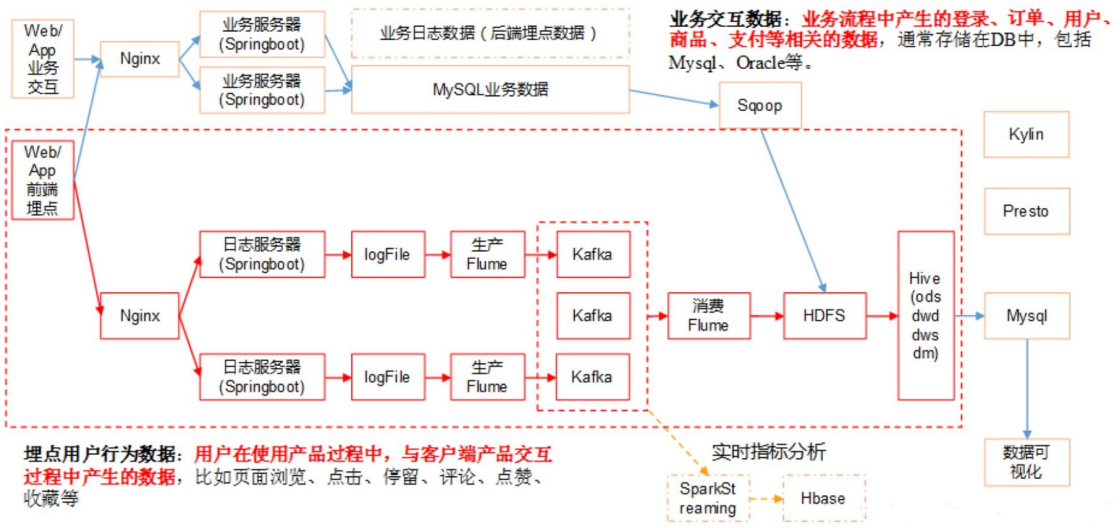
第3章离线数仓

1. 数仓搭建ODS、 DWD与DWS层



1.1 创建数据库

```
1、创建数据库的相关命令：
create database if not exists hdp_bw_ods_global;
create database if not exists hdp_bw_dwd_global;
create database if not exists hdp_bw_dws_global;
create database if not exists hdp_bw_ads_global;
create database if not exists hdp_bw_dim_global;
create database if not exists hdp_bw_tmp_global;

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

2、使用数据库(举例：hdp_bw_ods_global)
use hdp_bw_ods_global;
```

1.2 创建ods层表

1.2.1 运营数据库表-清单

序号	运营数据库表名	表中文名	ods层表名	表同步策略
1	bw_order_info	订单表	ods_bw_order_info_inc_1d	新增及变化
2	bw_order_detail	订单详情表	ods_bw_order_detail_inc_1d	增量
3	bw_payment_info	支付流水表	ods_bw_payment_info_1d	增量
4	bw_sku_info	商品表	ods_bw_sku_info_full_1d	全量
5	bw_user_info	用户表	ods_bw_user_info_full_1d	全量
6	bw_category1	商品一级分类表	ods_bw_category1_full_1d	全量
7	bw_category2	商品二级分类表	ods_bw_category2_full_1d	全量
8	bw_category3	商品三级分类表	ods_bw_category3_full_1d	全量

1.2.2 创建表

```
创建 ODS 层对应表
订单表
use hdp_bw_ods_global;
drop table if exists ods_bw_order_info_inc_1d;
create EXTERNAL table ods_bw_order_info_inc_1d (
`id` string COMMENT '订单编号',
`total_amount` decimal(10,2) COMMENT '订单金额',
`order_status` string COMMENT '订单状态',
`user_id` string COMMENT '用户id' ,
`payment_way` string COMMENT '支付方式',
`out_trade_no` string COMMENT '支付流水号',
`create_time` string COMMENT '创建时间',
`operate_time` string COMMENT '操作时间'
) COMMENT '订单表'
PARTITIONED BY ( `dt` string)
row format delimited fields terminated by '\t';

订单详情表
use hdp_bw_ods_global;
drop table if exists ods_bw_order_detail_inc_1d;
create EXTERNAL table ods_bw_order_detail_inc_1d(
`id` string COMMENT '订单编号',
`order_id` string COMMENT '订单号',
`user_id` string COMMENT '用户id' ,
`sku_id` string COMMENT '商品id',
`sku_name` string COMMENT '商品名称',
`order_price` string COMMENT '下单价格',
`sku_num` string COMMENT '商品数量',
`create_time` string COMMENT '创建时间'
) COMMENT '订单明细表'
PARTITIONED BY ( `dt` string)
row format delimited fields terminated by '\t';
```

```
支付流水表
use hdp_bw_ods_global;
drop table if exists ods_bw_payment_info_inc_1d;
create EXTERNAL table ods_bw_payment_info_inc_1d(
`id` bigint COMMENT '编号',
`out_trade_no` string COMMENT '对外业务编号',
`order_id` string COMMENT '订单编号',
`user_id` string COMMENT '用户编号',
`alipay_trade_no` string COMMENT '支付宝交易流水编号',
`total_amount` decimal(16,2) COMMENT '支付金额',
`subject` string COMMENT '交易内容',
`payment_type` string COMMENT '支付类型',
`payment_time` string COMMENT '支付时间'
) COMMENT '支付流水表'
PARTITIONED BY ( `dt` string)
row format delimited fields terminated by '\t';

商品表
use hdp_bw_ods_global;
drop table if exists ods_bw_sku_info_full_1d;
create EXTERNAL table ods_bw_sku_info_full_1d(
`id` string COMMENT 'skuId',
`spu_id` string COMMENT 'spuid',
`price` decimal(10,2) COMMENT '价格' ,
`sku_name` string COMMENT '商品名称',
`sku_desc` string COMMENT '商品描述',
`weight` string COMMENT '重量',
`tm_id` string COMMENT '品牌id',
`category3_id` string COMMENT '品类id',
`create_time` string COMMENT '创建时间'
) COMMENT '商品表'
PARTITIONED BY ( `dt` string)
row format delimited fields terminated by '\t';

用户表
use hdp_bw_ods_global;
drop table if exists ods_bw_user_info_full_1d;
create EXTERNAL table ods_bw_user_info_full_1d(
`id` string COMMENT '用户id',
`name` string COMMENT '姓名',
`birthday` string COMMENT '生日',
`gender` string COMMENT '性别',
`email` string COMMENT '邮箱',
`user_level` string COMMENT '用户等级',
`create_time` string COMMENT '创建时间'
) COMMENT '用户信息'
PARTITIONED BY ( `dt` string)
row format delimited fields terminated by '\t';

商品一级分类表
use hdp_bw_ods_global;
```

```
drop table if exists ods_bw_category1_full_1d;
create EXTERNAL table ods_bw_category1_full_1d(
`id` string COMMENT 'id',
`name` string COMMENT '名称'
) COMMENT '商品一级分类'
PARTITIONED BY ( `dt` string)
row format delimited fields terminated by '\t';

商品二级分类表
use hdp_bw_ods_global;
drop table if exists ods_bw_category2_full_1d;
create EXTERNAL table ods_bw_category2_full_1d(
`id` string COMMENT ' id',
`name` string COMMENT '名称',
category1_id string COMMENT '一级品类id'
) COMMENT '商品二级分类'
PARTITIONED BY ( `dt` string)
row format delimited fields terminated by '\t';

商品三级分类表
use hdp_bw_ods_global;
drop table if exists ods_bw_category3_full_1d;
create EXTERNAL table ods_bw_category3_full_1d(
`id` string COMMENT ' id',
`name` string COMMENT '名称',
category2_id string COMMENT '二级品类id'
) COMMENT '商品三级分类'
PARTITIONED BY ( `dt` string)
row format delimited fields terminated by '\t';
```

1.2.3 行为埋点日志-清单

- 原始数据层，存放原始数据，直接加载原始日志、数据，数据保持原貌不做处理。

序号	埋点日志类型	表中文名	ods层表名	更新方式
1	用户启动日志	启动日志表	ods_log_start_app_inc_1d	增量
2	用户行为日志	事件日志表	ods_log_event_inc_1d	增量

1.2.4 创建表

```
启动日志表
use hdp_bw_ods_global;
drop table if exists ods_log_start_app_inc_1d;
create EXTERNAL table ods_log_start_app_inc_1d(
`line` string COMMENT '日志行'
) COMMENT '启动日志表'
PARTITIONED BY ( `dt` string)
row format delimited fields terminated by '\t';
```

```
事件日志表
use hdp_bw_ods_global;
drop table if exists ods_log_event_inc_1d;
create EXTERNAL table ods_log_event_inc_1d(
`line` string COMMENT '日志行'
) COMMENT '事件日志表'
PARTITIONED BY ( `dt` string)
row format delimited fields terminated by '\t';


关于这10张表还有两个细节:
1、表的数据存储格式:  ods层, textfile,  补充一点:  dwd dws parquet+snappy
2、所有的表都是 external 表
```

1.2.5 脚本导入ODS层数据

```
1、创建 load_data.sh
vim load_data.sh

在脚本中填写如下内容
#!/bin/bash
do_date=$1
dbname=hdp_bw_ods_global
hive=/root/install/hive2.3.6/bin/hive

sql="load data local inpath '/root/hdp/dw_project/data/bw_order_info/$do_date'
OVERWRITE into table $dbname.ods_bw_order_info_inc_1d partition(dt='$do_date');

load data local inpath '/root/hdp/dw_project/data/bw_order_detail/$do_date'
OVERWRITE into table $dbname.ods_bw_order_detail_inc_1d partition(dt='$do_date');

load data local inpath '/root/hdp/dw_project/data/bw_payment_info/$do_date'
OVERWRITE into table $dbname.ods_bw_payment_info_inc_1d partition(dt='$do_date');

load data local inpath '/root/hdp/dw_project/data/bw_sku_info/$do_date' OVERWRITE
into table $dbname.ods_bw_sku_info_full_1d partition(dt='$do_date');

load data local inpath '/root/hdp/dw_project/data/bw_user_info/$do_date'
OVERWRITE into table $dbname.ods_bw_user_info_full_1d partition(dt='$do_date');

load data local inpath '/root/hdp/dw_project/data/bw_category1/$do_date'
OVERWRITE into table $dbname.ods_bw_category1_full_1d partition(dt='$do_date');

load data local inpath '/root/hdp/dw_project/data/bw_category2/$do_date'
OVERWRITE into table $dbname.ods_bw_category2_full_1d partition(dt='$do_date');

load data local inpath '/root/hdp/dw_project/data/bw_category3/$do_date'
OVERWRITE into table $dbname.ods_bw_category3_full_1d partition(dt='$do_date');
```

```
load data local inpath '/root/hdp/dw_project/data/start_app/$do_date' OVERWRITE
into table $dbname.ods_log_start_app_inc_1d partition(dt='$do_date');

load data local inpath '/root/hdp/dw_project/data/event/$do_date' OVERWRITE into
table $dbname.ods_log_event_inc_1d partition(dt='$do_date');"

$hive -e "$sql"

2、添加脚本执行权限
chmod u+x load_data.sh

3、执行脚本导入数据
sh load_data.sh 2021-04-20
sh load_data.sh 2020-04-21

4、检测数据
select * from ods_bw_user_info_full_1d limit 3;
select * from ods_bw_sku_info_full_1d limit 3;
select * from ods_bw_order_info_inc_1d limit 3;
select * from ods_bw_order_detail_inc_1d limit 3;
select * from ods_bw_payment_info_inc_1d limit 3;
select * from ods_bw_category1_full_1d limit 3;
select * from ods_bw_category2_full_1d limit 3;
select * from ods_bw_category3_full_1d limit 3;
select * from ods_log_start_app_inc_1d limit 3;
select * from ods_log_event_inc_1d limit 3;

Hive 检查:
desc formatted hdp_bw_ods_global.ods_bw_sku_info_full_1d;

5、检查数据：三张商品分类表的join查询
set hive.cli.print.header=true;
use hdp_bw_ods_global;
select c1.id,
c1.name,
c2.id,
c2.name,
c3.id,
c3.name
from ods_bw_category3_full_1d c3
join ods_bw_category2_full_1d c2 on (c3.category2_id=c2.id)
join ods_bw_category1_full_1d c1 on (c2.category1_id=c1.id)
where c1.dt='2021-05-01' and c2.dt='2021-05-01' and c3.dt='2021-05-01'
limit 100;
```

1.3 DWD层数据解析

- 对 ODS 层数据进行清洗（去除空值，脏数据，超过极限范围的数据，行式存储改为列存储，改压缩格式，维度退化等），具体还需看自己的数据情况。

1.3.1 创建DWD表

```
创建订单表
use hdp_bw_dwd_global;
drop table if exists dwd_mysql_order_info_inc_1d;
create external table dwd_mysql_order_info_inc_1d (
`id` string COMMENT '订单编号',
`total_amount` decimal(10,2) COMMENT '订单金额',
`order_status` string COMMENT '订单状态',
`user_id` string COMMENT '用户id' ,
`payment_way` string COMMENT '支付方式',
`out_trade_no` string COMMENT '支付流水号',
`create_time` string COMMENT '创建时间',
`operate_time` string COMMENT '操作时间'
) COMMENT '订单表'
PARTITIONED BY ( `dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");

创建订单详情表
use hdp_bw_dwd_global;
drop table if exists dwd_mysql_order_detail_inc_1d;
create EXTERNAL table dwd_mysql_order_detail_inc_1d(
`id` string COMMENT '订单编号',
`order_id` string COMMENT '订单号',
`user_id` string COMMENT '用户id' ,
`sku_id` string COMMENT '商品id',
`sku_name` string COMMENT '商品名称',
`order_price` string COMMENT '下单价格',
`sku_num` string COMMENT '商品数量',
`create_time` string COMMENT '创建时间'
) COMMENT '订单明细表'
PARTITIONED BY ( `dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");

创建支付流水表
use hdp_bw_dwd_global;
drop table if exists dwd_mysql_payment_info_inc_1d;
create EXTERNAL table dwd_mysql_payment_info_inc_1d(
`id` bigint COMMENT '编号',
`out_trade_no` string COMMENT '对外业务编号',
`order_id` string COMMENT '订单编号',
`user_id` string COMMENT '用户编号',
`alipay_trade_no` string COMMENT '支付宝交易流水编号',
`total_amount` decimal(16,2) COMMENT '支付金额',
`subject` string COMMENT '交易内容',
`payment_type` string COMMENT '支付类型',
`payment_time` string COMMENT '支付时间'
) COMMENT '支付流水表'
PARTITIONED BY ( `dt` string)
```

```
stored as parquet
tblproperties ("parquet.compression"="snappy");

创建用户表(维度表)
use hdp_bw_dwd_global;
drop table if exists dwd_mysql_user_info_full_1d;
create EXTERNAL table dwd_mysql_user_info_full_1d(
`id` string COMMENT '用户id',
`name` string COMMENT '姓名',
`birthday` string COMMENT '生日',
`gender` string COMMENT '性别',
`email` string COMMENT '邮箱',
`user_level` string COMMENT '用户等级',
`create_time` string COMMENT '创建时间'
) COMMENT '用户信息'
PARTITIONED BY ( `dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");

创建商品表（加分类，降维）
use hdp_bw_dwd_global;
drop table if exists dwd_mysql_sku_info_full_1d;
create external table dwd_mysql_sku_info_full_1d(
`id` string COMMENT 'skuId',
`spu_id` string COMMENT 'spuid',
`price` decimal(10,2) COMMENT '价格',
`sku_name` string COMMENT '商品名称',
`sku_desc` string COMMENT '商品描述',
`weight` string COMMENT '重量',
`tm_id` string COMMENT '品牌id',
`category3_id` string COMMENT '3级分类id',
`category2_id` string COMMENT '2级分类id',
`category1_id` string COMMENT '1级分类id',
`category3_name` string COMMENT '3级分类名称',
`category2_name` string COMMENT '2级分类名称',
`category1_name` string COMMENT '1级分类名称',
`create_time` string COMMENT ''
) COMMENT '商品表'
PARTITIONED BY ( `dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");


创建启动日志基础明细表
use hdp_bw_dwd_global;
drop table if exists dwd_log_base_start_inc_1d;
CREATE EXTERNAL TABLE dwd_log_base_start_inc_1d (
`mid_id` string comment '移动互联网设备id',
`user_id` string comment '用户id',
`version_code` string comment '版本code',
`version_name` string comment '版本名称',
`lang` string comment '系统语言',
```

```
`source` string comment '渠道',
`os` string comment '操作系统版本',
`area` string comment '区域',
`model` string comment '型号',
`brand` string comment '品牌',
`sdk_version` string comment 'sdk版本',
`gmail` string comment '邮箱',
`height_width` string comment '',
`app_time` string comment '客户端日志产生时间',
`network` string comment '网络类型',
`lng` string comment '经度',
`lat` string comment '纬度',
`event_name` string comment '事件名',
`event_json` string comment '事件详细信息',
`server_time` string comment '日志服务端时间')
PARTITIONED BY (`dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");

创建事件日志基础明细表
use hdp_bw_dwd_global;
drop table if exists dwd_log_base_event_inc_1d;
CREATE EXTERNAL TABLE dwd_log_base_event_inc_1d(
`mid_id` string comment '移动互联网设备id',
`user_id` string comment '用户id',
`version_code` string comment '版本code',
`version_name` string comment '版本名称',
`lang` string comment '系统语言',
`source` string comment '终端',
`os` string comment '操作系统版本',
`area` string comment '区域',
`model` string comment '型号',
`brand` string comment '品牌',
`sdk_version` string comment 'sdk版本',
`gmail` string comment '邮箱',
`height_width` string comment '',
`app_time` string comment '客户端时间',
`network` string comment '网络类型',
`lng` string comment '经度',
`lat` string comment '纬度',
`event_name` string comment '事件名',
`event_json` string comment '事件详细信息',
`server_time` string comment '日志服务端时间')
PARTITIONED BY (`dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");
```

1.3.2 DWD层数据ETL

- 补充知识点：

```
```

```
#日志数据表
#hive中的 get_json_object()和json_tuple()
line
{"movie":"1412","rate":"5","timeStamp":"51231241145","uid":"1"}
{"movie":"5123","rate":"4","timeStamp":"51231241231","uid":"1"}
{"movie":"2512","rate":"3","timeStamp":"51265581245","uid":"1"}
{"movie":"5611","rate":"2","timeStamp":"51215864145","uid":"1"}


select get_json_object(line,‘$movie’) as movie,
    get_json_object(line,'$.rate') as rate,
    get_json_object(line,'$.timeStamp') as time,
    get_json_object(line,'$.uid') as uid
from rate_json;
1412 5 51231241145 1
5123 4 51231241145 1
2512 3 51231241145 1
5611 2 51231241145 1

json_tuple()
{"movie":"1412","rate":"5","timeStamp":"51231241145","uid":"1"}
{"movie":"5123","rate":"4","timeStamp":"51231241231","uid":"1"}
{"movie":"2512","rate":"3","timeStamp":"51265581245","uid":"1"}
{"movie":"5611","rate":"2","timeStamp":"51215864145","uid":"1"}

select b.movie,b.rate,b.time,b.uid
from rate_json
lateral view json_tuple(line,'movie','rate','timeStamp','uid') b
as movie,rate,time,uid;

1412 5 51231241145 1
5123 4 51231241145 1
2512 3 51231241145 1
5611 2 51231241145 1
```

- 数据变换脚本

```
第1步：创建：etl_dwd_data.sh

第2步：在脚本中填写如下内容

#!/bin/bash
ods_dbname=hdp_bw_ods_global
dwd_dbname=hdp_bw_dwd_global
hive=/root/install/apache-hive-2.3.6-bin/bin/hive

if [ -n $1 ] ;then
    log_date=$1
else
    log_date=`date -d "-1 day" +%F`
```

```
fi

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

insert overwrite table "$dwd_dbname".dwd_mysql_order_info_inc_1d partition(dt)
select * from "$ods_dbname".ods_bw_order_info_inc_1d
where dt='$log_date' and id is not null;

insert overwrite table "$dwd_dbname".dwd_mysql_order_detail_inc_1d partition(dt)
select * from "$ods_dbname".ods_bw_order_detail_inc_1d
where dt='$log_date' and id is not null;

insert overwrite table "$dwd_dbname".dwd_mysql_payment_info_inc_1d partition(dt)
select * from "$ods_dbname".ods_bw_payment_info_inc_1d
where dt='$log_date' and id is not null;

insert overwrite table "$dwd_dbname".dwd_mysql_user_info_full_1d partition(dt)
select * from "$ods_dbname".ods_bw_user_info_full_1d
where dt='$log_date' and id is not null;

insert overwrite table "$dwd_dbname".dwd_mysql_sku_info_full_1d partition(dt)
select
sku.id,
sku.spu_id,
sku.price,
sku.sku_name,
sku.sku_desc,
sku.weight,
sku.tm_id,
sku.category3_id,
c2.id category2_id ,
c1.id category1_id,
c3.name category3_name,
c2.name category2_name,
c1.name category1_name,
sku.create_time,
sku.dt
from
"$ods_dbname".ods_bw_sku_info_full_1d sku
join "$ods_dbname".ods_bw_category3_full_1d c3 on sku.category3_id=c3.id
join "$ods_dbname".ods_bw_category2_full_1d c2 on c3.category2_id=c2.id
join "$ods_dbname".ods_bw_category1_full_1d c1 on c2.category1_id=c1.id
where sku.dt='$log_date' and c2.dt='$log_date'
and c3.dt='$log_date' and c1.dt='$log_date'
and sku.id is not null;

insert overwrite table "$dwd_dbname".dwd_log_base_start_inc_1d PARTITION (dt)
select
mid_id,
user_id,
version_code,
```

```
version_name,
lang,
source ,
os ,
area ,
model ,
brand ,
sdk_version ,
gmail ,
height_width ,
app_time ,
network ,
lng ,
lat ,
event_name ,
event_json ,
server_time ,
dt
from "$ods_dbname".ods_log_start_app_inc_1d lateral view
json_tuple(line,'mid','uid','vc','vn','l','sr','os','ar','md','ba','sv','g','hw'
,'t','nw','ln','la','en','kv','ett') tmp_k as
mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_ver
sion,gmail,height_width,app_time,network,lng,lat,event_name,
event_json,server_time
where dt='$log_date' and event_name <> '' and server_time > 0 and mid_id <> '';

#事件数据表
insert overwrite table "$dwd_dbname".dwd_log_base_event_inc_1d PARTITION (dt)
select
mid_id,
user_id,
version_code,
version_name,
lang,
source ,
os ,
area ,
model ,
brand ,
sdk_version ,
gmail ,
height_width ,
app_time ,
network ,
lng ,
lat ,
event_name ,
event_json ,
server_time,
dt
from "$ods_dbname".ods_log_event_inc_1d lateral view
json_tuple(line,'mid','uid','vc','vn','l','sr','os','ar','md','ba','sv','g','hw'
```

```
,'t','nw','ln','la','en','kv','ett') tmp_k as
mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_ver
sion,gmail,height_width,app_time,network,lng,lat,event_name,event_json,server_tim
e where dt='$log_date' and event_name <> '' and server_time > 0 and mid_id <> '';
"

$hive -e "$sql"

添加脚本执行权限
chmod u+x etl_dwd_data.sh

执行脚本清洗数据

sh etl_dwd_data.sh 2021-04-21

检查数据:
select * from dwd_log_base_event_inc_1d limit 3;
select * from dwd_log_base_start_inc_1d limit 3;
select * from dwd_mysql_order_detail_inc_1d limit 3;
select * from dwd_mysql_order_info_inc_1d limit 3;
select * from dwd_mysql_payment_info_inc_1d limit 3;
select * from dwd_mysql_sku_info_full_1d limit 3;
select * from dwd_mysql_user_info_full_1d limit 3;

总结:
DWD:  ODS中的表数据进行ETL处理得来的
DWS:  基于DWD做了一个轻度汇总
```

1.4 DWS层用户行为宽表

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

```
我们期望，这张表甚至包含所有的信息
明细宽表的构建是违反范式的，利用数据的冗余来提高查询分析的效率!
1、商品表 join 三张 商品分类表
2、先提前去构建这张明细宽表:  做任何的分析，都是基于这张宽表

构建非常详细的订单明细宽表:订单信息，用户信息，商品信息，支付信息
```

1.4.1 创建用户行为宽表

```
use hdp_bw_dws_global;
drop table if exists dws_traffic_ub_inc_1d;
create external table dws_traffic_ub_inc_1d (
user_id string comment '用户 id',
order_count bigint comment '下单次数',
order_amount decimal(16,2) comment '下单金额',
payment_count bigint comment '支付次数',
payment_amount decimal(16,2) comment '支付金额',
comment_count bigint comment '评论次数'
) COMMENT '每日用户行为宽表'
PARTITIONED BY (`dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");
```

1.4.2 生成每日数据

```
第1步：创建 etl_dws_data.sh
vi etl_dws_data.sh

第2步：在脚本中填写如下内容
#!/bin/bash
dwd_dbname=hdp_bw_dwd_global
dws_dbname=hdp_bw_dws_global
if [ -n $1 ] ;then
    log_date=$1
else
    log_date=`date -d "-1 day" +%F`
fi

sql="
with tmp_order as (
select
user_id,
sum(oc.total_amount) order_amount,
count(*) order_count
from "$dwd_dbname".dwd_mysql_order_info_inc_1d oc
where date_format(oc.create_time,'yyyy-MM-dd')='$log_date'
group by user_id
),
tmp_payment as (
select
user_id,
sum(pi.total_amount) payment_amount,
count(*) payment_count
from "$dwd_dbname".dwd_mysql_payment_info_inc_1d pi
where date_format(pi.payment_time,'yyyy-MM-dd')='$log_date'
group by user_id
),
tmp_comment as (
select
```

```
user_id,
count(*) comment_count
from "$dwd_dbname".dwd_log_base_event_inc_1d c
where date_format(c.dt,'yyyy-MM-dd')='$log_date' and c.event_name='comment'
group by user_id
)
insert overwrite table "$dws_dbname".dws_traffic_ub_inc_1d
partition(dt='$log_date')
select
user_actions.user_id,
sum(user_actions.order_count),
sum(user_actions.order_amount),
sum(user_actions.payment_count),
sum(user_actions.payment_amount),
sum(user_actions.comment_count)
from (
select
    user_id,
    order_count,
    order_amount,
    0 payment_count ,
    0 payment_amount,
    0 comment_count
from tmp_order

union all
select
    user_id,
    0,
    0,
    payment_count,
    payment_amount,
    0
from tmp_payment

union all
select
    user_id,
    0,
    0,
    0,
    0,
    comment_count
    from tmp_comment
) user_actions
group by user_id;
"
$hive -e "$sql"


添加脚本执行权限
chmod u+x etl_dws_data.sh
```

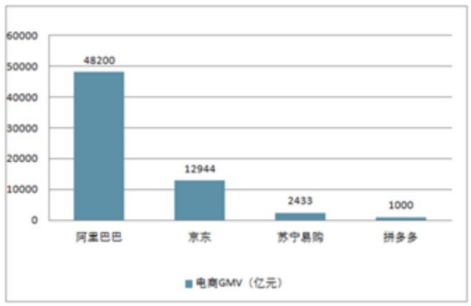
```
执行脚本清洗数据
sh etl_dws_data.sh 2021-04-21

验证数据:
select * from dws_traffic_ub_inc_1d limit 10;

use hdp_bw_dws_global;
show partitions dws_traffic_ub_inc_1d;
```

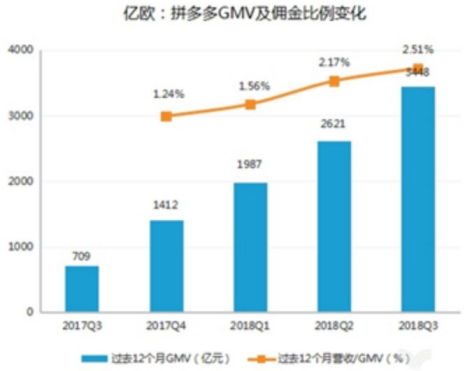
1.5 ADS层数据统计分析

1.5.1 需求一：GMV成交额分析

GMV：Gross Merchandise Volume，是成交总额（一定时间段内）的意思。

在电商网站定义里面是网站成交金额。这个实际指的是拍下订单金额，包含付款和未付款的部分。GMV是电商平台非常重视的统计指标，甚至写在招股书里。





不同公司计算GMV的算法不同：

GMV＝下单金额

GMV=下单金额 -（大额订单） 10万

GMV=下单金额＋预定金额

日期	当日下单uv	当日支付uv	当日gmv订单数	当日gvm订单额	当日支订单数	当日支付订单额
2021 11-11	100	50	1000	999999	500	555555

```
建表：gmv的ads层表
use hdp_bw_ads_global;
drop table if exists ads_trade_gmv_sum_inc_1d_0p;
create external table ads_trade_gmv_sum_inc_1d_0p(
`dt` string COMMENT '统计日期',
`gmv_uv` bigint comment '当日下单uv',
`pay_uv` bigint comment '当日支付uv',
`gmv_pv` bigint comment '当日gmv订单数',
`gmv_amount` decimal(16,2) comment '当日gmv下单金额',
`pay_pv` bigint comment '当日支付订单数',
`pay_amount` decimal(16,2) comment '当日支付订单额'
) COMMENT '每日gmv汇总统计'
row format delimited fields terminated by '\t' ;

数据统计
```

```
SQL实现:
set hivevar:stat_date=2021-04-21;
insert into table hdp_bw_ads_global.ads_trade_gmv_sum_inc_1d_0p
select '${stat_date}',
count(distinct case when order_count > 0 then user_id end) gmv_uv,
count(distinct case when payment_count > 0 then user_id end) pay_uv,
sum(order_count) gmv_pv,
sum(order_amount) gmv_amount,
sum(payment_count) pay_pv,
sum(payment_amount) pay_amount
from hdp_bw_dws_global.dws_traffic_ub_inc_1d
where dt ='${stat_date}'
group by dt;

查询统计结果
select * from hdp_bw_ads_global.ads_trade_gmv_sum_inc_1d_0p where dt='2021-04-
21';

结果
2021-04-21 90491 69207 170889 16559849119.00 107420 2159455438.00
```

1.5.2 需求二：用户活跃&新增&累计设备主题

- 相关业务术语

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

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

【活跃用户】：打开应用的用户即为活跃用户，不考虑用户的使用情况。每天一台设备打开多次会被计为一个
活跃用户。
【周（月）活跃用户】：某个自然周（月）内启动过应用的用户，该周（月）内的多次启动只记一个活跃用
户。

【月活跃率】：月活跃用户与截止到该月累计的用户总和之间的比例。

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

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

【本周回流用户】：上周未启动过应用，本周启动了应用的用户。

【连续n周活跃用户】：连续n周，每周至少启动一次。

【忠诚用户】：连续活跃5周以上的用户
```

【连续活跃用户】：连续2周及以上活跃的用户

【近期流失用户】：连续n(2<= n <= 4)周没有启动应用的用户。（第n+1周没有启动过）

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

【用户新鲜度】：每天启动应用的新老用户比例，即新增用户数占活跃用户数的比例。

【单次使用时长】：每次启动使用的时间长度。

【日使用时长】：累计一天内的使用时间长度。

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

- 需求实现

具体需求：统计当日、当周、当月访问的每个设备明细和每日新增设备明细

1、设计创建对应 DWS 层表(每日、每周、每月设备明细表)

```
use hdp_bw_dws_global;
drop table if exists dws_user_active_device_detail_inc_1d;
create external table dws_user_active_device_detail_inc_1d(
`mid_id` string COMMENT '设备唯一标识',
`user_id` string COMMENT '用户标识',
`version_code` string COMMENT '程序版本号',
`version_name` string COMMENT '程序版本名',
`lang` string COMMENT '系统语言',
`source` string COMMENT '渠道号',
`os` string COMMENT '安卓系统版本',
`area` string COMMENT '区域',
`model` string COMMENT '手机型号',
`brand` string COMMENT '手机品牌',
`sdk_version` string COMMENT 'sdkVersion',
`gmail` string COMMENT 'gmail',
`height_width` string COMMENT '屏幕宽高',
`app_time` string COMMENT '客户端日志产生时的时间',
`network` string COMMENT '网络模式',
`lng` string COMMENT '经度',
`lat` string COMMENT '纬度'
) COMMENT '活跃用户按天明细'
PARTITIONED BY ( `dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");

drop table if exists dws_user_active_device_detail_inc_1w;
```

```
create external table dws_user_active_device_detail_inc_1w(
`mid_id` string COMMENT '设备唯一标识',
`user_id` string COMMENT '用户标识',
`version_code` string COMMENT '程序版本号',
`version_name` string COMMENT '程序版本名',
`lang` string COMMENT '系统语言',
`source` string COMMENT '渠道号',
`os` string COMMENT '安卓系统版本',
`area` string COMMENT '区域',
`model` string COMMENT '手机型号',
`brand` string COMMENT '手机品牌',
`sdk_version` string COMMENT 'sdkVersion',
`gmail` string COMMENT 'gmail',
`height_width` string COMMENT '屏幕宽高',
`app_time` string COMMENT '客户端日志产生时的时间',
`network` string COMMENT '网络模式',
`lng` string COMMENT '经度',
`lat` string COMMENT '纬度'
) COMMENT '活跃用户按周明细'
PARTITIONED BY (`dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");

drop table if exists dws_user_active_device_detail_inc_1m;
create external table dws_user_active_device_detail_inc_1m(
`mid_id` string COMMENT '设备唯一标识',
`user_id` string COMMENT '用户标识',
`version_code` string COMMENT '程序版本号',
`version_name` string COMMENT '程序版本名',
`lang` string COMMENT '系统语言',
`source` string COMMENT '渠道号',
`os` string COMMENT '安卓系统版本',
`area` string COMMENT '区域',
`model` string COMMENT '手机型号',
`brand` string COMMENT '手机品牌',
`sdk_version` string COMMENT 'sdkVersion',
`gmail` string COMMENT 'gmail',
`height_width` string COMMENT '屏幕宽高',
`app_time` string COMMENT '客户端日志产生时的时间',
`network` string COMMENT '网络模式',
`lng` string COMMENT '经度',
`lat` string COMMENT '纬度'
) COMMENT '活跃用户按月明细'
PARTITIONED BY ( `dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");
```

数据清洗

按天统计表:

```
set hivevar:stat_date=2021-05-01;
```

```
set hive.exec.dynamic.partition.mode=nonstrict;
insert overwrite table
hdp_bw_dws_global.dws_user_active_device_detail_inc_1d partition(dt)
select
mid_id,
collect_set(user_id)[0] user_id,
collect_set(version_code)[0] version_code,
collect_set(version_name)[0] version_name,
collect_set(lang)[0]lang,
collect_set(source)[0] source,
collect_set(os)[0] os,
collect_set(area)[0] area,
collect_set(model)[0] model,
collect_set(brand)[0] brand,
collect_set(sdk_version)[0] sdk_version,
collect_set(gmail)[0] gmail,
collect_set(height_width)[0] height_width,
collect_set(app_time)[0]app_time,
collect_set(network)[0] network,
collect_set(lng)[0]lng,
collect_set(lat)[0]lat,
'${stat_date}'
from hdp_bw_dwd_global.dwd_log_base_start_inc_1d
where dt='${stat_date}'
group by mid_id;
```

按周统计表:

```
set hivevar:stat_date=2020-09-19;
set hive.exec.dynamic.partition.mode=nonstrict;
insert overwrite table
hdp_bw_dws_global.dws_user_active_device_detail_inc_1w partition(dt)
select
mid_id,
collect_set(user_id)[0] user_id,
collect_set(version_code)[0] version_code,
collect_set(version_name)[0] version_name,
collect_set(lang)[0] lang,
collect_set(source)[0] source,
collect_set(os)[0] os,
collect_set(area)[0] area,
collect_set(model)[0] model,
collect_set(brand)[0] brand,
collect_set(sdk_version)[0] sdk_version,
collect_set(gmail)[0] gmail,
collect_set(height_width)[0] height_width,
collect_set(app_time)[0] app_time,
collect_set(network)[0] network,
collect_set(lng)[0] lng,
collect_set(lat)[0] lat,
date_add(next_day('${stat_date}','MO'), -1)
from hdp_bw_dwd_global.dwd_log_base_start_inc_1d
```

```
where dt>=date_add(next_day('${stat_date}','MO'),-7) and
dt<=date_add(next_day('${stat_date}','MO'),-1)
group by mid_id;

按月统计表:
set hivevar:stat_date=2020-09-19;
set hive.exec.dynamic.partition.mode=nonstrict;
insert overwrite table
hdp_bw_dws_global.dws_user_active_device_detail_inc_1m partition(dt)
select
mid_id,
collect_set(user_id)[0] user_id,
collect_set(version_code)[0] version_code,
collect_set(version_name)[0] version_name,
collect_set(lang)[0] lang,
collect_set(source)[0] source,
collect_set(os)[0] os,
collect_set(area)[0] area,
collect_set(model)[0] model,
collect_set(brand)[0] brand,
collect_set(sdk_version)[0] sdk_version,
collect_set(gmail)[0] gmail,
collect_set(height_width)[0] height_width,
collect_set(app_time)[0] app_time,
collect_set(network)[0] network,
collect_set(lng)[0] lng,
collect_set(lat)[0] lat,
date_format('${stat_date}','yyyy-MM')
from hdp_bw_dwd_global.dwd_log_base_start_inc_1d
where date_format(dt,'yyyy-MM') = date_format('${stat_date}','yyyy-MM')
group by mid_id;
```

```
3、设计创建对应DWS层表(每日新增设备明细表)
use hdp_bw_dws_global;
drop table if exists dws_user_new_device_detail_inc_1d;
create external table dws_user_new_device_detail_inc_1d(
`mid_id` string COMMENT '设备唯一标识',
`user_id` string COMMENT '用户标识',
`version_code` string COMMENT '程序版本号',
`version_name` string COMMENT '程序版本名',
`lang` string COMMENT '系统语言',
`source` string COMMENT '渠道号',
`os` string COMMENT '安卓系统版本',
`area` string COMMENT '区域',
`model` string COMMENT '手机型号',
`brand` string COMMENT '手机品牌',
`sdk_version` string COMMENT 'sdkVersion',
`gmail` string COMMENT 'gmail',
`height_width` string COMMENT '屏幕宽高',
`app_time` string COMMENT '客户端日志产生时的时间',
```

```
`network` string COMMENT '网络模式',
`lng` string COMMENT '经度',
`lat` string COMMENT '纬度'
) COMMENT '新用户设备按天明细'
PARTITIONED BY ( `dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");
```

4、数据清洗

```
set hivevar:stat_date=2020-09-19;
set hive.exec.dynamic.partition.mode=nonstrict;
insert overwrite table dws_user_new_device_detail_inc_1d partition (dt)
select
dau.mid_id,
dau.user_id ,
dau.version_code ,
dau.version_name ,
dau.lang ,
dau.source,
dau.os,
dau.area,
dau.model,
dau.brand,
dau.sdk_version,
dau.gmail,
dau.height_width,
dau.app_time,
dau.network,
dau.lng,
dau.lat,
'${stat_date}'
from hdp_bw_dws_global.dws_user_active_device_detail_inc_1d dau left
join hdp_bw_dws_global.dws_user_new_device_detail_inc_1d nd on
dau.mid_id=nd.mid_id and nd.dt<'${stat_date}'
where dau.dt='${stat_date}' and nd.mid_id is null;
```

查询结果数据:

```
select * from hdp_bw_dws_global.dws_user_new_device_detail_inc_1d limit 10;
```

5、ADS层计算当日、当周、当月活跃设备数和当日新用户数

```
drop table if exists ads_activity_uv_inc_1d_0p;
create external table ads_activity_uv_inc_1d_0p
(
`dt` string COMMENT '统计日期',
`dau_count` bigint COMMENT '当日用户数量',
`wau_count` bigint COMMENT '当周用户数量',
`mau_count` bigint COMMENT '当月用户数量',
`nau_count` bigint COMMENT '当日新用户数量',
`is_weekend` string COMMENT 'Y,N是否是周末,用于得到本周最终结果',
`is_monthend` string COMMENT 'Y,N是否是月末,用于得到本月最终结果'
```

```
) COMMENT '每日活跃用户汇总统计'
row format delimited fields terminated by '\t' ;


set hivevar:stat_date=2021-04-21;
insert into table hdp_bw_ads_global.ads_activity_uv_inc_1d_0p
select
'${stat_date}' dt,
d.ct,
w.ct,
m.ct,
n.ct,
if(date_add(next_day('${stat_date}','MO'), -1)='${stat_date}', 'Y', 'N') ,
if(last_day('${stat_date}')='${stat_date}', 'Y', 'N')
from
(
select
'${stat_date}' dt,
count(mid_id) ct
from hdp_bw_dws_global.dws_user_active_device_detail_inc_1d
where dt='${stat_date}'
)d join
(
select
'${stat_date}' dt,
count (mid_id) ct
from hdp_bw_dws_global.dws_user_active_device_detail_inc_1w
where dt=date_add(next_day('${stat_date}','MO'), -1)
)w on d.dt=w.dt
join
(
select
'${stat_date}' dt,
count (mid_id) ct
from hdp_bw_dws_global.dws_user_active_device_detail_inc_1m
where dt=date_format('${stat_date}','yyyy-MM')
)m on d.dt=m.dt
join
(
select
'${stat_date}' dt,
count (mid_id) ct
from hdp_bw_dws_global.dws_user_new_device_detail_inc_1d
where dt='${stat_date}'
)n on d.dt=n.dt;


查询结果数据:
select * from hdp_bw_ads_global.ads_activity_uv_inc_1d_0p limit 10;
```

1.5.3 需求三：用户留存主题

留存用户：某段时间内的新增用户（活跃用户），经过一段时间后，又继续使用应用的被认作是留存用户；

留存率：留存用户占当时新增用户（活跃用户）的比例即是留存率。

例如，2月10日新增用户100，这100人在2月11日启动过应用的有30人，2月12日启动过应用的有25人，2月13日启动过应用的有32人；

则2月10日新增用户次日的留存率是30/100 = 30%，两日留存率是25/100=25%，三日留存率是32/100=32%。

时间	新增用户	1天后	2天后	3天后
2019-02-10	100	30% (2-11)	25% (2-12)	32% (2-13)
2019-02-11	200	20% (2-12)	15% (2-13)	
2019-02-12	100	25% (2-13)		
2019-02-13				

- 统计每天1、3、7、30日留存率

```
1、DWS层（每日留存用户明细表）创建
use hdp_zhuanzhuan_dws_global;
drop table if exists dws_user_new_device_retention_inc_1d;
create external table dws_user_new_device_retention_inc_1d(
`mid_id` string COMMENT '设备唯一标识',
`user_id` string COMMENT '用户标识',
`version_code` string COMMENT '程序版本号',
`version_name` string COMMENT '程序版本名',
`lang` string COMMENT '系统语言',
`source` string COMMENT '渠道号',
`os` string COMMENT '安卓系统版本',
`area` string COMMENT '区域',
`model` string COMMENT '手机型号',
`brand` string COMMENT '手机品牌',
`sdk_version` string COMMENT 'sdkVersion',
`gmail` string COMMENT 'gmail',
`height_width` string COMMENT '屏幕宽高',
`app_time` string COMMENT '客户端日志产生时的时间',
`network` string COMMENT '网络模式',
`lng` string COMMENT '经度',
`lat` string COMMENT '纬度',
`create_date` string comment '设备新增时间',
`retention_day` int comment '截止当前日期留存天数'
) COMMENT '每日新用户留存明细'
PARTITIONED BY ( `dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");

2、清洗数据（每天计算前1,3,7,n天的新用户访问留存明细）
set hivevar:stat_date=2020-09-19;
insert overwrite table
hdp_zhuanzhuan_dws_global.dws_user_new_device_retention_inc_1d
partition(dt='${stat_date}')
select
nm.mid_id,
nm.user_id ,
```

```
nm.version_code ,
nm.version_name ,
nm.lang ,
nm.source,
nm.os,
nm.area,
nm.model,
nm.brand,
nm.sdk_version,
nm.gmail,
nm.height_width,
nm.app_time,
nm.network,
nm.lng,
nm.lat,
nm.dt,
datediff(ud.dt,nm.dt) retention_day
from hdp_zhuanzhuan_dws_global.dws_user_active_device_detail_inc_1d ud join
hdp_zhuanzhuan_dws_global.dws_user_new_device_detail_inc_1d nm on ud.mid_id
=nm.mid_id
where ud.dt='${stat_date}' and nm.dt in(date_add('${stat_date}',-1),
date_add('${stat_date}',-3),
date_add('${stat_date}',-7),date_add('${stat_date}',-30));
```

思路分两步走：
1、先求的过去1天，过去3天，过去7天，过去30这个四个日期的新增用户 联合到一起形成一个集合
四个值（四个日期的新增用户） + 一个集合（四个日期的新增用户合起来作为条件作为今天的筛选条件）
2、在这个集合中，还有多少用户是今天登陆使用了APP的 （200个） （result 交集 过去1天） / 过去1天

3、查询数据（每天计算前1,3,7,30天的新用户访问留存明细）

```
set hivevar:stat_date=2020-09-19;
select retention_day,count(*) from
hdp_zhuanzhuan_dws_global.dws_user_new_device_retention_inc_1d where
dt='${stat_date}' group by retention_day;
```

4、ADS 层留存用户数
三日留存率： 对于过去3天那一天来说，用几天的登陆用户 / 过去三天那一天的新增用户

```
use hdp_zhuanzhuan_ads_global;
drop table if exists ads_activity_new_dau_retention_inc_1d_0p;
create external table ads_activity_new_dau_retention_inc_1d_0p
(
`dt` string COMMENT '设备新增日期',
`retention_day` int comment '截止当前日期留存天数',
`retention_count` bigint comment '留存数量'
) COMMENT '每日用户留存统计'
row format delimited fields terminated by '\t';

set hivevar:stat_date=2020-09-19;
insert into table
hdp_zhuanzhuan_ads_global.ads_activity_new_dau_retention_inc_1d_0p
```

```
select
create_date,
retention_day,
count(*) retention_count
from hdp_zhuanzhuan_dws_global.dws_user_new_device_retention_inc_1d
where dt='${stat_date}'
group by create_date,retention_day;
```

1.6 订单拉链表设计

1.6.1 什么是拉链表

> 拉链表，记录每条信息的生命周期为单位，一条记录的生命周期结束，就重新开始一条新的记录，并把当前日期放入生效开始日期。

- 如果当前信息至今有效，在生效结束日期中植入一个极大值（如 9999-99-99 或者 9999-12-31）

订单ID	订单额	状态	生效开始日期	生效结束日期
1	2000.00	待支付	2020-01-01	9999-99-99
2	1000.00	待支付	2020-01-01	2020-01-01
2	1000.00	已支付	2020-01-02	9999-99-99
3	210.00	已支付	2020-01-01	9999-99-99
4	3300.00	待支付	2020-01-02	9999-99-99
5	55.00	已支付	2020-01-02	9999-99-99

1.6.2 为什么要做拉链表

- 1）需要查看某些业务信息的某个时间点当日的信息
- 2）数据会发生变化，但是大部分是不变的。比如订单信息从下单、支付、发货、签收等状态经历了一周，大部分时间是不变的。（无法每日做增量）
- 3）数据量有一定规模，无法按照每日全量的方式保存。比如：10亿订单*365天，每天一份订单

1.6.3 拉链表形成过程

1）假设， 2019年 1月1日的订单全量表是最初始的订单表，如下

订单 Id	状态
1	待支付
2	待支付
3	已支付

2）初始的拉链表就等于最开始的2019年 1月1日的订单全量表

订单 Id	状态	生效开始日期	生效结束日期
1	待支付	2019-01-01	9999-99-99
2	待支付	2019-01-01	9999-99-99
3	已支付	2019-01-01	9999-99-99

3）第二天 1月2日 订单全量表（订单2发生状态修改；订单4、5增加）

订单 Id	状态
1	待支付
2	已支付
3	已支付
4	待支付
5	已支付

4）根据订单表的创建时间和操作时间，得到订单变化表。

订单 Id	状态
2	已支付
4	待支付
5	已支付

5）订单变化表与之前的拉链表合并得到

订单 Id	状态	生效开始日期	生效结束日期
1	待支付	2019-01-01	9999-99-99
2	待支付	2019-01-01	2019-01-01
2	已支付	2019-01-02	9999-99-99
3	已支付	2019-01-01	9999-99-99
4	待支付	2019-01-02	9999-99-99
5	已支付	2019-01-02	9999-99-99

通过，某个日期<=生效开始日期 且 某个日期>=生效结束日期，能够得到某个时间点的数据全量切片。

1）拉链表数据

订单 Id	状态	生效开始日期	生效结束日期
1	待支付	2019-01-01	9999-99-99
2	待支付	2019-01-01	2019-01-01
2	已支付	2019-01-02	9999-99-99
3	已支付	2019-01-01	9999-99-99
4	待支付	2019-01-02	9999-99-99
5	已支付	2019-01-02	9999-99-99

2）例如获取2019-01-01的历史切片：select * from x_order where start_date<='2019-01-01' and end_date>='2019-01-01'

订单 Id	状态	生效开始日期	生效结束日期
1	待支付	2019-01-01	9999-99-99
2	待支付	2019-01-01	2019-01-01
3	已支付	2019-01-01	9999-99-99

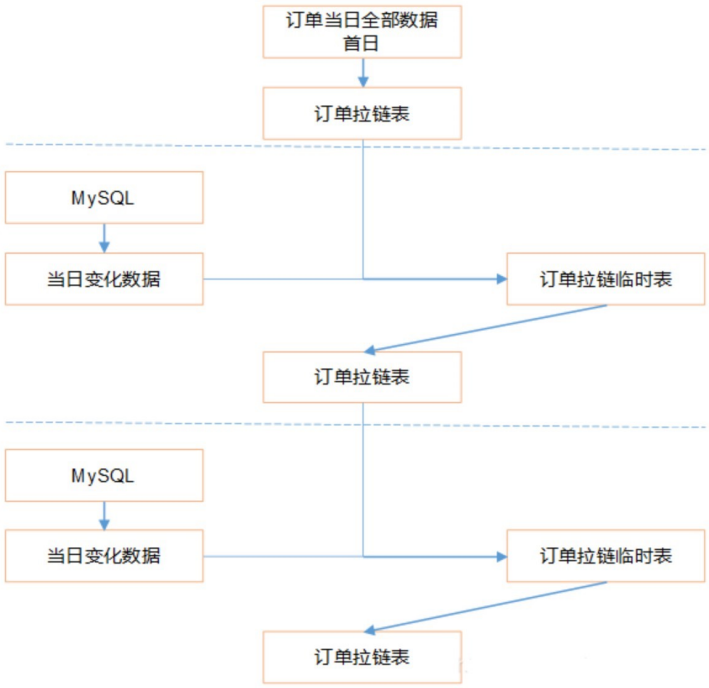
3）例如获取2019-01-02的历史切片： select * from x_order where start_date<='2019-01-02' and end_date>='2019-01-02'

订单 Id	状态	生效开始日期	生效结束日期
1	待支付	2019-01-01	9999-99-99
2	已支付	2019-01-02	9999-99-99
3	已支付	2019-01-01	9999-99-99
4	待支付	2019-01-02	9999-99-99
5	已支付	2019-01-02	9999-99-99

如何使用拉链表

1.6.4 拉链表制作过程

订单当日全部数据和MySQL中每天变化的数据拼接在一起，形成一个新的临时拉链表数据。用临时的拉链表覆盖旧的拉链表数据。（这就解决了hive表中数据不能更新的问题）



- 1、初始化拉链表，首次独立执行

```
use hdp_zhuanzhuan_dwd_global;
drop table if exists dwd_mysql_order_his_full_1d_0p;
create external table dwd_mysql_order_his_full_1d_0p (
`id` string COMMENT '订单编号',
`total_amount` decimal(10,2) COMMENT '订单金额',
`order_status` string COMMENT '订单状态',
`user_id` string COMMENT '用户id',
`payment_way` string COMMENT '支付方式',
`out_trade_no` string COMMENT '支付流水号',
`create_time` string COMMENT '创建时间',
`operate_time` string COMMENT '操作时间',
`start_date` string COMMENT '有效开始日期',
`end_date` string COMMENT '有效结束日期'
) COMMENT '订单拉链表'
stored as parquet
tblproperties ("parquet.compression"="snappy");

set hivevar:stat_date=2020-09-18;
insert overwrite table hdp_zhuanzhuan_dwd_global.dwd_mysql_order_his_full_1d_0p
select
id,
total_amount,
order_status,
user_id,
payment_way,
out_trade_no,
create_time,
operate_time,
'${stat_date}',
'9999-99-99'
from hdp_zhuanzhuan_ods_global.ods_zz_order_info_inc_1d oi
where oi.dt<='${stat_date}';

2、获取当日变动（包括新增，修改）每日执行
如何获得每日变动表
（a）最好表内有创建时间和变动时间
（b）如果没有，可以利用第三方工具监控比如canal，监控MySQL的实时变化进行记录(麻烦)。
（c）逐行对比前后两天的数据，检查md5(concat(全部有可能变化的字段))是否相同(low)
（d）要求业务数据库提供变动流水(人品，颜值)
因为 dwd_mysql_order_info_inc_1d 本身导入过来就是新增变动明细的表，所以不用处理

3、先合并变动信息，再追加新增信息覆盖更新 dwd_mysql_order_his_full_1d_0p
set hivevar:stat_date=2020-09-19;
drop table if exists hdp_zhuanzhuan_tmp_global.tmp_dwd_mysql_order_his_full;
create table hdp_zhuanzhuan_tmp_global.tmp_dwd_mysql_order_his_full as
select * from
(
select id ,
total_amount ,
order_status ,
user_id ,
```

```
payment_way ,
out_trade_no ,
create_time ,
operate_time ,
'${stat_date}' start_date,
'9999-99-99' end_date
from hdp_zhuanzhuan_dwd_global.dwd_mysql_order_info_inc_1d where
dt='${stat_date}'
union all
select oh.id,
oh.total_amount,
oh.order_status,
oh.user_id,
oh.payment_way,
oh.out_trade_no,
oh.create_time,
oh.operate_time,
oh.start_date,
if(oi.id is null, oh.end_date, date_add(oi.dt, -1)) end_date
from hdp_zhuanzhuan_dwd_global.dwd_mysql_order_his_full_1d_0p oh left join
(
select
*
from hdp_zhuanzhuan_dwd_global.dwd_mysql_order_info_inc_1d
where dt = '${stat_date}'
) oi
on oh.id = oi.id and oh.end_date = '9999-99-99'
) his;
insert overwrite table hdp_zhuanzhuan_dwd_global.dwd_mysql_order_his_full_1d_0p
select * from hdp_zhuanzhuan_tmp_global.tmp_dwd_mysql_order_his_full;

4、获取某个时间的全量切片数据
set hivevar:stat_date=2020-09-19;
select t.* from hdp_zhuanzhuan_dwd_global.dwd_mysql_order_his_full_1d_0p t where
start_date<='${stat_date}' and end_date>='${star_date}' limit 10;
```

2. 本次课程总结

```
今天的主要内容是:
1、数仓搭建ODS、DWD与DWS层
2、ADS层数据统计分析
3、订单拉链表设计
```