话平均浏览页面数 |
| 最近1/7/30日 | 渠道 | 会话总数 | 统计会话总数 |
| 最近1/7/30日 | 渠道 | 跳出率 | 只有一个页面的会话的比例 |

**1）建表语句**

DROP TABLE IF EXISTS ads\_traffic\_stats\_by\_channel;

CREATE EXTERNAL TABLE ads\_traffic\_stats\_by\_channel

(

`dt` STRING COMMENT '统计日期',

`recent\_days` BIGINT COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

`channel` STRING COMMENT '渠道',

`uv\_count` BIGINT COMMENT '访客人数',

`avg\_duration\_sec` BIGINT COMMENT '会话平均停留时长，单位为秒',

`avg\_page\_count` BIGINT COMMENT '会话平均浏览页面数',

`sv\_count` BIGINT COMMENT '会话数',

`bounce\_rate` DECIMAL(16, 2) COMMENT '跳出率'

) COMMENT '各渠道流量统计'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_traffic\_stats\_by\_channel/';

**2）数据装载**

insert overwrite table ads\_traffic\_stats\_by\_channel

select \* from ads\_traffic\_stats\_by\_channel

union

select

'2020-06-14' dt,

recent\_days,

channel,

cast(count(distinct(mid\_id)) as bigint) uv\_count,

cast(avg(during\_time\_1d)/1000 as bigint) avg\_duration\_sec,

cast(avg(page\_count\_1d) as bigint) avg\_page\_count,

cast(count(\*) as bigint) sv\_count,

cast(sum(if(page\_count\_1d=1,1,0))/count(\*) as decimal(16,2)) bounce\_rate

from dws\_traffic\_session\_page\_view\_1d lateral view explode(array(1,7,30)) tmp as recent\_days

where dt>=date\_add('2020-06-14',-recent\_days+1)

group by recent\_days,channel;

### 11.1.2 路径分析

用户路径分析，顾名思义，就是指用户在APP或网站中的访问路径。为了衡量网站优化的效果或营销推广的效果，以及了解用户行为偏好，时常要对访问路径进行分析。

用户访问路径的可视化通常使用桑基图。如下图所示，该图可真实还原用户的访问路径，包括页面跳转和页面访问次序。

桑基图需要我们提供每种页面跳转的次数，每个跳转由source/target表示，source指跳转起始页面，target表示跳转终到页面。

图形用户界面, 应用程序

描述已自动生成

**1）建表语句**

DROP TABLE IF EXISTS ads\_page\_path;

CREATE EXTERNAL TABLE ads\_page\_path

(

`dt` STRING COMMENT '统计日期',

`recent\_days` BIGINT COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

`source` STRING COMMENT '跳转起始页面ID',

`target` STRING COMMENT '跳转终到页面ID',

`path\_count` BIGINT COMMENT '跳转次数'

) COMMENT '页面浏览路径分析'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_page\_path/';

**2）数据装载**

insert overwrite table ads\_page\_path

select \* from ads\_page\_path

union

select

'2020-06-14' dt,

recent\_days,

source,

nvl(target,'null'),

count(\*) path\_count

from

(

select

recent\_days,

concat('step-',rn,':',page\_id) source,

concat('step-',rn+1,':',next\_page\_id) target

from

(

select

recent\_days,

page\_id,

lead(page\_id,1,null) over(partition by session\_id,recent\_days) next\_page\_id,

row\_number() over (partition by session\_id,recent\_days order by view\_time) rn

from dwd\_traffic\_page\_view\_inc lateral view explode(array(1,7,30)) tmp as recent\_days

where dt>=date\_add('2020-06-14',-recent\_days+1)

)t1

)t2

group by recent\_days,source,target;

## 11.2 用户主题

### 11.2.1 用户变动统计

该需求包括两个指标，分别为流失用户数和回流用户数，以下为对两个指标的解释说明。

|  |  |  |
| --- | --- | --- |
| **统计周期** | **指标** | **说明** |
| **最近1日** | 流失用户数 | 之前活跃过的用户，最近一段时间未活跃，就称为流失用户。此处要求统计7日前（只包含7日前当天）活跃，但最近7日未活跃的用户总数。 |
| **最近1日** | 回流用户数 | 之前的活跃用户，一段时间未活跃（流失），今日又活跃了，就称为回流用户。此处要求统计回流用户总数。 |

**1）建表语句**

DROP TABLE IF EXISTS ads\_user\_change;

CREATE EXTERNAL TABLE ads\_user\_change

(

`dt` STRING COMMENT '统计日期',

`user\_churn\_count` BIGINT COMMENT '流失用户数',

`user\_back\_count` BIGINT COMMENT '回流用户数'

) COMMENT '用户变动统计'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_user\_change/';

**2）数据装载**

insert overwrite table ads\_user\_change

select \* from ads\_user\_change

union

select

churn.dt,

user\_churn\_count,

user\_back\_count

from

(

select

'2020-06-14' dt,

count(\*) user\_churn\_count

from dws\_user\_user\_login\_td

where dt='2020-06-14'

and login\_date\_last=date\_add('2020-06-14',-7)

)churn

join

(

select

'2020-06-14' dt,

count(\*) user\_back\_count

from

(

select

user\_id,

login\_date\_last

from dws\_user\_user\_login\_td

where dt='2020-06-14'

)t1

join

(

select

user\_id,

login\_date\_last login\_date\_previous

from dws\_user\_user\_login\_td

where dt=date\_add('2020-06-14',-1)

)t2

on t1.user\_id=t2.user\_id

where datediff(login\_date\_last,login\_date\_previous)>=8

)back

on churn.dt=back.dt;

### 11.2.2 用户留存率

留存分析一般包含新增留存和活跃留存分析。

新增留存分析是分析某天的新增用户中，有多少人有后续的活跃行为。活跃留存分析是分析某天的活跃用户中，有多少人有后续的活跃行为。

留存分析是衡量产品对用户价值高低的重要指标。

此处要求统计新增留存率，新增留存率具体是指留存用户数与新增用户数的比值，例如2020-06-14新增100个用户，1日之后（2020-06-15）这100人中有80个人活跃了，那2020-06-14的1日留存数则为80，2020-06-14的1日留存率则为80%。

要求统计每天的1至7日留存率，如下图所示。

图形用户界面, 应用程序

描述已自动生成

**1）建表语句**

DROP TABLE IF EXISTS ads\_user\_retention;

CREATE EXTERNAL TABLE ads\_user\_retention

(

`dt` STRING COMMENT '统计日期',

`create\_date` STRING COMMENT '用户新增日期',

`retention\_day` INT COMMENT '截至当前日期留存天数',

`retention\_count` BIGINT COMMENT '留存用户数量',

`new\_user\_count` BIGINT COMMENT '新增用户数量',

`retention\_rate` DECIMAL(16, 2) COMMENT '留存率'

) COMMENT '用户留存率'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_user\_retention/';

**2）数据装载**

insert overwrite table ads\_user\_retention

select \* from ads\_user\_retention

union

select

'2020-06-14' dt,

login\_date\_first create\_date,

datediff('2020-06-14',login\_date\_first) retention\_day,

sum(if(login\_date\_last='2020-06-14',1,0)) retention\_count,

count(\*) new\_user\_count,

cast(sum(if(login\_date\_last='2020-06-14',1,0))/count(\*)\*100 as decimal(16,2)) retention\_rate

from

(

select

user\_id,

date\_id login\_date\_first

from dwd\_user\_register\_inc

where dt>=date\_add('2020-06-14',-7)

and dt<'2020-06-14'

)t1

join

(

select

user\_id,

login\_date\_last

from dws\_user\_user\_login\_td

where dt='2020-06-14'

)t2

on t1.user\_id=t2.user\_id

group by login\_date\_first;

### 11.2.3 用户新增活跃统计

需求说明如下

|  |  |  |
| --- | --- | --- |
| **统计周期** | **指标** | **指标说明** |
| 最近1、7、30日 | 新增用户数 | 略 |
| 最近1、7、30日 | 活跃用户数 | 略 |

**1）建表语句**

DROP TABLE IF EXISTS ads\_user\_stats;

CREATE EXTERNAL TABLE ads\_user\_stats

(

`dt` STRING COMMENT '统计日期',

`recent\_days` BIGINT COMMENT '最近n日,1:最近1日,7:最近7日,30:最近30日',

`new\_user\_count` BIGINT COMMENT '新增用户数',

`active\_user\_count` BIGINT COMMENT '活跃用户数'

) COMMENT '用户新增活跃统计'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_user\_stats/';

**2）数据装载**

insert overwrite table ads\_user\_stats

select \* from ads\_user\_stats

union

select

'2020-06-14' dt,

t1.recent\_days,

new\_user\_count,

active\_user\_count

from

(

select

recent\_days,

sum(if(login\_date\_last>=date\_add('2020-06-14',-recent\_days+1),1,0)) new\_user\_count

from dws\_user\_user\_login\_td lateral view explode(array(1,7,30)) tmp as recent\_days

where dt='2020-06-14'

group by recent\_days

)t1

join

(

select

recent\_days,

sum(if(date\_id>=date\_add('2020-06-14',-recent\_days+1),1,0)) active\_user\_count

from dwd\_user\_register\_inc lateral view explode(array(1,7,30)) tmp as recent\_days

group by recent\_days

)t2

on t1.recent\_days=t2.recent\_days;

### 11.2.4 用户行为漏斗分析

漏斗分析是一个数据分析模型，它能够科学反映一个业务过程从起点到终点各阶段用户转化情况。由于其能将各阶段环节都展示出来，故哪个阶段存在问题，就能一目了然。

图表, 漏斗图

描述已自动生成

该需求要求统计一个完整的购物流程各个阶段的人数，具体说明如下：

|  |  |  |
| --- | --- | --- |
| **统计周期** | **指标** | **说明** |
| 最近1、7、30日 | 首页浏览人数 | 略 |
| 最近1、7、30日 | 商品详情页浏览人数 | 略 |
| 最近1、7、30日 | 加购人数 | 略 |
| 最近1、7、30日 | 下单人数 | 略 |
| 最近1、7、30日 | 支付人数 | 支付成功人数 |

**1）建表语句**

DROP TABLE IF EXISTS ads\_user\_action;

CREATE EXTERNAL TABLE ads\_user\_action

(

`dt` STRING COMMENT '统计日期',

`recent\_days` BIGINT COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

`home\_count` BIGINT COMMENT '浏览首页人数',

`good\_detail\_count` BIGINT COMMENT '浏览商品详情页人数',

`cart\_count` BIGINT COMMENT '加入购物车人数',

`order\_count` BIGINT COMMENT '下单人数',

`payment\_count` BIGINT COMMENT '支付人数'

) COMMENT '漏斗分析'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_user\_action/';

**2）数据装载**

insert overwrite table ads\_user\_action

select \* from ads\_user\_action

union

select

'2020-06-14' dt,

page.recent\_days,

home\_count,

good\_detail\_count,

cart\_count,

order\_count,

payment\_count

from

(

select

1 recent\_days,

sum(if(page\_id='home',1,0)) home\_count,

sum(if(page\_id='good\_detail',1,0)) good\_detail\_count

from dws\_traffic\_page\_visitor\_page\_view\_1d

where dt='2020-06-14'

and page\_id in ('home','good\_detail')

union all

select

recent\_days,

sum(if(page\_id='home' and view\_count>0,1,0)),

sum(if(page\_id='good\_detail' and view\_count>0,1,0))

from

(

select

recent\_days,

page\_id,

case recent\_days

when 7 then view\_count\_7d

when 30 then view\_count\_30d

end view\_count

from dws\_traffic\_page\_visitor\_page\_view\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='2020-06-14'

and page\_id in ('home','good\_detail')

)t1

group by recent\_days

)page

join

(

select

1 recent\_days,

count(\*) cart\_count

from dws\_trade\_user\_cart\_add\_1d

where dt='2020-06-14'

union all

select

recent\_days,

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

from

(

select

recent\_days,

case recent\_days

when 7 then cart\_add\_count\_7d

when 30 then cart\_add\_count\_30d

end cart\_count

from dws\_trade\_user\_cart\_add\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='2020-06-14'

)t1

group by recent\_days

)cart

on page.recent\_days=cart.recent\_days

join

(

select

1 recent\_days,

count(\*) order\_count

from dws\_trade\_user\_order\_1d

where dt='2020-06-14'

union all

select

recent\_days,

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

from

(

select

recent\_days,

case recent\_days

when 7 then order\_count\_7d

when 30 then order\_count\_30d

end order\_count

from dws\_trade\_user\_order\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='2020-06-14'

)t1

group by recent\_days

)ord

on page.recent\_days=ord.recent\_days

join

(

select

1 recent\_days,

count(\*) payment\_count

from dws\_trade\_user\_payment\_1d

where dt='2020-06-14'

union all

select

recent\_days,

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

from

(

select

recent\_days,

case recent\_days

when 7 then payment\_count\_7d

when 30 then payment\_count\_30d

end order\_count

from dws\_trade\_user\_payment\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='2020-06-14'

)t1

group by recent\_days

)pay

on page.recent\_days=pay.recent\_days;

### 11.2.5 新增交易用户统计

需求说明如下。

|  |  |  |
| --- | --- | --- |
| **统计周期** | **指标** | **说明** |
| 最近1、7、30日 | 新增下单人数 | 略 |
| 最近1、7、30日 | 新增支付人数 | 略 |

**1）建表语句**

DROP TABLE IF EXISTS ads\_new\_buyer\_stats;

CREATE EXTERNAL TABLE ads\_new\_buyer\_stats

(

`dt` STRING COMMENT '统计日期',

`recent\_days` BIGINT COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

`new\_order\_user\_count` BIGINT COMMENT '新增下单人数',

`new\_payment\_user\_count` BIGINT COMMENT '新增支付人数'

) COMMENT '新增交易用户统计'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_new\_buyer\_stats/';

**2）数据装载**

insert overwrite table ads\_new\_buyer\_stats

select \* from ads\_new\_buyer\_stats

union

select

'2020-06-14',

odr.recent\_days,

new\_order\_user\_count,

new\_payment\_user\_count

from

(

select

recent\_days,

sum(if(order\_date\_first>=date\_add('2020-06-14',-recent\_days+1),1,0)) new\_order\_user\_count

from dws\_trade\_user\_order\_td lateral view explode(array(1,7,30)) tmp as recent\_days

where dt='2020-06-14'

group by recent\_days

)odr

join

(

select

recent\_days,

sum(if(payment\_date\_first>=date\_add('2020-06-14',-recent\_days+1),1,0)) new\_payment\_user\_count

from dws\_trade\_user\_payment\_td lateral view explode(array(1,7,30)) tmp as recent\_days

where dt='2020-06-14'

group by recent\_days

)pay

on odr.recent\_days=pay.recent\_days;

## 11.3 商品主题

### 11.3.1 最近7/30日各品牌复购率

需求说明如下。

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | **指标** | **说明** |
| 最近7、30日 | 品牌 | 复购率 | 重复购买人数占购买人数比例 |

**1）建表语句**

DROP TABLE IF EXISTS ads\_repeat\_purchase\_by\_tm;

CREATE EXTERNAL TABLE ads\_repeat\_purchase\_by\_tm

(

`dt` STRING COMMENT '统计日期',

`recent\_days` BIGINT COMMENT '最近天数,7:最近7天,30:最近30天',

`tm\_id` STRING COMMENT '品牌ID',

`tm\_name` STRING COMMENT '品牌名称',

`order\_repeat\_rate` DECIMAL(16, 2) COMMENT '复购率'

) COMMENT '各品牌复购率统计'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_repeat\_purchase\_by\_tm/';

**2）数据装载**

insert overwrite table ads\_repeat\_purchase\_by\_tm

select \* from ads\_repeat\_purchase\_by\_tm

union

select

'2020-06-14' dt,

recent\_days,

tm\_id,

tm\_name,

cast(sum(if(order\_count>=2,1,0))/sum(if(order\_count>=1,1,0)) as decimal(16,2))

from

(

select

'2020-06-14' dt,

recent\_days,

user\_id,

tm\_id,

tm\_name,

sum(order\_count) order\_count

from

(

select

recent\_days,

user\_id,

tm\_id,

tm\_name,

case recent\_days

when 7 then order\_count\_7d

when 30 then order\_count\_30d

end order\_count

from dws\_trade\_user\_sku\_order\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='2020-06-14'

)t1

group by recent\_days,user\_id,tm\_id,tm\_name

)t2

group by recent\_days,tm\_id,tm\_name;

### 11.3.2 各品牌商品交易统计

需求说明如下。

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | **指标** | **说明** |
| 最近1、7、30日 | 品牌 | 订单数 | 略 |
| 最近1、7、30日 | 品牌 | 订单人数 | 略 |
| 最近1、7、30日 | 品牌 | 退单数 | 略 |
| 最近1、7、30日 | 品牌 | 退单人数 | 略 |

**1）建表语句**

DROP TABLE IF EXISTS ads\_trade\_stats\_by\_tm;

CREATE EXTERNAL TABLE ads\_trade\_stats\_by\_tm

(

`dt` STRING COMMENT '统计日期',

`recent\_days` BIGINT COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

`tm\_id` STRING COMMENT '品牌ID',

`tm\_name` STRING COMMENT '品牌名称',

`order\_count` BIGINT COMMENT '订单数',

`order\_user\_count` BIGINT COMMENT '订单人数',

`order\_refund\_count` BIGINT COMMENT '退单数',

`order\_refund\_user\_count` BIGINT COMMENT '退单人数'

) COMMENT '各品牌商品交易统计'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_trade\_stats\_by\_tm/';

**2）数据装载**

insert overwrite table ads\_trade\_stats\_by\_tm

select \* from ads\_trade\_stats\_by\_tm

union

select

'2020-06-14' dt,

nvl(odr.recent\_days,refund.recent\_days),

nvl(odr.tm\_id,refund.tm\_id),

nvl(odr.tm\_name,refund.tm\_name),

nvl(order\_count,0),

nvl(order\_user\_count,0),

nvl(order\_refund\_count,0),

nvl(order\_refund\_user\_count,0)

from

(

select

1 recent\_days,

tm\_id,

tm\_name,

sum(order\_count\_1d) order\_count,

count(distinct(user\_id)) order\_user\_count

from dws\_trade\_user\_sku\_order\_1d

where dt='2020-06-14'

group by tm\_id,tm\_name

union all

select

recent\_days,

tm\_id,

tm\_name,

sum(order\_count),

count(distinct(if(order\_count>0,user\_id,null)))

from

(

select

recent\_days,

user\_id,

tm\_id,

tm\_name,

case recent\_days

when 7 then order\_count\_7d

when 30 then order\_count\_30d

end order\_count

from dws\_trade\_user\_sku\_order\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='2020-06-14'

)t1

group by recent\_days,tm\_id,tm\_name

)odr

full outer join

(

select

1 recent\_days,

tm\_id,

tm\_name,

sum(order\_refund\_count\_1d) order\_refund\_count,

count(distinct(user\_id)) order\_refund\_user\_count

from dws\_trade\_user\_sku\_order\_refund\_1d

where dt='2020-06-14'

group by tm\_id,tm\_name

union all

select

recent\_days,

tm\_id,

tm\_name,

sum(order\_refund\_count),

count(if(order\_refund\_count>0,user\_id,null))

from

(

select

recent\_days,

user\_id,

tm\_id,

tm\_name,

case recent\_days

when 7 then order\_refund\_count\_7d

when 30 then order\_refund\_count\_30d

end order\_refund\_count

from dws\_trade\_user\_sku\_order\_refund\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='2020-06-14'

)t1

group by recent\_days,tm\_id,tm\_name

)refund

on odr.recent\_days=refund.recent\_days

and odr.tm\_id=refund.tm\_id

and odr.tm\_name=refund.tm\_name;

### 11.3.3 各品类商品交易统计

需求说明如下。

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | **指标** | **说明** |
| 最近1、7、30日 | 品类 | 订单数 | 略 |
| 最近1、7、30日 | 品类 | 订单人数 | 略 |
| 最近1、7、30日 | 品类 | 退单数 | 略 |
| 最近1、7、30日 | 品类 | 退单人数 | 略 |

**1）建表语句**

DROP TABLE IF EXISTS ads\_trade\_stats\_by\_cate;

CREATE EXTERNAL TABLE ads\_trade\_stats\_by\_cate

(

`dt` STRING COMMENT '统计日期',

`recent\_days` BIGINT COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

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

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

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

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

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

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

`order\_count` BIGINT COMMENT '订单数',

`order\_user\_count` BIGINT COMMENT '订单人数',

`order\_refund\_count` BIGINT COMMENT '退单数',

`order\_refund\_user\_count` BIGINT COMMENT '退单人数'

) COMMENT '各分类商品交易统计'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_trade\_stats\_by\_cate/';

**2）数据装载**

insert overwrite table ads\_trade\_stats\_by\_cate

select \* from ads\_trade\_stats\_by\_cate

union

select

'2020-06-14' dt,

nvl(odr.recent\_days,refund.recent\_days),

nvl(odr.category1\_id,refund.category1\_id),

nvl(odr.category1\_name,refund.category1\_name),

nvl(odr.category2\_id,refund.category2\_id),

nvl(odr.category2\_name,refund.category2\_name),

nvl(odr.category3\_id,refund.category3\_id),

nvl(odr.category3\_name,refund.category3\_name),

nvl(order\_count,0),

nvl(order\_user\_count,0),

nvl(order\_refund\_count,0),

nvl(order\_refund\_user\_count,0)

from

(

select

1 recent\_days,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

sum(order\_count\_1d) order\_count,

count(distinct(user\_id)) order\_user\_count

from dws\_trade\_user\_sku\_order\_1d

where dt='2020-06-14'

group by category1\_id,category1\_name,category2\_id,category2\_name,category3\_id,category3\_name

union all

select

recent\_days,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

sum(order\_count),

count(distinct(if(order\_count>0,user\_id,null)))

from

(

select

recent\_days,

user\_id,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

case recent\_days

when 7 then order\_count\_7d

when 30 then order\_count\_30d

end order\_count

from dws\_trade\_user\_sku\_order\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='2020-06-14'

)t1

group by recent\_days,category1\_id,category1\_name,category2\_id,category2\_name,category3\_id,category3\_name

)odr

full outer join

(

select

1 recent\_days,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

sum(order\_refund\_count\_1d) order\_refund\_count,

count(distinct(user\_id)) order\_refund\_user\_count

from dws\_trade\_user\_sku\_order\_refund\_1d

where dt='2020-06-14'

group by category1\_id,category1\_name,category2\_id,category2\_name,category3\_id,category3\_name

union all

select

recent\_days,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

sum(order\_refund\_count),

count(distinct(if(order\_refund\_count>0,user\_id,null)))

from

(

select

recent\_days,

user\_id,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

case recent\_days

when 7 then order\_refund\_count\_7d

when 30 then order\_refund\_count\_30d

end order\_refund\_count

from dws\_trade\_user\_sku\_order\_refund\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='2020-06-14'

)t1

group by recent\_days,category1\_id,category1\_name,category2\_id,category2\_name,category3\_id,category3\_name

)refund

on odr.recent\_days=refund.recent\_days

and odr.category1\_id=refund.category1\_id

and odr.category1\_name=refund.category1\_name

and odr.category2\_id=refund.category2\_id

and odr.category2\_name=refund.category2\_name

and odr.category3\_id=refund.category3\_id

and odr.category3\_name=refund.category3\_name;

### 11.3.4 各分类商品购物车存量Top10

**1）建表语句**

DROP TABLE IF EXISTS ads\_sku\_cart\_num\_top3\_by\_cate;

CREATE EXTERNAL TABLE ads\_sku\_cart\_num\_top3\_by\_cate

(

`dt` STRING COMMENT '统计日期',

`category1\_id` STRING COMMENT '一级分类ID',

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

`category2\_id` STRING COMMENT '二级分类ID',

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

`category3\_id` STRING COMMENT '三级分类ID',

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

`sku\_id` STRING COMMENT '商品id',

`sku\_name` STRING COMMENT '商品名称',

`cart\_num` BIGINT COMMENT '购物车中商品数量',

`rk` BIGINT COMMENT '排名'

) COMMENT '各分类商品购物车存量Top10'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_sku\_cart\_num\_top3\_by\_cate/';

**2）数据装载**

insert overwrite table ads\_sku\_cart\_num\_top3\_by\_cate

select \* from ads\_sku\_cart\_num\_top3\_by\_cate

union

select

'2020-06-14' dt,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

sku\_id,

sku\_name,

cart\_num,

rk

from

(

select

sku\_id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

cart\_num,

rank() over (partition by category1\_id,category2\_id,category3\_id order by cart\_num desc) rk

from

(

select

sku\_id,

sum(sku\_num) cart\_num

from dwd\_trade\_cart\_full

where dt='2020-06-14'

group by sku\_id

)cart

left join

(

select

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name

from dim\_sku\_full

where dt='2020-06-14'

)sku

on cart.sku\_id=sku.id

)t1

where rk<=3;

## 11.4 交易主题

### 11.4.1 交易综合统计

需求说明如下

|  |  |  |
| --- | --- | --- |
| **统计周期** | **指标** | **说明** |
| 最近1、7、30日 | 订单总额 | 订单最终金额 |
| 最近1、7、30日 | 订单数 | 略 |
| 最近1、7、30日 | 订单人数 | 略 |
| 最近1、7、30日 | 退单数 | 略 |
| 最近1、7、30日 | 退单人数 | 略 |

**1）建表语句**

DROP TABLE IF EXISTS ads\_trade\_stats;

CREATE EXTERNAL TABLE ads\_trade\_stats

(

`dt` STRING COMMENT '统计日期',

`recent\_days` BIGINT COMMENT '最近天数,1:最近1日,7:最近7天,30:最近30天',

`order\_total\_amount` DECIMAL(16, 2) COMMENT '订单总额,GMV',

`order\_count` BIGINT COMMENT '订单数',

`order\_user\_count` BIGINT COMMENT '下单人数',

`order\_refund\_count` BIGINT COMMENT '退单数',

`order\_refund\_user\_count` BIGINT COMMENT '退单人数'

) COMMENT '交易统计'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_trade\_stats/';

**2）数据装载**

insert overwrite table ads\_trade\_stats

select \* from ads\_trade\_stats

union

select

'2020-06-14',

odr.recent\_days,

order\_total\_amount,

order\_count,

order\_user\_count,

order\_refund\_count,

order\_refund\_user\_count

from

(

select

1 recent\_days,

sum(order\_total\_amount\_1d) order\_total\_amount,

sum(order\_count\_1d) order\_count,

count(\*) order\_user\_count

from dws\_trade\_user\_order\_1d

where dt='2020-06-14'

union all

select

recent\_days,

sum(order\_total\_amount),

sum(order\_count),

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

from

(

select

recent\_days,

case recent\_days

when 7 then order\_total\_amount\_7d

when 30 then order\_total\_amount\_30d

end order\_total\_amount,

case recent\_days

when 7 then order\_count\_7d

when 30 then order\_count\_30d

end order\_count

from dws\_trade\_user\_order\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='2020-06-14'

)t1

group by recent\_days

)odr

join

(

select

1 recent\_days,

sum(order\_refund\_count\_1d) order\_refund\_count,

count(\*) order\_refund\_user\_count

from dws\_trade\_user\_order\_refund\_1d

where dt='2020-06-14'

union all

select

recent\_days,

sum(order\_refund\_count),

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

from

(

select

recent\_days,

case recent\_days

when 7 then order\_refund\_count\_7d

when 30 then order\_refund\_count\_30d

end order\_refund\_count

from dws\_trade\_user\_order\_refund\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='2020-06-14'

)t1

group by recent\_days

)refund

on odr.recent\_days=refund.recent\_days;

### 11.4.2 各省份交易统计

需求说明如下。

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | **指标** | **说明** |
| 最近1、7、30日 | 省份 | 订单数 | 略 |
| 最近1、7、30日 | 省份 | 订单金额 | 略 |

**1）建表语句**

DROP TABLE IF EXISTS ads\_order\_by\_province;

CREATE EXTERNAL TABLE ads\_order\_by\_province

(

`dt` STRING COMMENT '统计日期',

`recent\_days` BIGINT COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

`province\_id` STRING COMMENT '省份ID',

`province\_name` STRING COMMENT '省份名称',

`area\_code` STRING COMMENT '地区编码',

`iso\_code` STRING COMMENT '国际标准地区编码',

`iso\_code\_3166\_2` STRING COMMENT '国际标准地区编码',

`order\_count` BIGINT COMMENT '订单数',

`order\_total\_amount` DECIMAL(16, 2) COMMENT '订单金额'

) COMMENT '各地区订单统计'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_order\_by\_province/';

**2）数据装载**

insert overwrite table ads\_order\_by\_province

select \* from ads\_order\_by\_province

union

select

'2020-06-14' dt,

1 recent\_days,

province\_id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2,

order\_count\_1d,

order\_total\_amount\_1d

from dws\_trade\_province\_order\_1d

where dt='2020-06-14'

union

select

'2020-06-14' dt,

recent\_days,

province\_id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2,

sum(order\_count),

sum(order\_total\_amount)

from

(

select

recent\_days,

province\_id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2,

case recent\_days

when 7 then order\_count\_7d

when 30 then order\_count\_30d

end order\_count,

case recent\_days

when 7 then order\_total\_amount\_7d

when 30 then order\_total\_amount\_30d

end order\_total\_amount

from dws\_trade\_province\_order\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='2020-06-14'

)t1

group by recent\_days,province\_id,province\_name,area\_code,iso\_code,iso\_3166\_2;

## 11.5 优惠券主题

### 11.5.1 最近30天发布的优惠券的补贴率

需求说明如下

|  |  |  |
| --- | --- | --- |
| **统计粒度** | **指标** | **说明** |
| **优惠券** | 补贴率 | 用券的订单明细优惠券减免金额总和/原始金额总和 |

**1）建表语句**

DROP TABLE IF EXISTS ads\_coupon\_stats;

CREATE EXTERNAL TABLE ads\_coupon\_stats

(

`dt` STRING COMMENT '统计日期',

`coupon\_id` STRING COMMENT '优惠券ID',

`coupon\_name` STRING COMMENT '优惠券名称',

`start\_date` STRING COMMENT '发布日期',

`rule\_name` STRING COMMENT '优惠规则，例如满100元减10元',

`reduce\_rate` DECIMAL(16, 2) COMMENT '补贴率'

) COMMENT '优惠券统计'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_coupon\_stats/';

**2）数据装载**

insert overwrite table ads\_coupon\_stats

select \* from ads\_coupon\_stats

union

select

'2020-06-14' dt,

coupon\_id,

coupon\_name,

start\_date,

coupon\_rule,

cast(coupon\_reduce\_amount\_30d/original\_amount\_30d as decimal(16,2))

from dws\_trade\_coupon\_order\_nd

where dt='2020-06-14';

## 11.6 活动主题

### 11.6.1 最近30天发布的活动的补贴率

需求说明如下

|  |  |  |
| --- | --- | --- |
| **统计粒度** | **指标** | **说明** |
| **活动** | 补贴率 | 参与促销活动的订单明细活动减免金额总和/原始金额总和 |

**1）建表语句**

DROP TABLE IF EXISTS ads\_activity\_stats;

CREATE EXTERNAL TABLE ads\_activity\_stats

(

`dt` STRING COMMENT '统计日期',

`activity\_id` STRING COMMENT '活动ID',

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

`start\_date` STRING COMMENT '活动开始日期',

`reduce\_rate` DECIMAL(16, 2) COMMENT '补贴率'

) COMMENT '活动统计'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LOCATION '/warehouse/gmall/ads/ads\_activity\_stats/';

**2）数据装载**

insert overwrite table ads\_activity\_stats

select \* from ads\_activity\_stats

union

select

'2020-06-14' dt,

activity\_id,

activity\_name,

start\_date,

cast(activity\_reduce\_amount\_30d/original\_amount\_30d as decimal(16,2))

from dws\_trade\_activity\_order\_nd

where dt='2020-06-14';

## 11.7 数据装载脚本

**1）每日数据装载脚本**

（1）在hadoop102的/home/atguigu/bin目录下创建dws\_to\_ads.sh

[atguigu@hadoop102 bin]$ vim dws\_to\_ads.sh

（2）编写如下内容

#!/bin/bash

APP=gmall

# 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

ads\_activity\_stats="

insert overwrite table ${APP}.ads\_activity\_stats

select \* from ${APP}.ads\_activity\_stats

union

select

'$do\_date' dt,

activity\_id,

activity\_name,

start\_date,

cast(activity\_reduce\_amount\_30d/original\_amount\_30d as decimal(16,2))

from ${APP}.dws\_trade\_activity\_order\_nd

where dt='$do\_date';

"

ads\_coupon\_stats="

insert overwrite table ${APP}.ads\_coupon\_stats

select \* from ${APP}.ads\_coupon\_stats

union

select

'$do\_date' dt,

coupon\_id,

coupon\_name,

start\_date,

coupon\_rule,

cast(coupon\_reduce\_amount\_30d/original\_amount\_30d as decimal(16,2))

from ${APP}.dws\_trade\_coupon\_order\_nd

where dt='$do\_date';

"

ads\_new\_buyer\_stats="

insert overwrite table ${APP}.ads\_new\_buyer\_stats

select \* from ${APP}.ads\_new\_buyer\_stats

union

select

'$do\_date',

odr.recent\_days,

new\_order\_user\_count,

new\_payment\_user\_count

from

(

select

recent\_days,

sum(if(order\_date\_first>=date\_add('$do\_date',-recent\_days+1),1,0)) new\_order\_user\_count

from ${APP}.dws\_trade\_user\_order\_td lateral view explode(array(1,7,30)) tmp as recent\_days

where dt='$do\_date'

group by recent\_days

)odr

join

(

select

recent\_days,

sum(if(payment\_date\_first>=date\_add('$do\_date',-recent\_days+1),1,0)) new\_payment\_user\_count

from ${APP}.dws\_trade\_user\_payment\_td lateral view explode(array(1,7,30)) tmp as recent\_days

where dt='$do\_date'

group by recent\_days

)pay

on odr.recent\_days=pay.recent\_days;

"

ads\_order\_by\_province="

insert overwrite table ${APP}.ads\_order\_by\_province

select \* from ${APP}.ads\_order\_by\_province

union

select

'$do\_date' dt,

1 recent\_days,

province\_id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2,

order\_count\_1d,

order\_total\_amount\_1d

from ${APP}.dws\_trade\_province\_order\_1d

where dt='$do\_date'

union

select

'$do\_date' dt,

recent\_days,

province\_id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2,

sum(order\_count),

sum(order\_total\_amount)

from

(

select

recent\_days,

province\_id,

province\_name,

area\_code,

iso\_code,

iso\_3166\_2,

case recent\_days

when 7 then order\_count\_7d

when 30 then order\_count\_30d

end order\_count,

case recent\_days

when 7 then order\_total\_amount\_7d

when 30 then order\_total\_amount\_30d

end order\_total\_amount

from ${APP}.dws\_trade\_province\_order\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='$do\_date'

)t1

group by recent\_days,province\_id,province\_name,area\_code,iso\_code,iso\_3166\_2;

"

ads\_page\_path="

insert overwrite table ${APP}.ads\_page\_path

select \* from ${APP}.ads\_page\_path

union

select

'$do\_date' dt,

recent\_days,

source,

nvl(target,'null'),

count(\*) path\_count

from

(

select

recent\_days,

concat('step-',rn,':',page\_id) source,

concat('step-',rn+1,':',next\_page\_id) target

from

(

select

recent\_days,

page\_id,

lead(page\_id,1,null) over(partition by session\_id,recent\_days) next\_page\_id,

row\_number() over (partition by session\_id,recent\_days order by view\_time) rn

from ${APP}.dwd\_traffic\_page\_view\_inc lateral view explode(array(1,7,30)) tmp as recent\_days

where dt>=date\_add('$do\_date',-recent\_days+1)

)t1

)t2

group by recent\_days,source,target;

"

ads\_repeat\_purchase\_by\_tm="

insert overwrite table ${APP}.ads\_repeat\_purchase\_by\_tm

select \* from ${APP}.ads\_repeat\_purchase\_by\_tm

union

select

'$do\_date' dt,

recent\_days,

tm\_id,

tm\_name,

cast(sum(if(order\_count>=2,1,0))/sum(if(order\_count>=1,1,0)) as decimal(16,2))

from

(

select

'$do\_date' dt,

recent\_days,

user\_id,

tm\_id,

tm\_name,

sum(order\_count) order\_count

from

(

select

recent\_days,

user\_id,

tm\_id,

tm\_name,

case recent\_days

when 7 then order\_count\_7d

when 30 then order\_count\_30d

end order\_count

from ${APP}.dws\_trade\_user\_sku\_order\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='$do\_date'

)t1

group by recent\_days,user\_id,tm\_id,tm\_name

)t2

group by recent\_days,tm\_id,tm\_name;

"

ads\_sku\_cart\_num\_top3\_by\_cate="

insert overwrite table ${APP}.ads\_sku\_cart\_num\_top3\_by\_cate

select \* from ${APP}.ads\_sku\_cart\_num\_top3\_by\_cate

union

select

'$do\_date' dt,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

sku\_id,

sku\_name,

cart\_num,

rk

from

(

select

sku\_id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

cart\_num,

rank() over (partition by category1\_id,category2\_id,category3\_id order by cart\_num desc) rk

from

(

select

sku\_id,

sum(sku\_num) cart\_num

from ${APP}.dwd\_trade\_cart\_full

where dt='$do\_date'

group by sku\_id

)cart

left join

(

select

id,

sku\_name,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name

from ${APP}.dim\_sku\_full

where dt='$do\_date'

)sku

on cart.sku\_id=sku.id

)t1

where rk<=3;

"

ads\_trade\_stats="

insert overwrite table ${APP}.ads\_trade\_stats

select \* from ${APP}.ads\_trade\_stats

union

select

'$do\_date',

odr.recent\_days,

order\_total\_amount,

order\_count,

order\_user\_count,

order\_refund\_count,

order\_refund\_user\_count

from

(

select

1 recent\_days,

sum(order\_total\_amount\_1d) order\_total\_amount,

sum(order\_count\_1d) order\_count,

count(\*) order\_user\_count

from ${APP}.dws\_trade\_user\_order\_1d

where dt='$do\_date'

union all

select

recent\_days,

sum(order\_total\_amount),

sum(order\_count),

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

from

(

select

recent\_days,

case recent\_days

when 7 then order\_total\_amount\_7d

when 30 then order\_total\_amount\_30d

end order\_total\_amount,

case recent\_days

when 7 then order\_count\_7d

when 30 then order\_count\_30d

end order\_count

from ${APP}.dws\_trade\_user\_order\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='$do\_date'

)t1

group by recent\_days

)odr

join

(

select

1 recent\_days,

sum(order\_refund\_count\_1d) order\_refund\_count,

count(\*) order\_refund\_user\_count

from ${APP}.dws\_trade\_user\_order\_refund\_1d

where dt='$do\_date'

union all

select

recent\_days,

sum(order\_refund\_count),

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

from

(

select

recent\_days,

case recent\_days

when 7 then order\_refund\_count\_7d

when 30 then order\_refund\_count\_30d

end order\_refund\_count

from ${APP}.dws\_trade\_user\_order\_refund\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='$do\_date'

)t1

group by recent\_days

)refund

on odr.recent\_days=refund.recent\_days;

"

ads\_trade\_stats\_by\_cate="

insert overwrite table ${APP}.ads\_trade\_stats\_by\_cate

select \* from ${APP}.ads\_trade\_stats\_by\_cate

union

select

'$do\_date' dt,

nvl(odr.recent\_days,refund.recent\_days),

nvl(odr.category1\_id,refund.category1\_id),

nvl(odr.category1\_name,refund.category1\_name),

nvl(odr.category2\_id,refund.category2\_id),

nvl(odr.category2\_name,refund.category2\_name),

nvl(odr.category3\_id,refund.category3\_id),

nvl(odr.category3\_name,refund.category3\_name),

nvl(order\_count,0),

nvl(order\_user\_count,0),

nvl(order\_refund\_count,0),

nvl(order\_refund\_user\_count,0)

from

(

select

1 recent\_days,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

sum(order\_count\_1d) order\_count,

count(distinct(user\_id)) order\_user\_count

from ${APP}.dws\_trade\_user\_sku\_order\_1d

where dt='$do\_date'

group by category1\_id,category1\_name,category2\_id,category2\_name,category3\_id,category3\_name

union all

select

recent\_days,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

sum(order\_count),

count(distinct(if(order\_count>0,user\_id,null)))

from

(

select

recent\_days,

user\_id,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

case recent\_days

when 7 then order\_count\_7d

when 30 then order\_count\_30d

end order\_count

from ${APP}.dws\_trade\_user\_sku\_order\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='$do\_date'

)t1

group by recent\_days,category1\_id,category1\_name,category2\_id,category2\_name,category3\_id,category3\_name

)odr

full outer join

(

select

1 recent\_days,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

sum(order\_refund\_count\_1d) order\_refund\_count,

count(distinct(user\_id)) order\_refund\_user\_count

from ${APP}.dws\_trade\_user\_sku\_order\_refund\_1d

where dt='$do\_date'

group by category1\_id,category1\_name,category2\_id,category2\_name,category3\_id,category3\_name

union all

select

recent\_days,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

sum(order\_refund\_count),

count(distinct(if(order\_refund\_count>0,user\_id,null)))

from

(

select

recent\_days,

user\_id,

category1\_id,

category1\_name,

category2\_id,

category2\_name,

category3\_id,

category3\_name,

case recent\_days

when 7 then order\_refund\_count\_7d

when 30 then order\_refund\_count\_30d

end order\_refund\_count

from ${APP}.dws\_trade\_user\_sku\_order\_refund\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='$do\_date'

)t1

group by recent\_days,category1\_id,category1\_name,category2\_id,category2\_name,category3\_id,category3\_name

)refund

on odr.recent\_days=refund.recent\_days

and odr.category1\_id=refund.category1\_id

and odr.category1\_name=refund.category1\_name

and odr.category2\_id=refund.category2\_id

and odr.category2\_name=refund.category2\_name

and odr.category3\_id=refund.category3\_id

and odr.category3\_name=refund.category3\_name;

"

ads\_trade\_stats\_by\_tm="

insert overwrite table ${APP}.ads\_trade\_stats\_by\_tm

select \* from ${APP}.ads\_trade\_stats\_by\_tm

union

select

'$do\_date' dt,

nvl(odr.recent\_days,refund.recent\_days),

nvl(odr.tm\_id,refund.tm\_id),

nvl(odr.tm\_name,refund.tm\_name),

nvl(order\_count,0),

nvl(order\_user\_count,0),

nvl(order\_refund\_count,0),

nvl(order\_refund\_user\_count,0)

from

(

select

1 recent\_days,

tm\_id,

tm\_name,

sum(order\_count\_1d) order\_count,

count(distinct(user\_id)) order\_user\_count

from ${APP}.dws\_trade\_user\_sku\_order\_1d

where dt='$do\_date'

group by tm\_id,tm\_name

union all

select

recent\_days,

tm\_id,

tm\_name,

sum(order\_count),

count(distinct(if(order\_count>0,user\_id,null)))

from

(

select

recent\_days,

user\_id,

tm\_id,

tm\_name,

case recent\_days

when 7 then order\_count\_7d

when 30 then order\_count\_30d

end order\_count

from ${APP}.dws\_trade\_user\_sku\_order\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='$do\_date'

)t1

group by recent\_days,tm\_id,tm\_name

)odr

full outer join

(

select

1 recent\_days,

tm\_id,

tm\_name,

sum(order\_refund\_count\_1d) order\_refund\_count,

count(distinct(user\_id)) order\_refund\_user\_count

from ${APP}.dws\_trade\_user\_sku\_order\_refund\_1d

where dt='$do\_date'

group by tm\_id,tm\_name

union all

select

recent\_days,

tm\_id,

tm\_name,

sum(order\_refund\_count),

count(if(order\_refund\_count>0,user\_id,null))

from

(

select

recent\_days,

user\_id,

tm\_id,

tm\_name,

case recent\_days

when 7 then order\_refund\_count\_7d

when 30 then order\_refund\_count\_30d

end order\_refund\_count

from ${APP}.dws\_trade\_user\_sku\_order\_refund\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='$do\_date'

)t1

group by recent\_days,tm\_id,tm\_name

)refund

on odr.recent\_days=refund.recent\_days

and odr.tm\_id=refund.tm\_id

and odr.tm\_name=refund.tm\_name;

"

ads\_traffic\_stats\_by\_channel="

insert overwrite table ${APP}.ads\_traffic\_stats\_by\_channel

select \* from ${APP}.ads\_traffic\_stats\_by\_channel

union

select

'$do\_date' dt,

recent\_days,

channel,

cast(count(distinct(mid\_id)) as bigint) uv\_count,

cast(avg(during\_time\_1d)/1000 as bigint) avg\_duration\_sec,

cast(avg(page\_count\_1d) as bigint) avg\_page\_count,

cast(count(\*) as bigint) sv\_count,

cast(sum(if(page\_count\_1d=1,1,0))/count(\*) as decimal(16,2)) bounce\_rate

from ${APP}.dws\_traffic\_session\_page\_view\_1d lateral view explode(array(1,7,30)) tmp as recent\_days

where dt>=date\_add('$do\_date',-recent\_days+1)

group by recent\_days,channel;

"

ads\_user\_action="

insert overwrite table ${APP}.ads\_user\_action

select \* from ${APP}.ads\_user\_action

union

select

'$do\_date' dt,

page.recent\_days,

home\_count,

good\_detail\_count,

cart\_count,

order\_count,

payment\_count

from

(

select

1 recent\_days,

sum(if(page\_id='home',1,0)) home\_count,

sum(if(page\_id='good\_detail',1,0)) good\_detail\_count

from ${APP}.dws\_traffic\_page\_visitor\_page\_view\_1d

where dt='$do\_date'

and page\_id in ('home','good\_detail')

union all

select

recent\_days,

sum(if(page\_id='home' and view\_count>0,1,0)),

sum(if(page\_id='good\_detail' and view\_count>0,1,0))

from

(

select

recent\_days,

page\_id,

case recent\_days

when 7 then view\_count\_7d

when 30 then view\_count\_30d

end view\_count

from ${APP}.dws\_traffic\_page\_visitor\_page\_view\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='$do\_date'

and page\_id in ('home','good\_detail')

)t1

group by recent\_days

)page

join

(

select

1 recent\_days,

count(\*) cart\_count

from ${APP}.dws\_trade\_user\_cart\_add\_1d

where dt='$do\_date'

union all

select

recent\_days,

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

from

(

select

recent\_days,

case recent\_days

when 7 then cart\_add\_count\_7d

when 30 then cart\_add\_count\_30d

end cart\_count

from ${APP}.dws\_trade\_user\_cart\_add\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='$do\_date'

)t1

group by recent\_days

)cart

on page.recent\_days=cart.recent\_days

join

(

select

1 recent\_days,

count(\*) order\_count

from ${APP}.dws\_trade\_user\_order\_1d

where dt='$do\_date'

union all

select

recent\_days,

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

from

(

select

recent\_days,

case recent\_days

when 7 then order\_count\_7d

when 30 then order\_count\_30d

end order\_count

from ${APP}.dws\_trade\_user\_order\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='$do\_date'

)t1

group by recent\_days

)ord

on page.recent\_days=ord.recent\_days

join

(

select

1 recent\_days,

count(\*) payment\_count

from ${APP}.dws\_trade\_user\_payment\_1d

where dt='$do\_date'

union all

select

recent\_days,

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

from

(

select

recent\_days,

case recent\_days

when 7 then payment\_count\_7d

when 30 then payment\_count\_30d

end order\_count

from ${APP}.dws\_trade\_user\_payment\_nd lateral view explode(array(7,30)) tmp as recent\_days

where dt='$do\_date'

)t1

group by recent\_days

)pay

on page.recent\_days=pay.recent\_days;

"

ads\_user\_change="

insert overwrite table ${APP}.ads\_user\_change

select \* from ${APP}.ads\_user\_change

union

select

churn.dt,

user\_churn\_count,

user\_back\_count

from

(

select

'$do\_date' dt,

count(\*) user\_churn\_count

from ${APP}.dws\_user\_user\_login\_td

where dt='$do\_date'

and login\_date\_last=date\_add('$do\_date',-7)

)churn

join

(

select

'$do\_date' dt,

count(\*) user\_back\_count

from

(

select

user\_id,

login\_date\_last

from ${APP}.dws\_user\_user\_login\_td

where dt='$do\_date'

)t1

join

(

select

user\_id,

login\_date\_last login\_date\_previous

from ${APP}.dws\_user\_user\_login\_td

where dt=date\_add('$do\_date',-1)

)t2

on t1.user\_id=t2.user\_id

where datediff(login\_date\_last,login\_date\_previous)>=8

)back

on churn.dt=back.dt;

"

ads\_user\_retention="

insert overwrite table ${APP}.ads\_user\_retention

select \* from ${APP}.ads\_user\_retention

union

select

'$do\_date' dt,

login\_date\_first create\_date,

datediff('$do\_date',login\_date\_first) retention\_day,

sum(if(login\_date\_last='$do\_date',1,0)) retention\_count,

count(\*) new\_user\_count,

cast(sum(if(login\_date\_last='$do\_date',1,0))/count(\*)\*100 as decimal(16,2)) retention\_rate

from

(

select

user\_id,

date\_id login\_date\_first

from ${APP}.dwd\_user\_register\_inc

where dt>=date\_add('$do\_date',-7)

and dt<'$do\_date'

)t1

join

(

select

user\_id,

login\_date\_last

from ${APP}.dws\_user\_user\_login\_td

where dt='$do\_date'

)t2

on t1.user\_id=t2.user\_id

group by login\_date\_first;

"

ads\_user\_stats="

insert overwrite table ${APP}.ads\_user\_stats

select \* from ${APP}.ads\_user\_stats

union

select

'$do\_date' dt,

t1.recent\_days,

new\_user\_count,

active\_user\_count

from

(

select

recent\_days,

sum(if(login\_date\_last>=date\_add('$do\_date',-recent\_days+1),1,0)) new\_user\_count

from ${APP}.dws\_user\_user\_login\_td lateral view explode(array(1,7,30)) tmp as recent\_days

where dt='$do\_date'

group by recent\_days

)t1

join

(

select

recent\_days,

sum(if(date\_id>=date\_add('$do\_date',-recent\_days+1),1,0)) active\_user\_count

from ${APP}.dwd\_user\_register\_inc lateral view explode(array(1,7,30)) tmp as recent\_days

group by recent\_days

)t2

on t1.recent\_days=t2.recent\_days;

"

case $1 in

"ads\_activity\_stats" )

hive -e "$ads\_activity\_stats"

;;

"ads\_coupon\_stats" )

hive -e "$ads\_coupon\_stats"

;;

"ads\_new\_buyer\_stats" )

hive -e "$ads\_new\_buyer\_stats"

;;

"ads\_order\_by\_province" )

hive -e "$ads\_order\_by\_province"

;;

"ads\_page\_path" )

hive -e "$ads\_page\_path"

;;

"ads\_repeat\_purchase\_by\_tm" )

hive -e "$ads\_repeat\_purchase\_by\_tm"

;;

"ads\_sku\_cart\_num\_top3\_by\_cate" )

hive -e "$ads\_sku\_cart\_num\_top3\_by\_cate"

;;

"ads\_trade\_stats" )

hive -e "$ads\_trade\_stats"

;;

"ads\_trade\_stats\_by\_cate" )

hive -e "$ads\_trade\_stats\_by\_cate"

;;

"ads\_trade\_stats\_by\_tm" )

hive -e "$ads\_trade\_stats\_by\_tm"

;;

"ads\_traffic\_stats\_by\_channel" )

hive -e "$ads\_traffic\_stats\_by\_channel"

;;

"ads\_user\_action" )

hive -e "$ads\_user\_action"

;;

"ads\_user\_change" )

hive -e "$ads\_user\_change"

;;

"ads\_user\_retention" )

hive -e "$ads\_user\_retention"

;;

"ads\_user\_stats" )

hive -e "$ads\_user\_stats"

;;

"all" )

hive -e "$ads\_activity\_stats$ads\_coupon\_stats$ads\_new\_buyer\_stats$ads\_order\_by\_province$ads\_page\_path$ads\_repeat\_purchase\_by\_tm$ads\_sku\_cart\_num\_top3\_by\_cate$ads\_trade\_stats$ads\_trade\_stats\_by\_cate$ads\_trade\_stats\_by\_tm$ads\_traffic\_stats\_by\_channel$ads\_user\_action$ads\_user\_change$ads\_user\_retention$ads\_user\_stats"

;;

esac

（3）增加脚本执行权限

[atguigu@hadoop102 bin]$ chmod +x dws\_to\_ads.sh

（4）脚本用法

[atguigu@hadoop102 bin]$ dws\_to\_ads.sh all 2020-06-14

# 这里运行dws\_to\_ads的时候一直疯狂报错：

ERROR org.apache.hive.spark.client.SparkClientImpl - Timed out waiting for client to connect.

Possible reasons include network issues, errors in remote driver or the cluster has no available resources, etc.

Please check YARN or Spark driver's logs for further information.

java.util.concurrent.ExecutionException: java.util.concurrent.TimeoutException: Timed out waiting for client connection.

重启一下hiveserver2问题解决。

# 第12章 报表数据导出

为方便报表应用使用数据，需将ads各指标的统计结果导出到MySQL数据库中。

## 12.1 MySQL建库建表

### 12.1.1 创建数据库

CREATE DATABASE IF NOT EXISTS gmall\_report DEFAULT CHARSET utf8 COLLATE utf8\_general\_ci;

### 12.1.2 创建表

**1）各活动补贴率**

DROP TABLE IF EXISTS `ads\_activity\_stats`;

CREATE TABLE `ads\_activity\_stats` (

`dt` date NOT NULL COMMENT '统计日期',

`activity\_id` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '活动ID',

`activity\_name` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '活动名称',

`start\_date` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '活动开始日期',

`reduce\_rate` decimal(16, 2) NULL DEFAULT NULL COMMENT '补贴率',

PRIMARY KEY (`dt`, `activity\_id`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '活动统计' ROW\_FORMAT = Dynamic;

**2）各优惠券补贴率**

DROP TABLE IF EXISTS `ads\_coupon\_stats`;

CREATE TABLE `ads\_coupon\_stats` (

`dt` date NOT NULL COMMENT '统计日期',

`coupon\_id` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '优惠券ID',

`coupon\_name` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '优惠券名称',

`start\_date` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '发布日期',

`rule\_name` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '优惠规则，例如满100元减10元',

`reduce\_rate` decimal(16, 2) NULL DEFAULT NULL COMMENT '补贴率',

PRIMARY KEY (`dt`, `coupon\_id`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '优惠券统计' ROW\_FORMAT = Dynamic;

**3）新增交易用户统计**

DROP TABLE IF EXISTS `ads\_new\_buyer\_stats`;

CREATE TABLE `ads\_new\_buyer\_stats` (

`dt` date NOT NULL COMMENT '统计日期',

`recent\_days` bigint(20) NOT NULL COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

`new\_order\_user\_count` bigint(20) NULL DEFAULT NULL COMMENT '新增下单人数',

`new\_payment\_user\_count` bigint(20) NULL DEFAULT NULL COMMENT '新增支付人数',

PRIMARY KEY (`dt`, `recent\_days`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '新增交易用户统计' ROW\_FORMAT = Dynamic;

**4）各省份订单统计**

DROP TABLE IF EXISTS `ads\_order\_by\_province`;

CREATE TABLE `ads\_order\_by\_province` (

`dt` date NOT NULL COMMENT '统计日期',

`recent\_days` bigint(20) NOT NULL COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

`province\_id` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '省份ID',

`province\_name` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '省份名称',

`area\_code` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '地区编码',

`iso\_code` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '国际标准地区编码',

`iso\_code\_3166\_2` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '国际标准地区编码',

`order\_count` bigint(20) NULL DEFAULT NULL COMMENT '订单数',

`order\_total\_amount` decimal(16, 2) NULL DEFAULT NULL COMMENT '订单金额',

PRIMARY KEY (`dt`, `recent\_days`, `province\_id`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '各地区订单统计' ROW\_FORMAT = Dynamic;

**5）用户路径分析**

DROP TABLE IF EXISTS `ads\_page\_path`;

CREATE TABLE `ads\_page\_path` (

`dt` date NOT NULL COMMENT '统计日期',

`recent\_days` bigint(20) NOT NULL COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

`source` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '跳转起始页面ID',

`target` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '跳转终到页面ID',

`path\_count` bigint(20) NULL DEFAULT NULL COMMENT '跳转次数',

PRIMARY KEY (`dt`, `recent\_days`, `source`, `target`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '页面浏览路径分析' ROW\_FORMAT = Dynamic;

**6）各品牌复购率**

DROP TABLE IF EXISTS `ads\_repeat\_purchase\_by\_tm`;

CREATE TABLE `ads\_repeat\_purchase\_by\_tm` (

`dt` date NOT NULL COMMENT '统计日期',

`recent\_days` bigint(20) NOT NULL COMMENT '最近天数,7:最近7天,30:最近30天',

`tm\_id` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '品牌ID',

`tm\_name` varchar(32) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '品牌名称',

`order\_repeat\_rate` decimal(16, 2) NULL DEFAULT NULL COMMENT '复购率',

PRIMARY KEY (`dt`, `recent\_days`, `tm\_id`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '各品牌复购率统计' ROW\_FORMAT = Dynamic;

**7）各品类商品购物车存量topN**

DROP TABLE IF EXISTS `ads\_sku\_cart\_num\_top3\_by\_cate`;

CREATE TABLE `ads\_sku\_cart\_num\_top3\_by\_cate` (

`dt` date NOT NULL COMMENT '统计日期',

`category1\_id` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '一级分类ID',

`category1\_name` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '一级分类名称',

`category2\_id` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '二级分类ID',

`category2\_name` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '二级分类名称',

`category3\_id` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '三级分类ID',

`category3\_name` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '三级分类名称',

`sku\_id` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '商品id',

`sku\_name` varchar(128) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '商品名称',

`cart\_num` bigint(20) NULL DEFAULT NULL COMMENT '购物车中商品数量',

`rk` bigint(20) NULL DEFAULT NULL COMMENT '排名',

PRIMARY KEY (`dt`, `sku\_id`, `category1\_id`, `category2\_id`, `category3\_id`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '各分类商品购物车存量Top10' ROW\_FORMAT = Dynamic;

**8）交易综合统计**

DROP TABLE IF EXISTS `ads\_trade\_stats`;

CREATE TABLE `ads\_trade\_stats` (

`dt` date NOT NULL COMMENT '统计日期',

`recent\_days` bigint(255) NOT NULL COMMENT '最近天数,1:最近1日,7:最近7天,30:最近30天',

`order\_total\_amount` decimal(16, 2) NULL DEFAULT NULL COMMENT '订单总额,GMV',

`order\_count` bigint(20) NULL DEFAULT NULL COMMENT '订单数',

`order\_user\_count` bigint(20) NULL DEFAULT NULL COMMENT '下单人数',

`order\_refund\_count` bigint(20) NULL DEFAULT NULL COMMENT '退单数',

`order\_refund\_user\_count` bigint(20) NULL DEFAULT NULL COMMENT '退单人数',

PRIMARY KEY (`dt`, `recent\_days`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '交易统计' ROW\_FORMAT = Dynamic;

**9）各品类商品交易统计**

DROP TABLE IF EXISTS `ads\_trade\_stats\_by\_cate`;

CREATE TABLE `ads\_trade\_stats\_by\_cate` (

`dt` date NOT NULL COMMENT '统计日期',

`recent\_days` bigint(20) NOT NULL COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

`category1\_id` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '一级分类id',

`category1\_name` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '一级分类名称',

`category2\_id` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '二级分类id',

`category2\_name` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '二级分类名称',

`category3\_id` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '三级分类id',

`category3\_name` varchar(64) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '三级分类名称',

`order\_count` bigint(20) NULL DEFAULT NULL COMMENT '订单数',

`order\_user\_count` bigint(20) NULL DEFAULT NULL COMMENT '订单人数',

`order\_refund\_count` bigint(20) NULL DEFAULT NULL COMMENT '退单数',

`order\_refund\_user\_count` bigint(20) NULL DEFAULT NULL COMMENT '退单人数',

PRIMARY KEY (`dt`, `recent\_days`, `category1\_id`, `category2\_id`, `category3\_id`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '各分类商品交易统计' ROW\_FORMAT = Dynamic;

**10）各品牌商品交易统计**

DROP TABLE IF EXISTS `ads\_trade\_stats\_by\_tm`;

CREATE TABLE `ads\_trade\_stats\_by\_tm` (

`dt` date NOT NULL COMMENT '统计日期',

`recent\_days` bigint(20) NOT NULL COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

`tm\_id` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '品牌ID',

`tm\_name` varchar(32) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '品牌名称',

`order\_count` bigint(20) NULL DEFAULT NULL COMMENT '订单数',

`order\_user\_count` bigint(20) NULL DEFAULT NULL COMMENT '订单人数',

`order\_refund\_count` bigint(20) NULL DEFAULT NULL COMMENT '退单数',

`order\_refund\_user\_count` bigint(20) NULL DEFAULT NULL COMMENT '退单人数',

PRIMARY KEY (`dt`, `recent\_days`, `tm\_id`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '各品牌商品交易统计' ROW\_FORMAT = Dynamic;

**11）各渠道流量统计**

DROP TABLE IF EXISTS `ads\_traffic\_stats\_by\_channel`;

CREATE TABLE `ads\_traffic\_stats\_by\_channel` (

`dt` date NOT NULL COMMENT '统计日期',

`recent\_days` bigint(20) NOT NULL COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

`channel` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '渠道',

`uv\_count` bigint(20) NULL DEFAULT NULL COMMENT '访客人数',

`avg\_duration\_sec` bigint(20) NULL DEFAULT NULL COMMENT '会话平均停留时长，单位为秒',

`avg\_page\_count` bigint(20) NULL DEFAULT NULL COMMENT '会话平均浏览页面数',

`sv\_count` bigint(20) NULL DEFAULT NULL COMMENT '会话数',

`bounce\_rate` decimal(16, 2) NULL DEFAULT NULL COMMENT '跳出率',

PRIMARY KEY (`dt`, `recent\_days`, `channel`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '各渠道流量统计' ROW\_FORMAT = Dynamic;

**12）用户行为漏斗分析**

DROP TABLE IF EXISTS `ads\_user\_action`;

CREATE TABLE `ads\_user\_action` (

`dt` date NOT NULL COMMENT '统计日期',

`recent\_days` bigint(20) NOT NULL COMMENT '最近天数,1:最近1天,7:最近7天,30:最近30天',

`home\_count` bigint(20) NULL DEFAULT NULL COMMENT '浏览首页人数',

`good\_detail\_count` bigint(20) NULL DEFAULT NULL COMMENT '浏览商品详情页人数',

`cart\_count` bigint(20) NULL DEFAULT NULL COMMENT '加入购物车人数',

`order\_count` bigint(20) NULL DEFAULT NULL COMMENT '下单人数',

`payment\_count` bigint(20) NULL DEFAULT NULL COMMENT '支付人数',

PRIMARY KEY (`dt`, `recent\_days`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '漏斗分析' ROW\_FORMAT = Dynamic;

**13）用户变动统计**

DROP TABLE IF EXISTS `ads\_user\_change`;

CREATE TABLE `ads\_user\_change` (

`dt` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '统计日期',

`user\_churn\_count` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '流失用户数',

`user\_back\_count` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '回流用户数',

PRIMARY KEY (`dt`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '用户变动统计' ROW\_FORMAT = Dynamic;

**14）用户留存率**

DROP TABLE IF EXISTS `ads\_user\_retention`;

CREATE TABLE `ads\_user\_retention` (

`dt` date NOT NULL COMMENT '统计日期',

`create\_date` varchar(16) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '用户新增日期',

`retention\_day` int(20) NOT NULL COMMENT '截至当前日期留存天数',

`retention\_count` bigint(20) NULL DEFAULT NULL COMMENT '留存用户数量',

`new\_user\_count` bigint(20) NULL DEFAULT NULL COMMENT '新增用户数量',

`retention\_rate` decimal(16, 2) NULL DEFAULT NULL COMMENT '留存率',

PRIMARY KEY (`dt`, `create\_date`, `retention\_day`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '留存率' ROW\_FORMAT = Dynamic;

**15）用户新增活跃统计**

DROP TABLE IF EXISTS `ads\_user\_stats`;

CREATE TABLE `ads\_user\_stats` (

`dt` date NOT NULL COMMENT '统计日期',

`recent\_days` bigint(20) NOT NULL COMMENT '最近n日,1:最近1日,7:最近7日,30:最近30日',

`new\_user\_count` bigint(20) NULL DEFAULT NULL COMMENT '新增用户数',

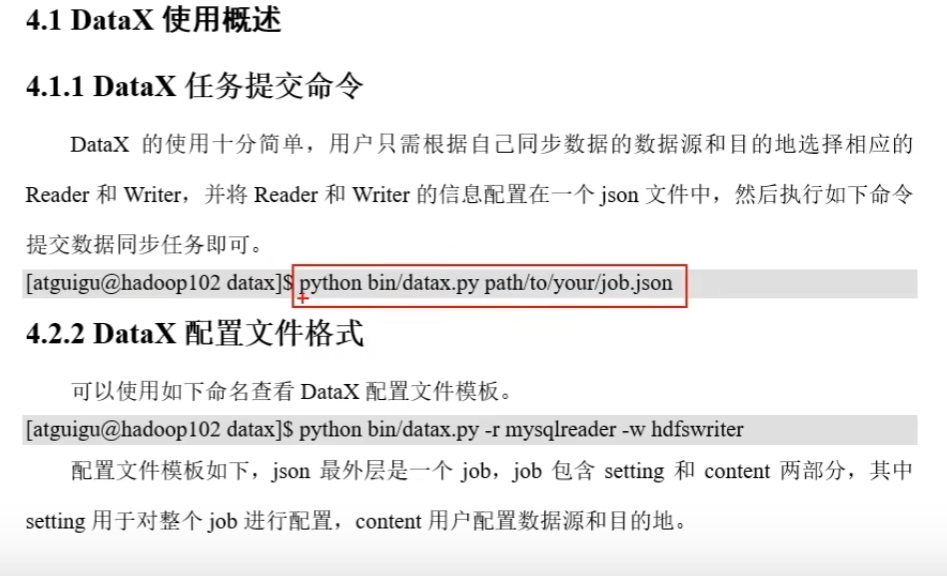
`active\_user\_count` bigint(20) NULL DEFAULT NULL COMMENT '活跃用户数',

PRIMARY KEY (`dt`, `recent\_days`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci COMMENT = '用户新增活跃统计' ROW\_FORMAT = Dynamic;

## 12.2 数据导出

数据导出工具选用DataX，选用HDFSReader和MySQLWriter。



### 12.2.1 编写DataX配置文件(hdfs写入到mysql的工具)

Insert into和replace into表中必须有组件

我们需要为每个张表编写一个DataX配置文件，此处以ads\_traffic\_stats\_by\_channel为例，配置文件内容如下：

Column属性为需要同步的列，可以使用索引选择所需列，例如[{“index”:0,”type”:”long”},{“index”:1,”type”:”boolean”}]标识前两列，[“\*”]标识所有列。

“fileType”:”text”,文件类型，目前支持textfile(text)、orcfile(orc)、rcfile(rc)、sequence file(seq)、csv文件(csv)

nullFormat:null值的存储格式\\N，在hive中的存储格式跟在mysql中的存储格式不一样。

Writemode:replace(写入方式)，控制写入数据采用insert into(insert)或者replace into(replace)

Replace into会直接把表中已存在的数据给删除掉，path当中需要传入一个exportdir

${exportdir}引用具体的文件，这意味着在导入具体的文件时候需要传入一个具体的参数

!!!写完之后运行下面的命令启动datax的进程!!!

python /export/server/datax/bin/datax.py -p”-Dexportdir=/warehouse/gmall/ads/ads\_traffic\_stats\_by\_channel” /export/server/datax/conf/datax\_config.json

{

"job": {

"content": [

{

"reader": {

"name": "hdfsreader",

"parameter": {

"column": [

"\*"

],

"defaultFS": "hdfs://node1:8020",

"encoding": "UTF-8",

"fieldDelimiter": "\t",

"fileType": "text",

"nullFormat": "\\N",

"path": "**${exportdir}**"

}

},

"writer": {

"name": "mysqlwriter",

"parameter": {

"column": [

"dt",

"recent\_days",

"channel",

"uv\_count",

"avg\_duration\_sec",

"avg\_page\_count",

"sv\_count",

"bounce\_rate"

],

"connection": [

{

"jdbcUrl": "jdbc:mysql://node1:3306/gmall\_report?useUnicode=true&characterEncoding=utf-8",

"table": [

"ads\_traffic\_stats\_by\_channel"

]

}

],

"password": "000000",

"username": "root",

"writeMode": "replace"

}

}

}

],

"setting": {

"errorLimit": {

"percentage": 0.02,

"record": 0

},

"speed": {

"channel": 3

}

}

}

}

注：导出路径path参数并未写死，需在提交任务时通过参数动态传入，参数名称为exportdir。

### 12.2.2 DataX配置文件生成脚本

方便起见，此处提供了DataX配置文件批量生成脚本，脚本内容及使用方式如下。

**1）在~/bin目录下创建gen\_export\_config.py脚本**

[atguigu@hadoop102 bin]$ vim ~/bin/gen\_export\_config.py

脚本内容如下

# coding=utf-8

import json

import getopt

import os

import sys

import pymysql

#MySQL相关配置，需根据实际情况作出修改

mysql\_host = "node1"

mysql\_port = "3306"

mysql\_user = "root"

mysql\_passwd = "000000"

#HDFS NameNode相关配置，需根据实际情况作出修改

hdfs\_nn\_host = "node1"

hdfs\_nn\_port = "8020"

#生成配置文件的目标路径，可根据实际情况作出修改

output\_path = "/export/datax/data/job/export"

def get\_connection():

return pymysql.connect(host=mysql\_host, port=int(mysql\_port), user=mysql\_user, passwd=mysql\_passwd)

def get\_mysql\_meta(database, table):

connection = get\_connection()

cursor = connection.cursor()

sql = "SELECT COLUMN\_NAME,DATA\_TYPE from information\_schema.COLUMNS WHERE TABLE\_SCHEMA=%s AND TABLE\_NAME=%s ORDER BY ORDINAL\_POSITION"

cursor.execute(sql, [database, table])

fetchall = cursor.fetchall()

cursor.close()

connection.close()

return fetchall

def get\_mysql\_columns(database, table):

results = get\_mysql\_meta(database,table)

resultlist = []

for data in results:

resultlist.append(data[0])

return resultlist

def generate\_json(target\_database, target\_table):

job = {

"job": {

"setting": {

"speed": {

"channel": 3

},

"errorLimit": {

"record": 0,

"percentage": 0.02

}

},

"content": [{

"reader": {

"name": "hdfsreader",

"parameter": {

"path": "${exportdir}",

"defaultFS": "hdfs://" + hdfs\_nn\_host + ":" + hdfs\_nn\_port,

"column": ["\*"],

"fileType": "text",

"encoding": "UTF-8",

"fieldDelimiter": "\t",

"nullFormat": "\\N"

}

},

"writer": {

"name": "mysqlwriter",

"parameter": {

"writeMode": "replace",

"username": mysql\_user,

"password": mysql\_passwd,

"column": get\_mysql\_columns(target\_database, target\_table),

"connection": [

{

"jdbcUrl":

"jdbc:mysql://" + mysql\_host + ":" + mysql\_port + "/" + target\_database + "?useUnicode=true&characterEncoding=utf-8",

"table": [target\_table]

}

]

}

}

}]

}

}

if not os.path.exists(output\_path):

os.makedirs(output\_path)

with open(os.path.join(output\_path, ".".join([target\_database, target\_table, "json"])), "w") as f:

json.dump(job, f)

def main(args):

target\_database = ""

target\_table = ""

options, arguments = getopt.getopt(args, '-d:-t:', ['targetdb=', 'targettbl='])

for opt\_name, opt\_value in options:

if opt\_name in ('-d', '--targetdb'):

target\_database = opt\_value

if opt\_name in ('-t', '--targettbl'):

target\_table = opt\_value

generate\_json(target\_database, target\_table)

if \_\_name\_\_ == '\_\_main\_\_':

main(sys.argv[1:])

**注：**

**（1）安装Python Mysql驱动**

**由于需要使用Python访问Mysql数据库，故需安装驱动，命令如下：**

[atguigu@hadoop102 bin]$ sudo yum install -y MySQL-python

**（2）脚本使用说明**

python gen\_export\_config.py **-d** database **-t** table

通过-d传入MySQL数据库名，-t传入MySQL表名，执行上述命令即可生成该表的DataX同步配置文件。

**2）在~/bin目录下创建gen\_export\_config.sh脚本**

[atguigu@hadoop102 bin]$ vim ~/bin/gen\_export\_config.sh

脚本内容如下。

#!/bin/bash

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_activity\_stats

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_coupon\_stats

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_new\_buyer\_stats

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_order\_by\_province

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_page\_path

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_repeat\_purchase\_by\_tm

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_sku\_cart\_num\_top3\_by\_cate

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_trade\_stats

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_trade\_stats\_by\_cate

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_trade\_stats\_by\_tm

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_traffic\_stats\_by\_channel

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_user\_action

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_user\_change

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_user\_retention

python ~/bin/gen\_export\_config.py -d gmall\_report -t ads\_user\_stats

**3）为gen\_export\_config.sh脚本增加执行权限**

[atguigu@hadoop102 bin]$ chmod +x ~/bin/gen\_export\_config.sh

**4）执行gen\_export\_config.sh脚本，生成配置文件**

[atguigu@hadoop102 bin]$ gen\_export\_config.sh

**5）观察生成的配置文件**

[atguigu@hadoop102 bin]$ ls /opt/module/datax/job/export/

总用量 64

gmall\_report.ads\_activity\_stats.json gmall\_report.ads\_trade\_stats\_by\_cate.json

gmall\_report.ads\_coupon\_stats.json gmall\_report.ads\_trade\_stats\_by\_tm.json

gmall\_report.ads\_new\_buyer\_stats.json gmall\_report.ads\_trade\_stats.json

gmall\_report.ads\_order\_by\_province.json gmall\_report.ads\_traffic\_stats\_by\_channel.json

gmall\_report.ads\_user\_action.json

gmall\_report.ads\_page\_path.json gmall\_report.ads\_user\_change.json

gmall\_report.ads\_repeat\_purchase\_by\_tm.json gmall\_report.ads\_user\_retention.json

gmall\_report.ads\_sku\_cart\_num\_top3\_by\_cate.json gmall\_report.ads\_user\_stats.json

### 12.2.3 测试生成的DataX配置文件

以ads\_traffic\_stats\_by\_channel为例，测试用脚本生成的配置文件是否可用。

**1）执行DataX同步命令**

[atguigu@hadoop102 bin]$ python /export/server/datax/bin/datax.py -p"-Dexportdir=/warehouse/gmall/ads/ads\_trade\_stats\_by\_tm" /export/data/datax/job/export/gmall\_report.ads\_trade\_stats\_by\_tm.json

**2）观察同步结果**

观察MySQL目标表是否出现数据。

运行如果报错的情况，不一定数据没传入，具体看mysql目标表中是否出现数据。

### 12.2.4 编写每日导出脚本

（1）在hadoop102的/home/atguigu/bin目录下创建hdfs\_to\_mysql.sh

[atguigu@hadoop102 bin]$ vim hdfs\_to\_mysql.sh

1. 编写如下内容（重写一下脚本）
2. Target

#! /bin/bash

DATAX\_HOME=/export/server/datax

#DataX导出路径不允许存在空文件，该函数作用为清理空文件

#这里的target\_file=$1中的1代表函数传入的第一个参数

handle\_export\_path(){

target\_file=$1

for i in `hadoop fs -ls -R $1 | awk '{print $8}'`; do

hadoop fs -test -z $i

#hadoop fs -help test展示-test这个命令的用法

#hadoop fs -test -z /warehouse/gmall/ads/ads\_activity\_stats/000000\_0

#这条命令有一个返回值，使用echo $?获取到这个返回值

#如果为0说明文件为空，将文件删除

if [[ $? -eq 0 ]]; then

echo "$i文件大小为0，正在删除"

hadoop fs -rm -r -f $i

fi

done

}

#hadoop fs -ls -R $1 | awk ‘{print $8}’显示的是文件的绝对路径

#数据导出

export\_data() {

datax\_config=$1

export\_dir=$2

hadoop fs -test -e $export\_dir

if [[ $? -eq 0 ]]

then

handle\_export\_path $export\_dir

file\_count=$(hadoop fs -ls $export\_dir | wc -l)

if [ $file\_count -gt 0 ]

then

set -e;

$DATAX\_HOME/bin/datax.py -p"-Dexportdir=$export\_dir" $datax\_config

set +e;

else

echo "$export\_dir 目录为空，跳过~"

fi

else

echo "路径 $export\_dir 不存在，跳过~"

fi

}

case $1 in

"ads\_new\_buyer\_stats")

export\_data /export/data/datax/job/export/gmall\_report.ads\_new\_buyer\_stats.json /warehouse/gmall/ads/ads\_new\_buyer\_stats

;;

"ads\_order\_by\_province")

export\_data /export/data/datax/job/export/gmall\_report.ads\_order\_by\_province.json /warehouse/gmall/ads/ads\_order\_by\_province

;;

"ads\_page\_path")

export\_data /export/data/datax/job/export/gmall\_report.ads\_page\_path.json /warehouse/gmall/ads/ads\_page\_path

;;

"ads\_repeat\_purchase\_by\_tm")

export\_data /export/data/datax/job/export/gmall\_report.ads\_repeat\_purchase\_by\_tm.json /warehouse/gmall/ads/ads\_repeat\_purchase\_by\_tm

;;

"ads\_trade\_stats")

export\_data /export/data/datax/job/export/gmall\_report.ads\_trade\_stats.json /warehouse/gmall/ads/ads\_trade\_stats

;;

"ads\_trade\_stats\_by\_cate")

export\_data /export/data/datax/job/export/gmall\_report.ads\_trade\_stats\_by\_cate.json /warehouse/gmall/ads/ads\_trade\_stats\_by\_cate

;;

"ads\_trade\_stats\_by\_tm")

export\_data /export/data/datax/job/export/gmall\_report.ads\_trade\_stats\_by\_tm.json /warehouse/gmall/ads/ads\_trade\_stats\_by\_tm

;;

"ads\_traffic\_stats\_by\_channel")

export\_data /export/data/datax/job/export/gmall\_report.ads\_traffic\_stats\_by\_channel.json /warehouse/gmall/ads/ads\_traffic\_stats\_by\_channel

;;

"ads\_user\_action")

export\_data /export/data/datax/job/export/gmall\_report.ads\_user\_action.json /warehouse/gmall/ads/ads\_user\_action

;;

"ads\_user\_change")

export\_data /export/data/datax/job/export/gmall\_report.ads\_user\_change.json /warehouse/gmall/ads/ads\_user\_change

;;

"ads\_user\_retention")

export\_data /export/data/datax/job/export/gmall\_report.ads\_user\_retention.json /warehouse/gmall/ads/ads\_user\_retention

;;

"ads\_user\_stats")

export\_data /export/data/datax/job/export/gmall\_report.ads\_user\_stats.json /warehouse/gmall/ads/ads\_user\_stats

;;

"ads\_activity\_stats")

export\_data /export/data/datax/job/export/gmall\_report.ads\_activity\_stats.json /warehouse/gmall/ads/ads\_activity\_stats

;;

"ads\_coupon\_stats")

export\_data /export/data/datax/job/export/gmall\_report.ads\_coupon\_stats.json /warehouse/gmall/ads/ads\_coupon\_stats

;;

"ads\_sku\_cart\_num\_top3\_by\_cate")

export\_data /export/data/datax/job/export/gmall\_report.ads\_sku\_cart\_num\_top3\_by\_cate.json /warehouse/gmall/ads/ads\_sku\_cart\_num\_top3\_by\_cate

;;

"all")

export\_data /export/data/datax/job/export/gmall\_report.ads\_new\_buyer\_stats.json /warehouse/gmall/ads/ads\_new\_buyer\_stats

export\_data /export/data/datax/job/export/gmall\_report.ads\_order\_by\_province.json /warehouse/gmall/ads/ads\_order\_by\_province

export\_data /export/data/datax/job/export/gmall\_report.ads\_page\_path.json /warehouse/gmall/ads/ads\_page\_path

export\_data /export/data/datax/job/export/gmall\_report.ads\_repeat\_purchase\_by\_tm.json /warehouse/gmall/ads/ads\_repeat\_purchase\_by\_tm

export\_data /export/data/datax/job/export/gmall\_report.ads\_trade\_stats.json /warehouse/gmall/ads/ads\_trade\_stats

export\_data /export/data/datax/job/export/gmall\_report.ads\_trade\_stats\_by\_cate.json /warehouse/gmall/ads/ads\_trade\_stats\_by\_cate

export\_data /export/data/datax/job/export/gmall\_report.ads\_trade\_stats\_by\_tm.json /warehouse/gmall/ads/ads\_trade\_stats\_by\_tm

export\_data /export/data/datax/job/export/gmall\_report.ads\_traffic\_stats\_by\_channel.json /warehouse/gmall/ads/ads\_traffic\_stats\_by\_channel

export\_data /export/data/datax/job/export/gmall\_report.ads\_user\_action.json /warehouse/gmall/ads/ads\_user\_action

export\_data /export/data/datax/job/export/gmall\_report.ads\_user\_change.json /warehouse/gmall/ads/ads\_user\_change

export\_data /export/data/datax/job/export/gmall\_report.ads\_user\_retention.json /warehouse/gmall/ads/ads\_user\_retention

export\_data /export/data/datax/job/export/gmall\_report.ads\_user\_stats.json /warehouse/gmall/ads/ads\_user\_stats

export\_data /export/data/datax/job/export/gmall\_report.ads\_activity\_stats.json /warehouse/gmall/ads/ads\_activity\_stats

export\_data /export/data/datax/job/export/gmall\_report.ads\_coupon\_stats.json /warehouse/gmall/ads/ads\_coupon\_stats

export\_data /export/data/datax/job/export/gmall\_report.ads\_sku\_cart\_num\_top3\_by\_cate.json /warehouse/gmall/ads/ads\_sku\_cart\_num\_top3\_by\_cate

;;

esac

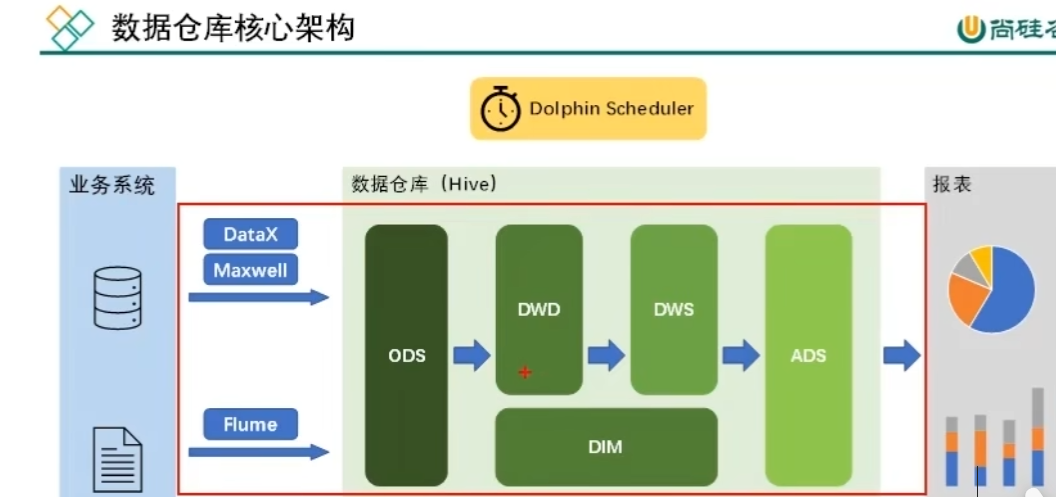
（3）增加脚本执行权限

[atguigu@hadoop102 bin]$ chmod +x hdfs\_to\_mysql.sh

（4）脚本用法

[atguigu@hadoop102 bin]$ hdfs\_to\_mysql.sh all

# 上述内容将核心架构的整个流程全部完成了



# 后面的工作流调度暂时没学

# 第13章 数据仓库工作流调度

## 13.1 调度工具部署



## 13.2 新数据生成

### 13.2.1 用户行为日志

**1）启动日志采集通道，包括Kafka、Flume等**

**（1）启动Zookeeper**

[atguigu@hadoop102 ~]$ zk.sh start

**（2）启动Kafka**

[atguigu@hadoop102 ~]$ kf.sh start

**（3）启动Flume**

[atguigu@hadoop102 ~]$ f1.sh start

[atguigu@hadoop102 ~]$ f2.sh start

**2）修改日志模拟器配置文件**

修改hadoop102和hadoop103两台节点中的/opt/module/applog/application.yml文件，修改mock.date参数如下。

mock.date: "2020-06-15"

**3）执行日志生成脚本**

[atguigu@hadoop102 ~]$ lg.sh

**4）观察HDFS上是否有2020-06-15的日志数据生成**

### 13.2.2 业务数据

**1）修改Maxwell配置文件**

**（1）修改/opt/module/maxwell/config.properties文件**

[atguigu@hadoop102 maxwell]$ vim /opt/module/maxwell/config.properties

**（2）修改mock\_date参数设置如下**

mock\_date=2020-06-15

**2）启动增量表采集通道，包括Maxwel、Kafka、Flume等**

**（1）启动Maxwell**

[atguigu@hadoop102 ~]$ mxw.sh start

**注：若Maxwell当前正在运行，为确保上述mock参数生效，需重启Maxwell。**

**（2）启动Flume**

[atguigu@hadoop102 ~]$ f3.sh start

**3）修改业务数据模拟器配置文件中的mock\_date参数**

mock.date=2020-06-15

**4）执行业务数据生成命令**

[atguigu@hadoop102 db\_log]$ java -jar gmall2020-mock-db-2021-10-10.jar

**5）观察HDFS上增量表是否有2020-06-15的数据生成**

## 13.3 工作流调度实操

由于DolphinScheduler集群模式启动进程较多，对虚拟机内存要求较高。故下面提供两种方式，可根据虚拟机内存情况进行选择。

### 13.3.1 DolphinScheduler集群模式

**1）启动DolphinScheduler**

[atguigu@hadoop102 dolphinscheduler]$ bin/start-all.sh

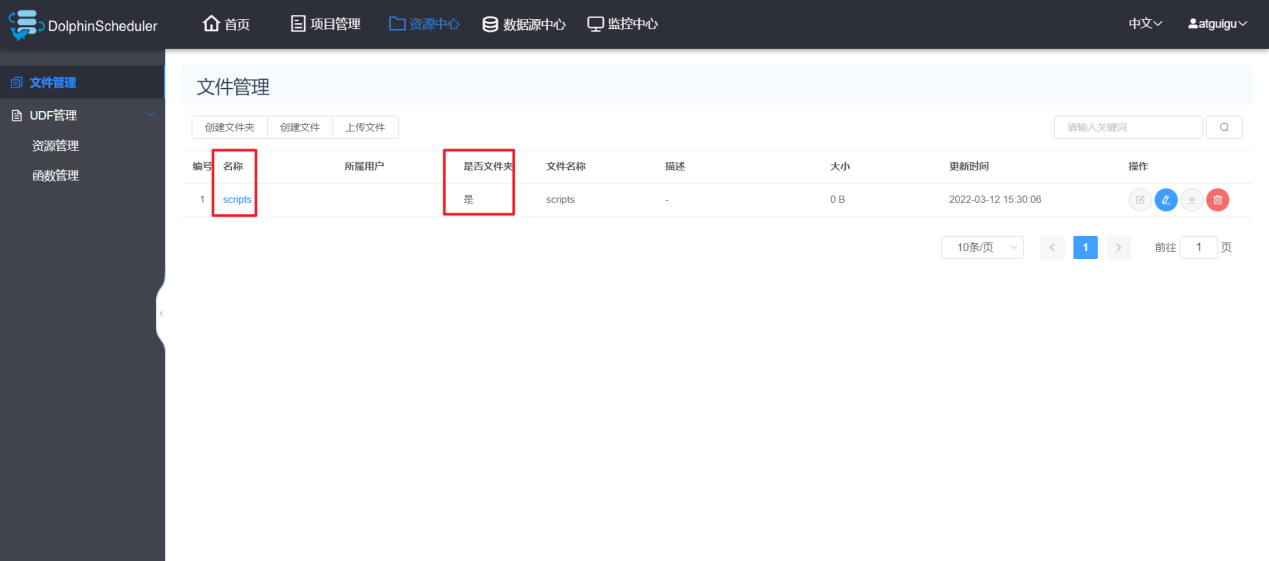
**2）使用普通用户登录**

图形用户界面, 应用程序

描述已自动生成

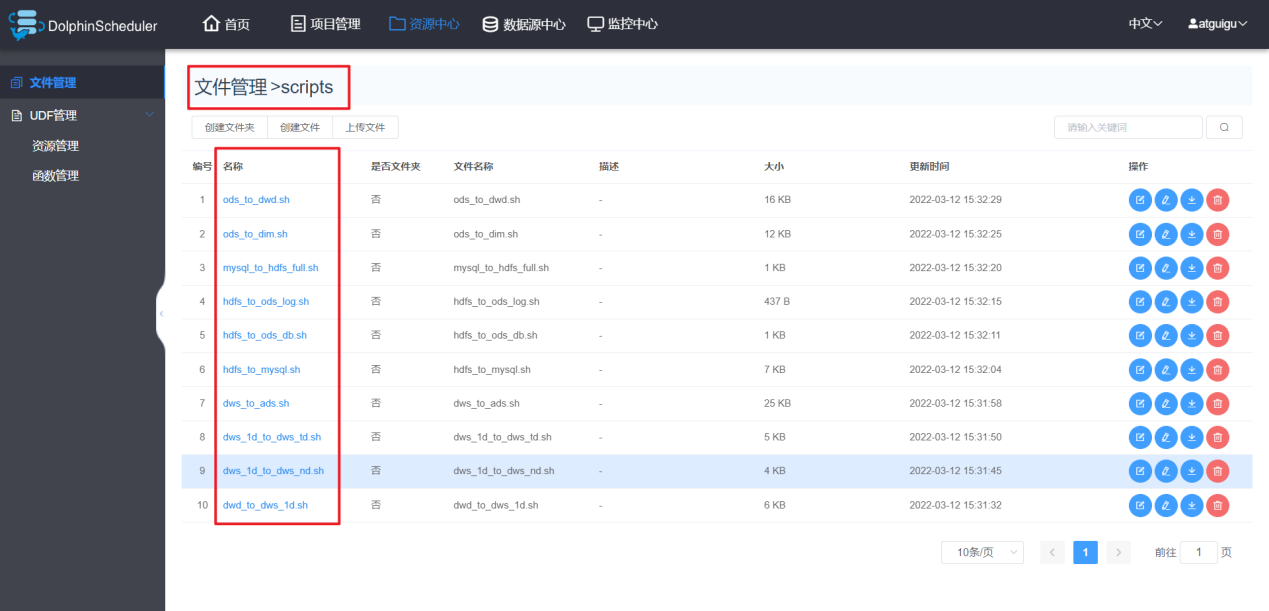
**3）向DolphinScheduler资源中心上传工作流所需脚本**

（1）创建文件夹



（2）上传工作流所需脚本

将工作流所需的所有脚本上传到资源中心scripts路径下，结果如下



**4）向DolphinScheduler的WorkerServer节点分发脚本依赖的组件**

由于工作流要执行的脚本需要调用Hive、DataX等组件，故在DolphinScheduler的集群模式下，需要确保每个WorkerServer节点都有脚本所依赖的组件。

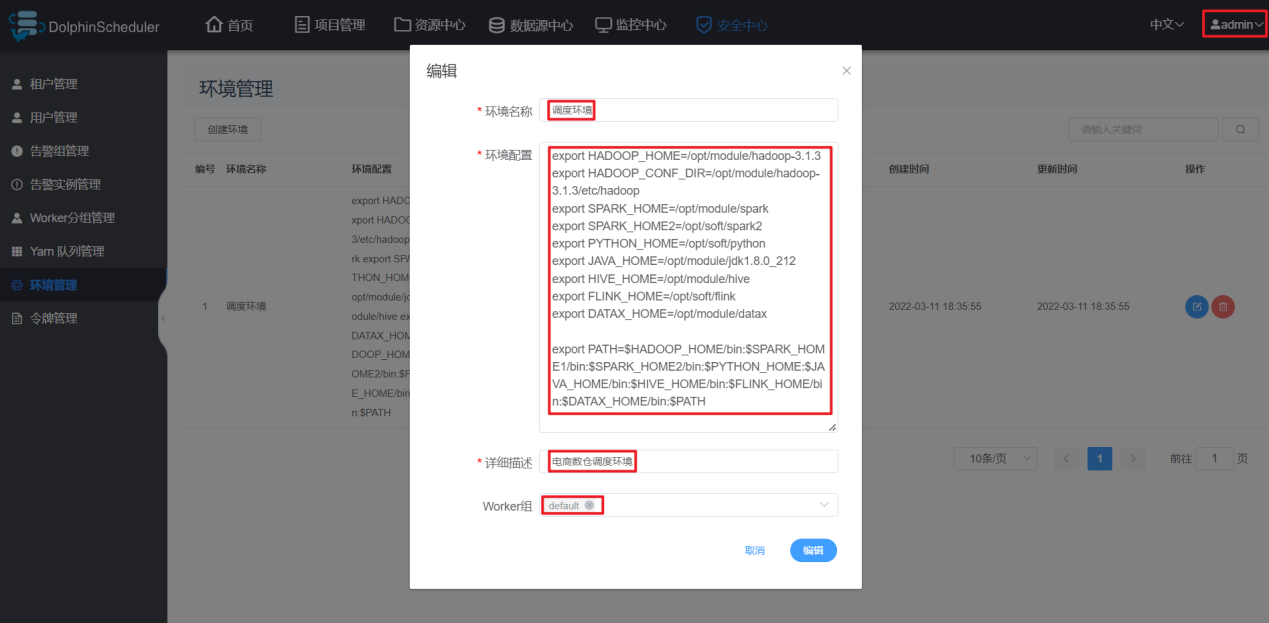
[atguigu@hadoop102 ~]$ xsync /opt/module/hive/

[atguigu@hadoop102 ~]$ xsync /opt/module/spark/

[atguigu@hadoop102 ~]$ xsync /opt/module/datax/

**5）配置运行环境**

（1）切换到 admin 用户，在环境管理下创建环境



（2）在环境配置中添加如下内容

标红部分根据集群情况修改。

**export HADOOP\_HOME=/opt/module/hadoop-3.1.3**

**export HADOOP\_CONF\_DIR=/opt/module/hadoop-3.1.3/etc/hadoop**

**export SPARK\_HOME=/opt/module/spark**

export SPARK\_HOME2=/opt/soft/spark2

export PYTHON\_HOME=/opt/soft/python

**export JAVA\_HOME=/opt/module/jdk1.8.0\_212**

**export HIVE\_HOME=/opt/module/hive**

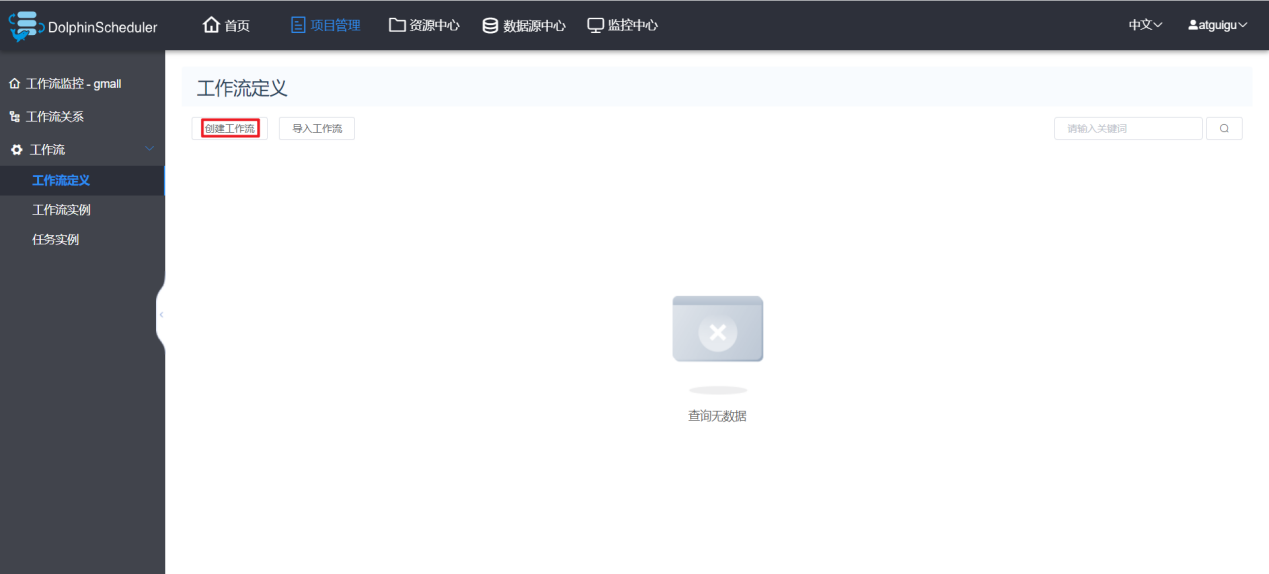
export FLINK\_HOME=/opt/soft/flink

**export DATAX\_HOME=/opt/module/datax**

export PATH=$HADOOP\_HOME/bin:$SPARK\_HOME1/bin:$SPARK\_HOME2/bin:$PYTHON\_HOME:$JAVA\_HOME/bin:$HIVE\_HOME/bin:$FLINK\_HOME/bin:$DATAX\_HOME/bin:$PATH

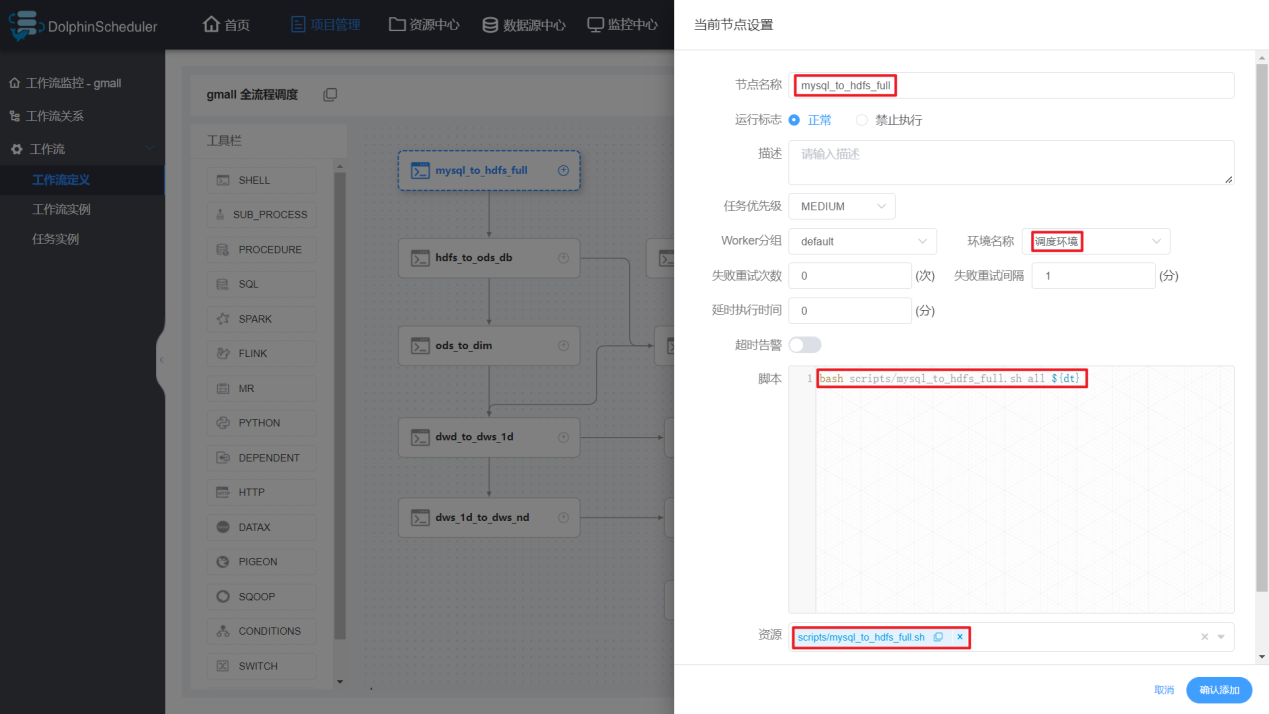
**6）创建工作流**

（1）在gmall项目下创建工作流

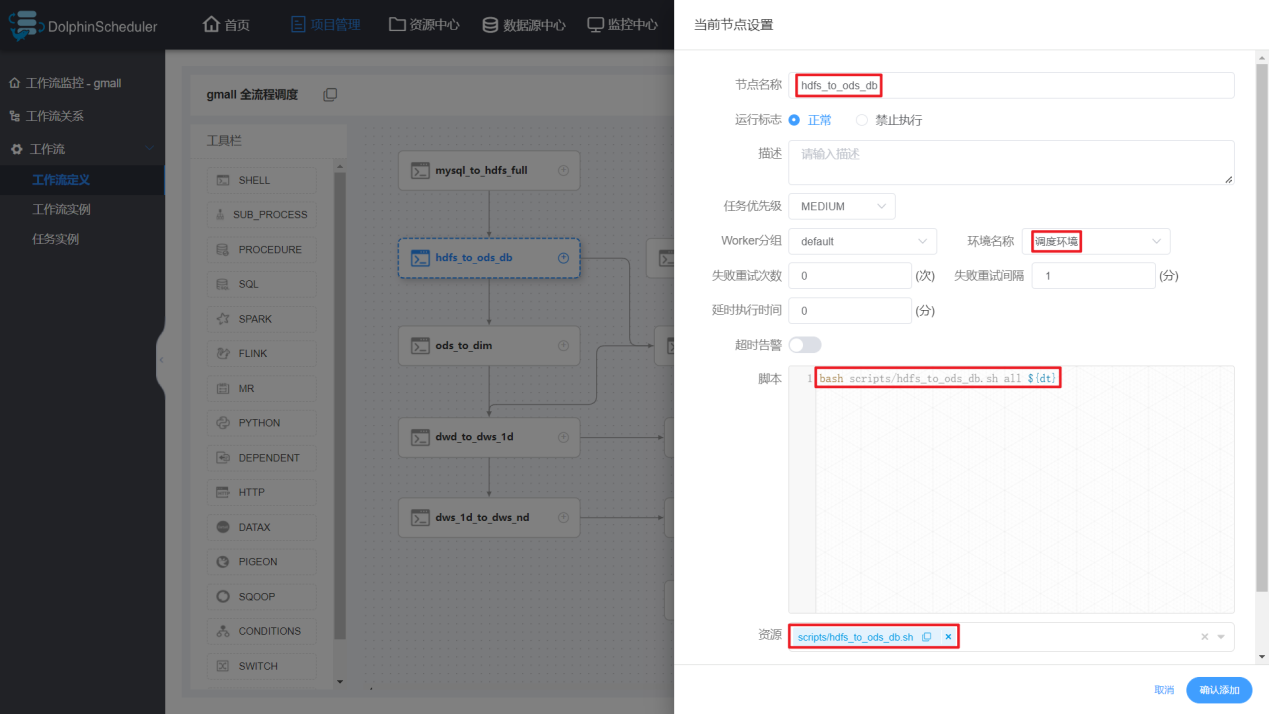


（2）各任务节点配置如下

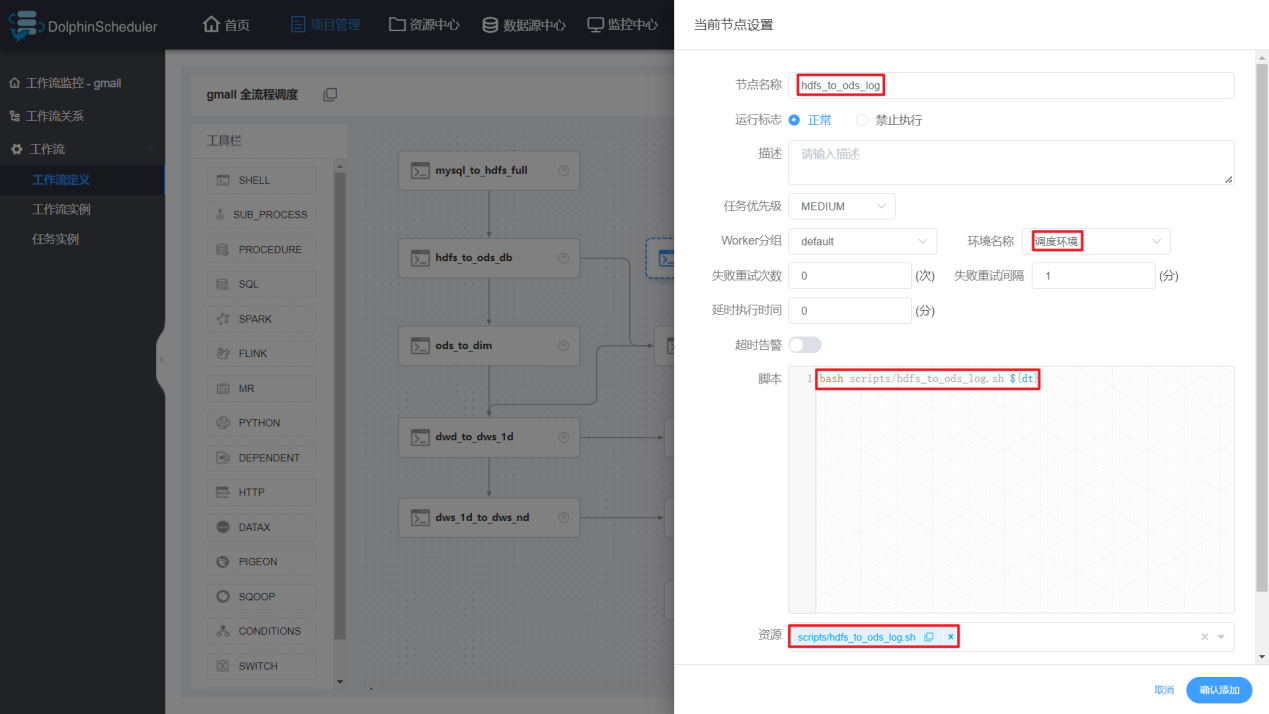
mysql\_to\_hdfs\_full



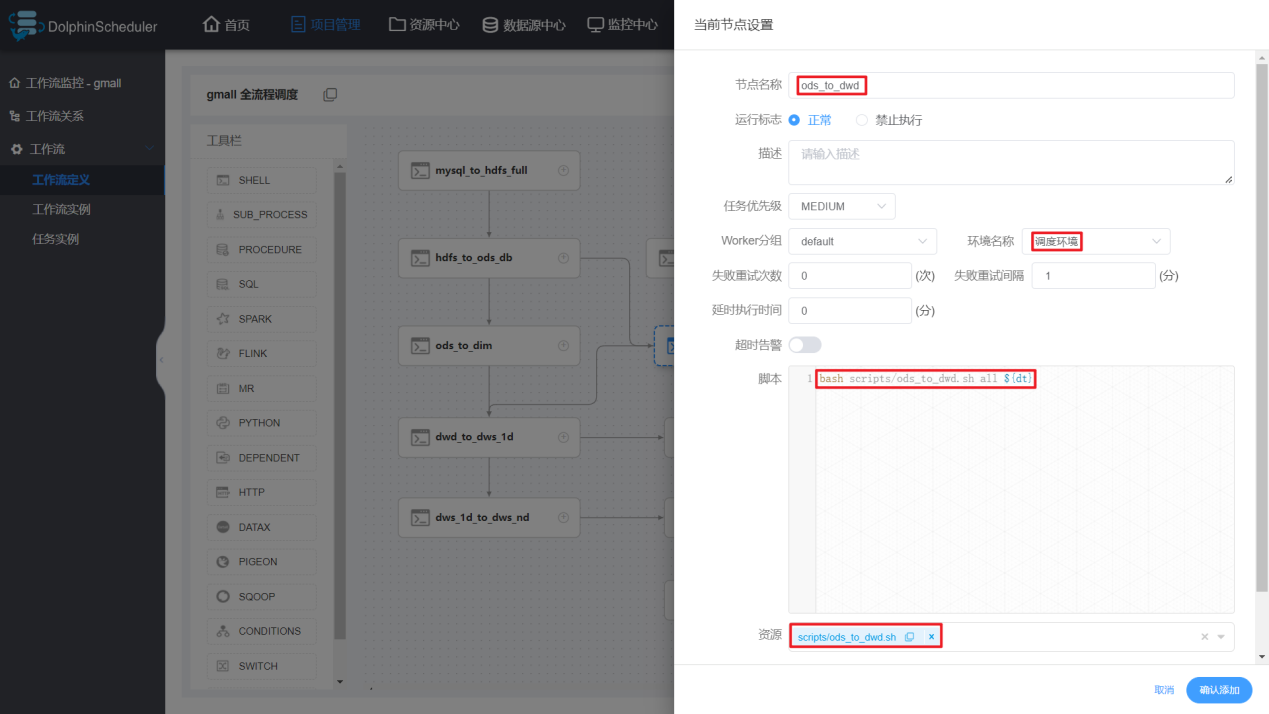
hdfs\_to\_ods\_db



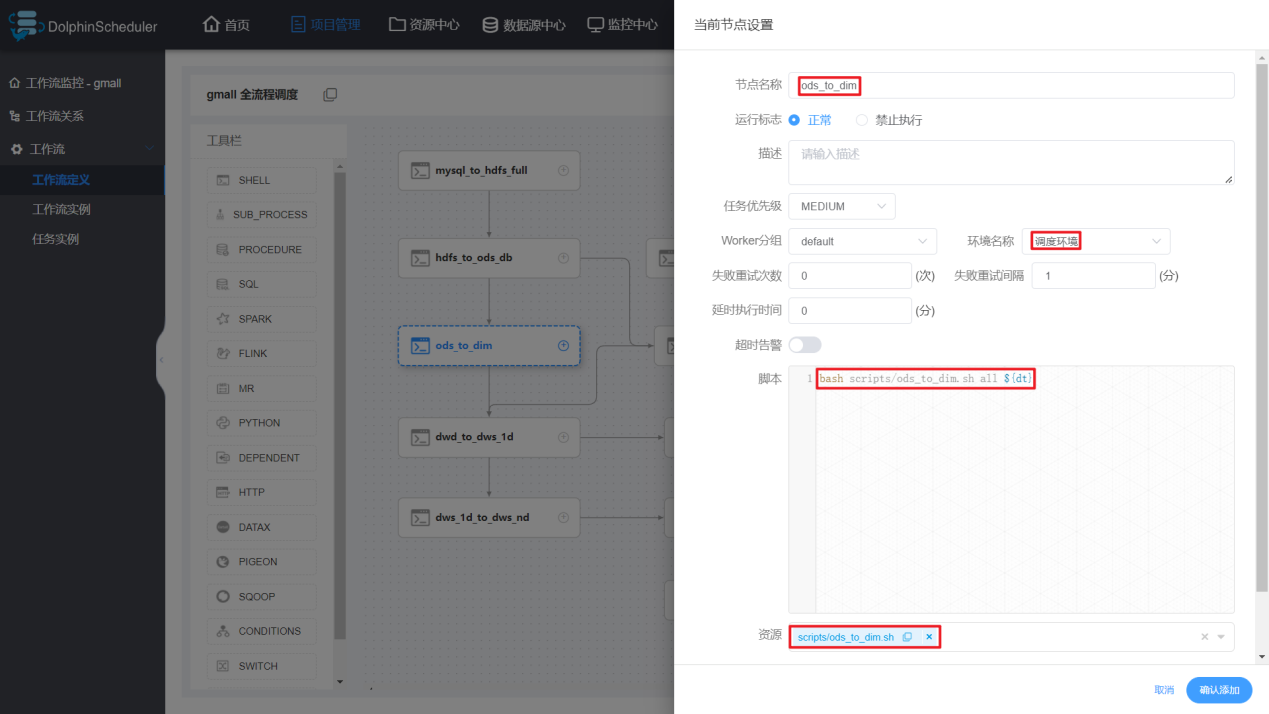
hdfs\_to\_ods\_log



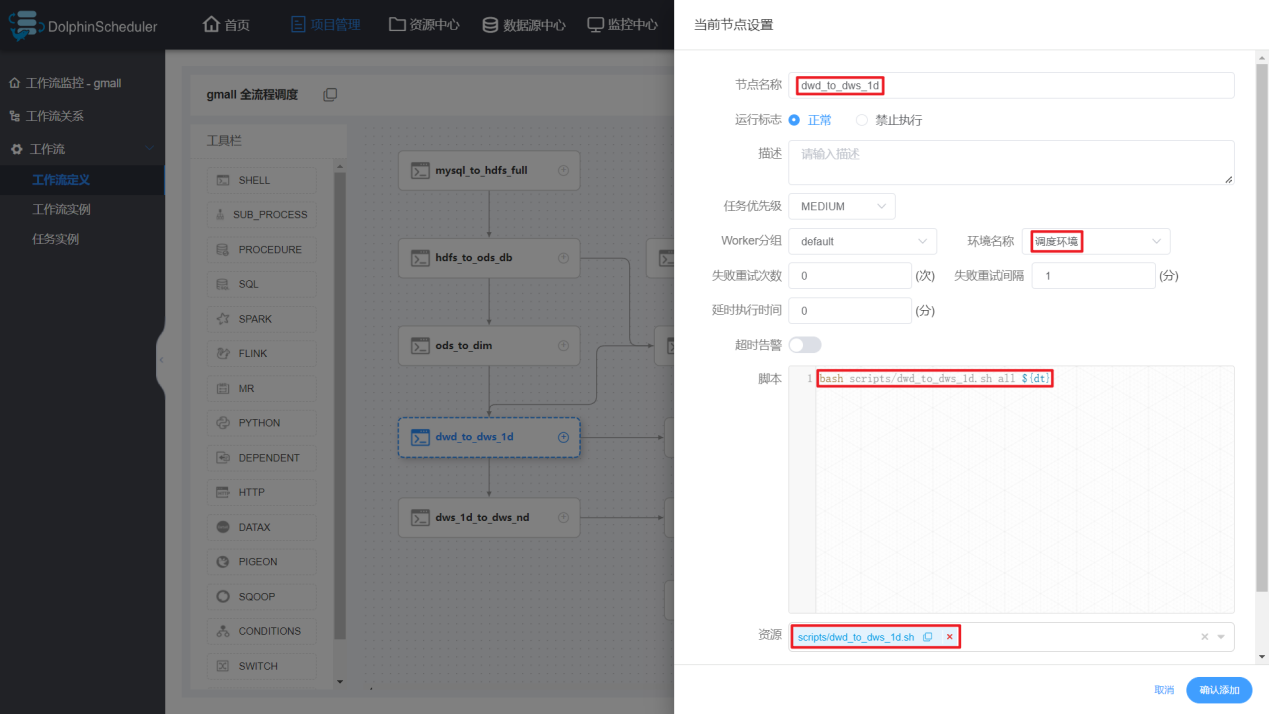
ods\_to\_dwd



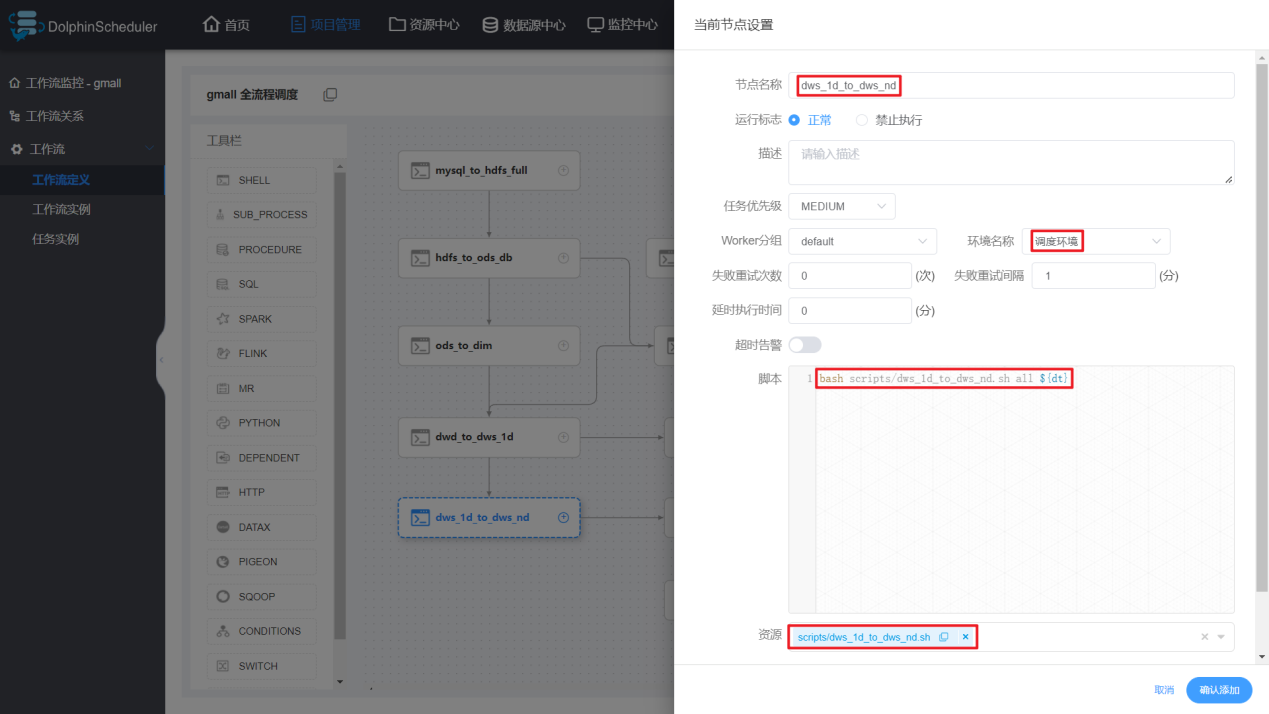
ods\_to\_dim



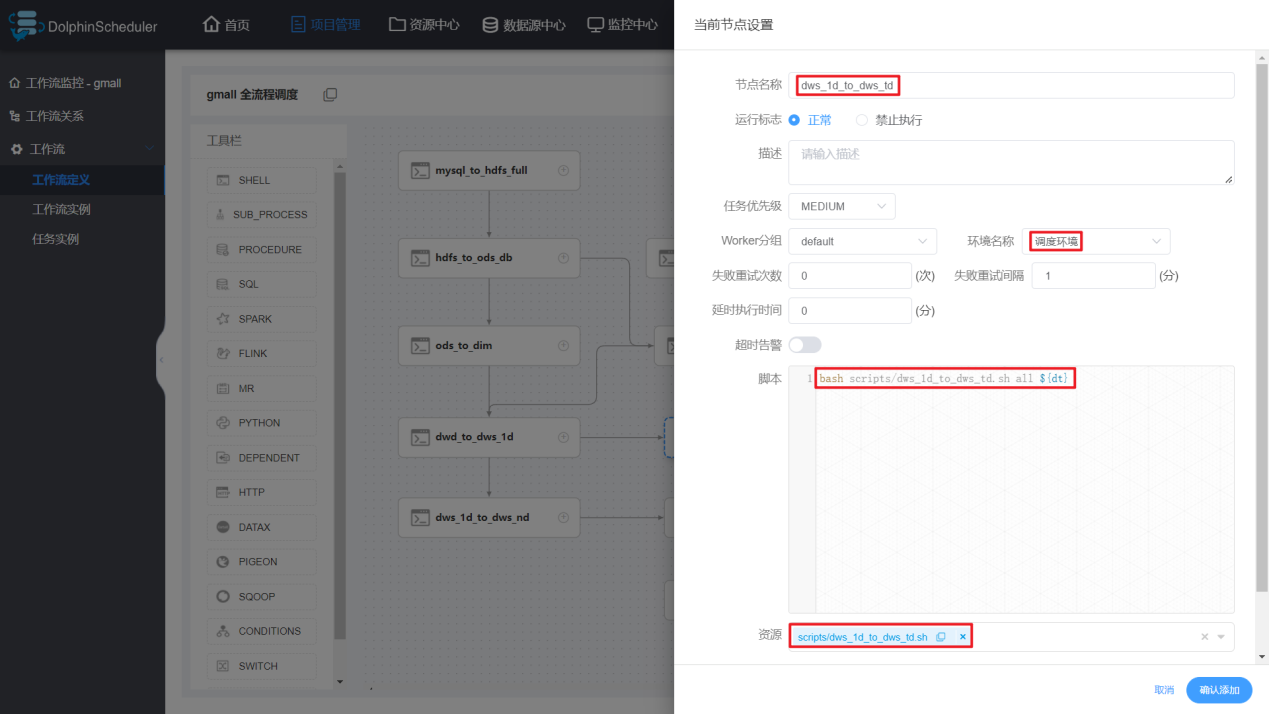
dwd\_to\_dws\_1d



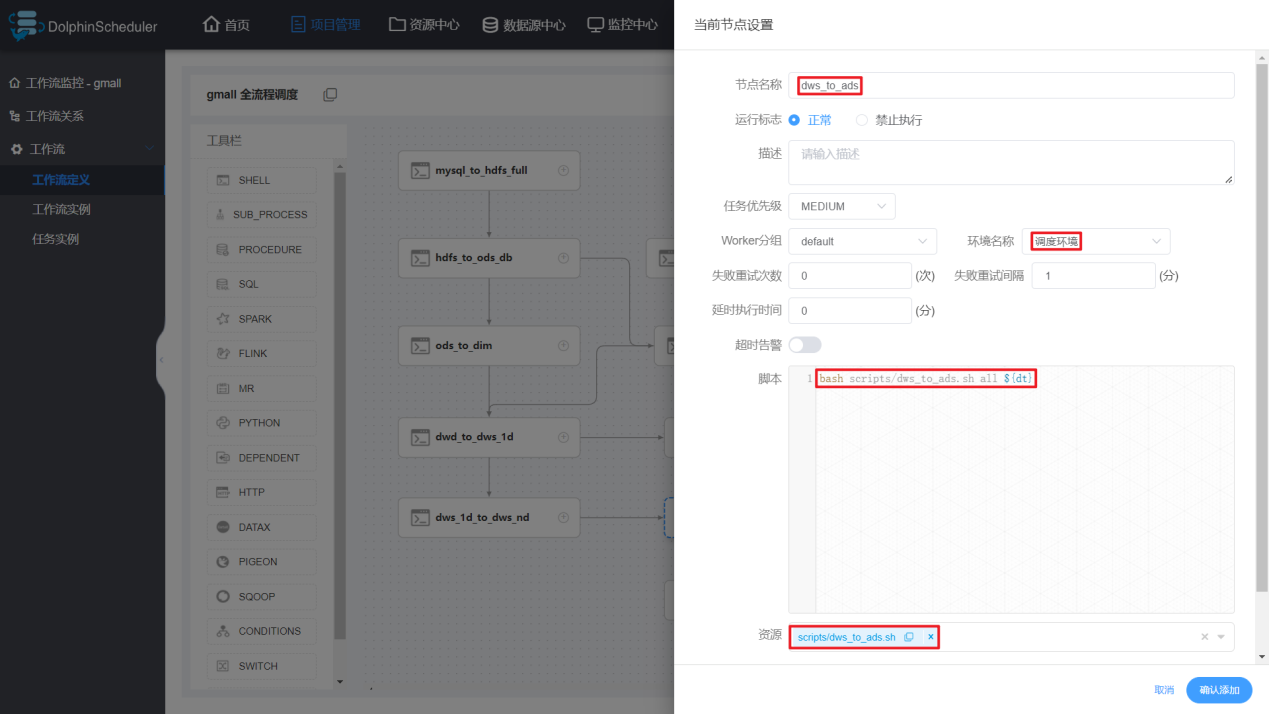
dws\_1d\_to\_dws\_nd



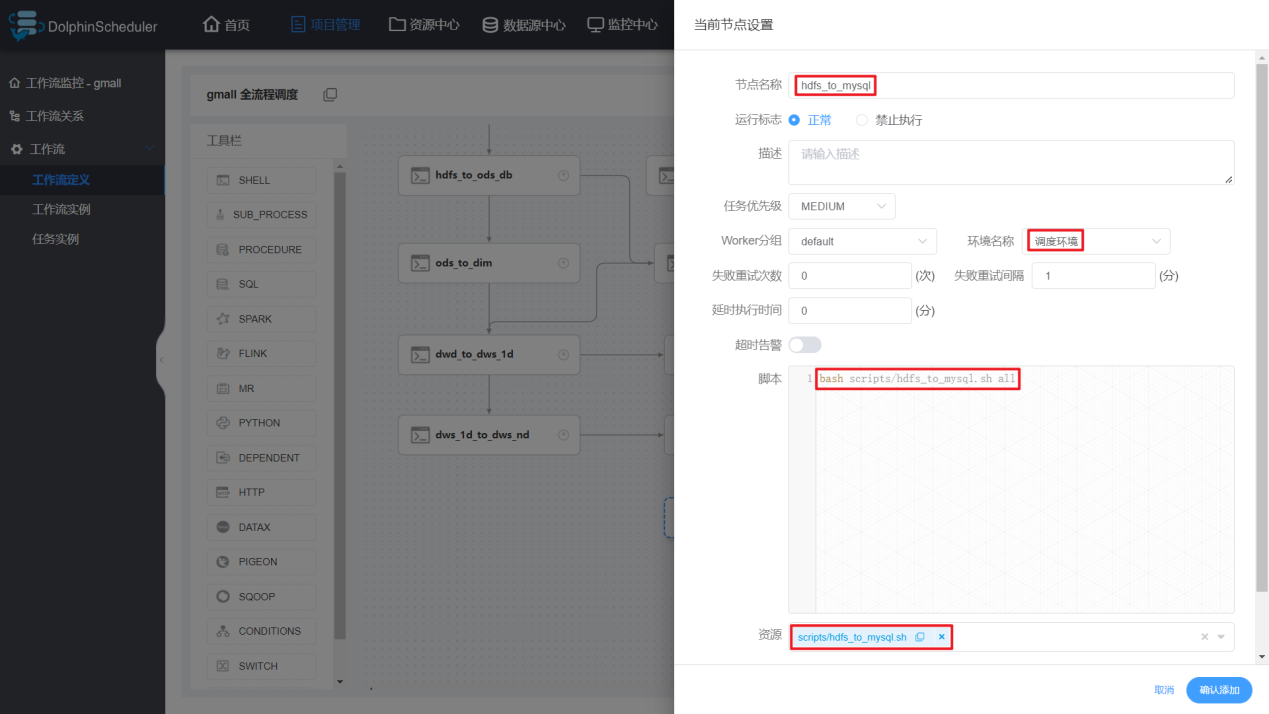
dws\_1d\_to\_dws\_td



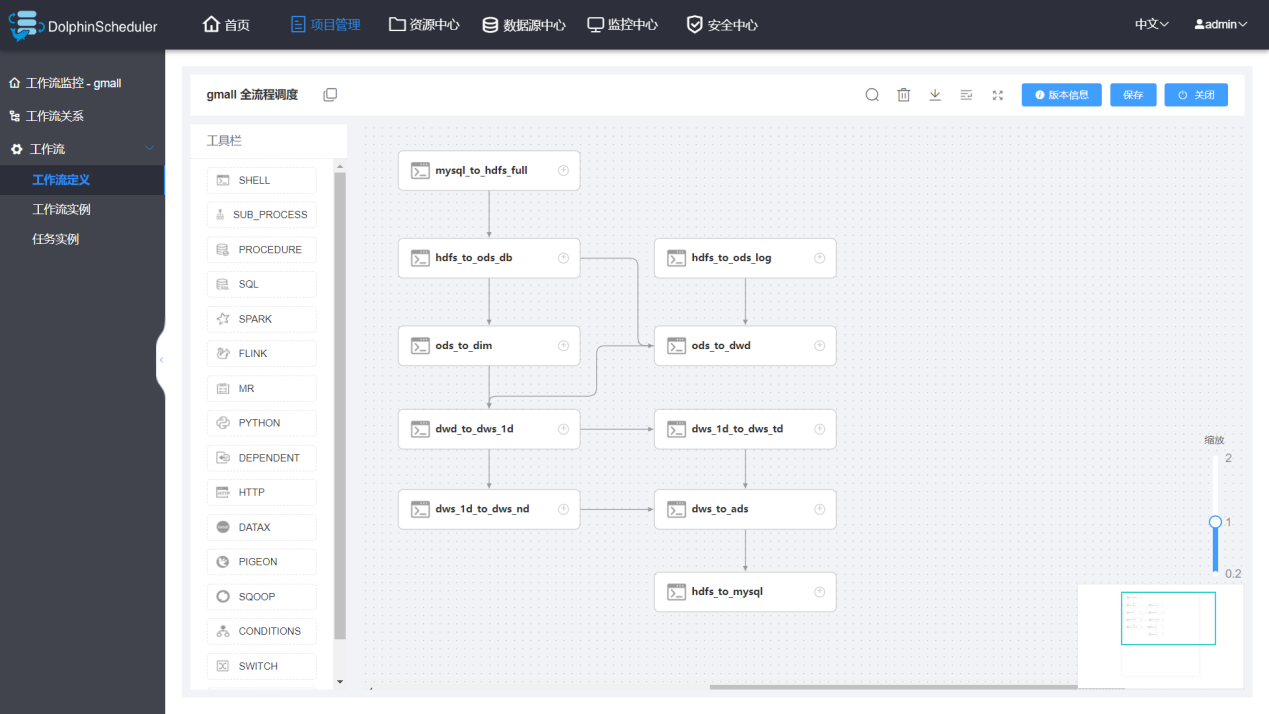
dws\_to\_ads



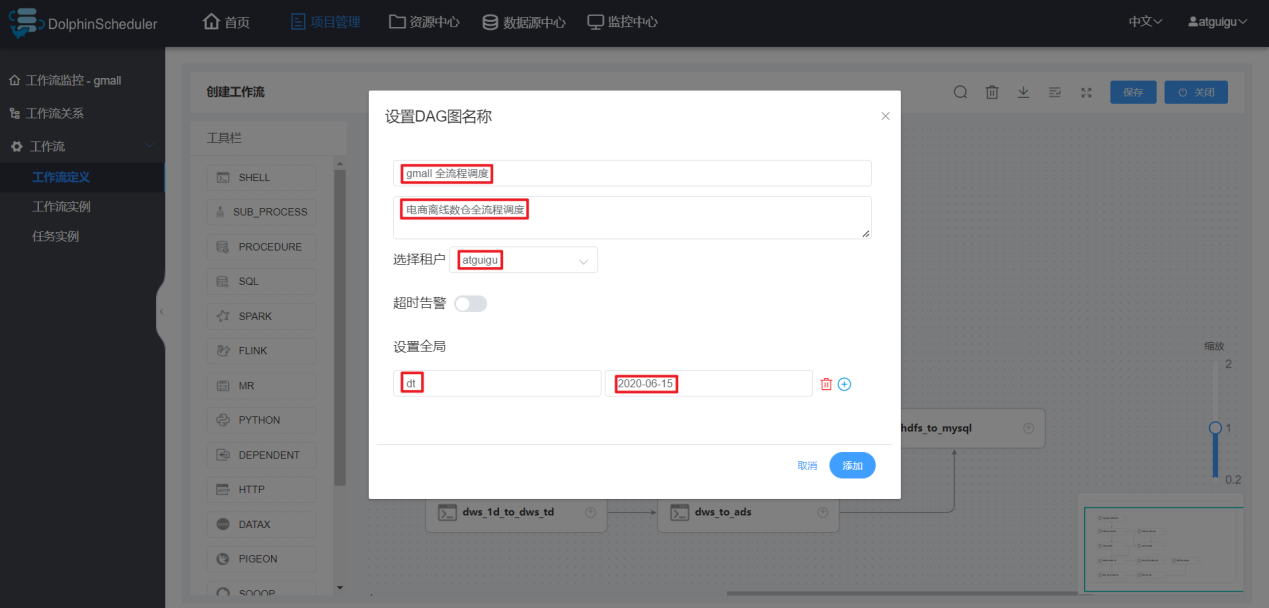
hdfs\_to\_mysql



（3）各节点依赖关系如下

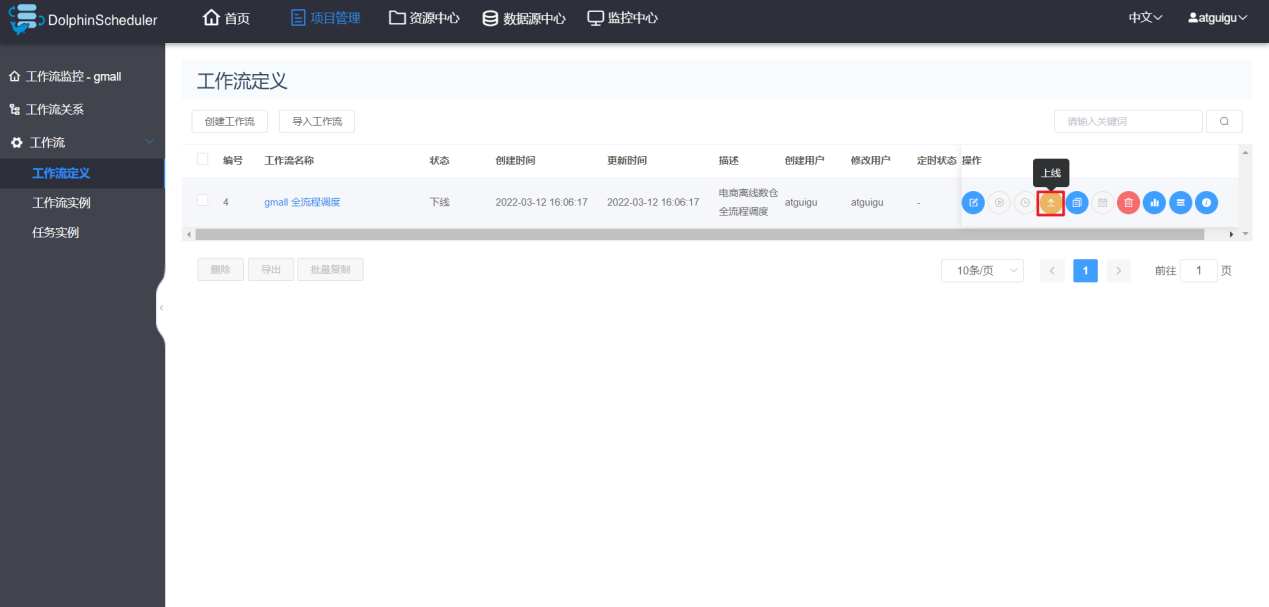


（4）保存工作流



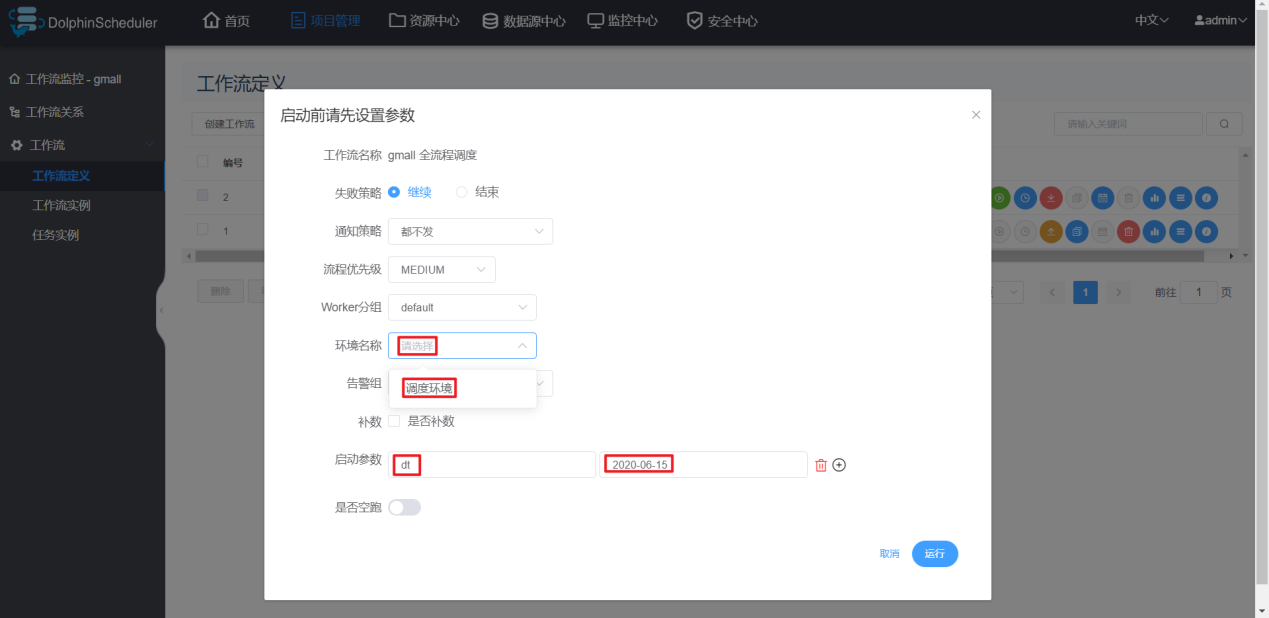
**注：定时调度时，全局参数值应设置为$[yyyy-MM-dd-1]或者空值。**

**7）上线工作流**



**8）执行工作流**





**注：工作流启动前选择的环境无效，必须在创建任务节点时选择对应环境。此处的“环境名称”可以置空。**

### 13.3.2 DolphinScheduler单机模式

**1）启动DolphinScheduler**

[atguigu@hadoop102 dolphinscheduler]$ bin/dolphinscheduler-daemon.sh start standalone-server

**2）安全中心配置**

由于DolphinScheduler的单机模式使用的是内置的ZK和数据库，故在集群模式下所做的相关配置在单机模式下并不可见，所以需要重新配置，必要的配置为创建租户和创建用户。

**（1）使用管理员用户登录**



**（2）创建租户**

图形用户界面

描述已自动生成

**（3）创建用户**

手机屏幕截图

描述已自动生成

**3）切换普通用户登录**

图形用户界面, 应用程序

描述已自动生成

**4）创建项目**

图形用户界面, 网站

描述已自动生成

**5）其余操作**

其余操作与集群模式基本一致，其中分发Hive、Spark、DataX这一步可以省略。