尚硅谷大数据技术之数据质量管理

(作者：尚硅谷大数据研发部)

版本：V1.0

第1章　数据质量

1.1　概述

数据质量的高低代表了该数据满足数据消费者期望的程度，这种程度基于他们对数据的使用预期，只有达到数据的使用预期才能给予管理层正确的决策参考。数据质量管理作为数据仓库的一个重要模块，主要可以分为数据的健康标准量化、监控和保障。

1.2. 数据质量标准分类

① 数据完整性：数据不存在大量的缺失值、不缺少某一日期/部门/地点等部分维度的数据，同时在 ETL 过程当中应保证数据的完整不丢失。验证数据时总数应符合正常规律时间推移，记录数总数的增长符合正常的趋势。

② 数据一致性：数仓各层的数据，应与上一层保持数据一致，最终经过数据清洗转化（ETL）的宽表/指标能和数据源保持一致。

1.3　数据质量管理解决方案

本文通过 Shell 命令和 Hive 脚本的方式，通过验证增量数据的记录数、全表空值记录数、全表记录数是否在合理的范围之内，以及验证数据来源表和目标表一致性，确定当日的数据是否符合健康标准，达到数据质量的监控与管理。

第 2 章　ODS 层数据校验

2.1 数据校验通用脚本

通过 shell 脚本调用 hive，检验当日分区增加的记录数量和全表记录数量是否在合理的范围之内，同时检验关键字段为空的记录的记录数量。

1）创建数据检查脚本文件夹，用于存放数据校验 shell 脚本

```
[atguigu@hadoop102 module]$ mkdir -p data_check/sh
[atguigu@ hadoop102 sh]$ pwd
/opt/module/data_check/sh
```

2）在 Hive 中创建表数据质量校验记录表，记录数据校验的各个指标：

```
[atguigu@atguigu data_check]$ hive
```

创建数据库，用于存放数据质量校验的结果数据：

```
hive (default)> create database datacheck;
```

创建数据表，用于存放 ods 层的数据检验结果：

```
hive (datacheck)> create table
datacheck.table_count_add_standard(
    data_date string comment '数据时间分区 dt',
    database_name string comment '库名',
    table_name string comment '表名',
    table_type string comment '表类型（全量/增量）',
    add_count bigint comment '当日增量数据的记录数',
    null_count bigint comment '表空值记录数',
    total_count bigint comment '全表记录数'
);
hive (datacheck)>quit;
```

3）在路径/opt/module/data_check/sh 下创建数据检验增量表通用 shell 脚本

```
[atguigu@hadoop102 sh]$ vim increment_data_check_public.sh
```

在脚本中编写如下内容：

```
#!/bin/bash
# 增量数据所在的日期分区
do_date=$1
# 校验数据的表名
table_name=$2
# 需要校验空值的列名，以逗号’,’隔开
null_column=$3
# 初始化 SQL 查询语句
null_where_sql_str=''
# 将空值检验字符串切成列名数组
array=(${null_column//,/ })
# 遍历数组，拼接空值查询条件
for(( i=0;i<${#array[@]};i++)) do
   if [ $i -eq 0 ];then
       null_where_sql_str=" where ${array[i]} is null "
   else
       null_where_sql_str="$null_where_sql_str or ${array[i]}
is null "
   fi
done;
# 执行当日增量数据记录数量 SQL 查询语句
add_count_query_result=`hive -e "select count(*) from
gmall.$table_name where dt='$do_date'"`
# 取出当日增量数据记录数量
add_count=${add_count_query_result:3}
# 执行当日全表数据记录数量 SQL 查询语句
total_count_query_result=`hive -e "select count(*) from
gmall.$table_name"`
# 取出当日全量数据记录数量
total_count=${total_count_query_result:3}
# 执行全表空值数据记录数量 SQL 查询语句
table_null_query_result=`hive -e "select count(*) from
gmall.$table_name $null_where_sql_str"`
```

```
# 取出全表空值数据记录数量
null_count=${table_null_query_result:3}
# 将所有数据检验结果插入到表中
hive -e "insert into datacheck.table_count_add_standard
values('$do_date','gmall','$table_name','increment_table',$add
_count,$null_count,'$total_count')"
```

脚本参数注释：

第一个参数：传入时间分区参数(dt)

第二个参数：需要进行数据校验的表名(table_name)

第三个参数：需要判断是否为空值的字段名称用逗号‘ ，’隔开，例如：col1,col2,col3

给脚本/opt/module/data_check/sh/increment_data_check_public.sh 赋权限：

```
[atguigu@hadoop102 sh]$ chmod 777 increment_data_check_public.s
h
```

脚本执行示例：

```
[atguigu@hadoop102 sh]$
./increment_data_check_public.sh 2020-06-14 ods_activity_rule
id,activity_id
```

4）在路径/opt/module/data_check/sh 下创建数据检验全量表通用 shell 脚本

```
[atguigu@hadoop102 sh]$ vim total_data_check_public.sh
```

在脚本中编写如下内容：

```
#!/bin/bash
# 增量数据所在的日期分区
do_date=$1
# 校验数据的表名
table_name=$2
# 需要校验空值的列名，以逗号’,’隔开
null_column=$3
# 将空值检验字符串切成列名数组
null_where_sql_str=''
# 遍历数组，拼接空值查询条件
array=(${null_column//,/ })
# 遍历数组，拼接空值查询条件
for(( i=0;i<${#array[@]};i++)) do
   if [ $i -eq 0 ];then
       null_where_sql_str=" where ${array[i]} is null "
   else
       null_where_sql_str="$null_where_sql_str or ${array[i]}
is null "
   fi
done;
# 执行当日全表数据记录数量 SQL 查询语句
table_count_query_result=`hive -e "select count(*) from
gmall.$table_name"`
# 取出当日全量数据记录数量
table_count=${table_count_query_result:3}
# 执行全表空值数据记录数量 SQL 查询语句
table_null_query_result=`hive -e "select count(*) from
```

```
gmall.$table_name $null_where_sql_str"`
# 取出全表空值数据记录数量
null_count=${table_null_query_result:3}
# 将所有数据检验结果插入到表中
hive -e "insert into datacheck.table_count_add_standard
values('$do_date','gmall','$table_name','total_table',null,$nu
ll_count,'$table_count')"
```

脚本参数注释：

第一个参数：传入数据校验日期(dt)

第二个参数：需要进行数据校验的表名(table_name)

第三个参数：需要判断是否为空值的字段名称用逗号’ , ‘隔开，例如：col1,col2,col3

给脚本/opt/module/data_check/sh/total_data_check_public.sh 赋权限：

```
[atguigu@hadoop102 sh]$ chmod 777 total_data_check_public.sh
```

脚本执行示例：

```
[atguigu@hadoop102 sh]$ ./total_data_check_public.sh 2020-06-
14 ods_activity_rule id,activity_id
```

2.2 ODS 层各表检验

1. 涉及表

增量检查

（1）订单详情表（ods_order_detail）
（2）用户表（ods_user_info）
（3）支付流水表（ods_payment_info）
（4）订单状态表（ods_order_status_log）
（5）商品评论表（ods_comment_info）
（6）退单表（ods_order_refund_info）
（7）活动订单关联表（ods_activity_order）

全量检查

（1）订单表（ods_order_info）
（2）SKU 商品表（ods_sku_info）
（3）商品一级分类表（ods_base_category1）
（4）商品二级分类表（ods_base_category2）
（5）商品三级分类表（ods_base_category3）
（6）品牌表（ods_base_trademark）
（7）SPU 商品表（ods_spu_info）
（8）加购表（ods_cart_info）
（9）商品收藏表（ods_favor_info）
（10）优惠券领用表（ods_coupon_use）
（11）优惠券表（ods_coupon_info）
（12）活动表（ods_activity_info）
（13）优惠规则表（ods_activity_rule）
（14）编码字典表（ods_base_dic）

2. ODS 层数据检查脚本

1）在路径/opt/module/data_check/sh 下创建 ODS 层数据检查脚本

```
[atguigu@atguigu sh]$ pwd
/opt/module/data_check/sh
[atguigu@atguigu sh]$ vim ods_data_check.sh
```

在脚本中编写如下内容：

```
#!/bin/bash
data_date=$1

# 增量检查
# 订单详情表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_order_detail
id,order_id,user_id,sku_id,sku_name,order_price,sku_num,create
_time
# 用户表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_user_info
id,name,birthday,gender,email,user_level,create_time,operate_t
ime
# 支付流水表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_payment_info
id,out_trade_no,order_id,user_id,alipay_trade_no,total_amount,
subject,payment_type,payment_time
# 订单状态表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_order_status_log
id,order_id,order_status,operate_time
# 商品评论表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_comment_info
id,user_id,sku_id,spu_id,order_id,appraise,create_time
# 退单表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_order_refund_info
id,user_id,order_id,sku_id,refund_type,refund_num,refund_amoun
t,refund_reason_type,create_time
# 活动订单关联表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_activity_order
id,activity_id,order_id,create_time
# 全量检查
# 订单表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_order_info
id,final_total_amount,order_status,user_id,out_trade_no,create
_time,operate_time,province_id,benefit_reduce_amount,original_
total_amount,feight_fee
# SKU 商品表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_sku_info
```

```
id,spu_id,price,sku_name,sku_desc,weight,tm_id,category3_id,cr
eate_time
# 商品一级分类表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_base_category1 id,name
# 商品二级分类表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_base_category2 id,name,category1_id
# 商品三级分类表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_base_category3 id,name,category2_id
# 品牌表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_base_trademark tm_id,tm_name
# SPU 商品表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_spu_info id,spu_name,category3_id,tm_id
# 加购表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_cart_info
id,user_id,sku_id,cart_price,sku_num,sku_name,create_time,oper
ate_time,is_ordered,order_time
# 商品收藏表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_favor_info
id,user_id,sku_id,spu_id,is_cancel,create_time,cancel_time
# 优惠券领用表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_coupon_use
id,coupon_id,user_id,order_id,coupon_status,get_time,using_tim
e,used_time
# 优惠券表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_coupon_info
id,coupon_name,coupon_type,condition_amount,condition_num,acti
vity_id,benefit_amount,benefit_discount,create_time,range_type
,spu_id,tm_id,category3_id,limit_num,operate_time,expire_time
# 活动表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_activity_info
id,activity_name,activity_type,start_time,end_time,create_time
# 优惠规则表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_activity_rule
id,activity_id,condition_amount,condition_num,benefit_amount,b
enefit_discount,benefit_level
# 编码字典表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date ods_base_dic
dic_code,dic_name,parent_code,create_time,operate_time
```

2）给脚本/opt/module/data_check/sh/ods_data_check.sh 赋权限：

```
[atguigu@atguigu sh]$ chmod 777 ods_data_check.sh
```

第 3 章 DWD 层数据校验

3.1 数据校验通用脚本

1）创建数据表，用于存放 dwd 层的数据检验结果：

```
hive (datacheck)> create table datacheck.dwd_table_data_check(
    data_date string comment '数据时间分区 dt',
    database_name string comment '库名',
    source_table_name string comment '数据源表表名',
    source_column string comment '数据源表字段名',
    target_table_name string comment '数据目标表表名',
    target_column string comment '数据目标表字段名',
    consistent_data_count bigint comment '全表数据一致记录数',
    source_table_count bigint comment '数据源表全表记录数',
    target_table_count bigint comment '数据目标表全表记录数'
);
hive (datacheck)>quit;
```

2）在路径/opt/module/data_check/sh 下创建 dwd 层数据一致性检验通用 shell 脚本

```
[atguigu@hadoop102 sh]$ vim table_consistent_check_public.sh
```

在脚本中编写如下内容：

```
#!/bin/bash
# 增量数据所在的日期分区
do_date=$1
# 校验数据源表的表名
source_table_name=$2
# 检验数据源表的字段（与目标表顺序一致才能对比两个字段）
source_column=$3
# 检验数据目标表的表名
target_table_name=$4
# 检验数据目标表的字段（与源表顺序一致才能对比两个字段）
target_column=$5

# 初始化 SQL 查询语句
join_on_sql_str=''
# 将检验数据源表的字段切成列名数组
source_column_array=(${source_column//,/ })
# 将检验数据目标表的字段切成列名数组
target_column_array=(${target_column//,/ })

# 遍历数组，拼接表关联条件，输入字段全部关联
for(( i=0;i<${#source_column_array[@]};i++)) do
   if [ $i -eq 0 ];then
       join_on_sql_str=" on
$source_table_name.${source_column_array[i]}=$target_table_nam
e.${target_column_array[i]} "
   else
       join_on_sql_str="$join_on_sql_str and
$source_table_name.${source_column_array[i]}=$target_table_nam
e.${target_column_array[i]} "
```

```
   fi
done;

echo "-------------ods-dwd  一致性检查-------------"
# 执行数据源表和目标表关联查询 SQL 语句，查询数据一致的条数
consistent_data_query_result=`hive -e "select count(*) from
gmall.$source_table_name join gmall.$target_table_name
$join_on_sql_str"`
# 取出全表查询数据一致的条数
consistent_data_count=${consistent_data_query_result:3}

echo "---------------ods 层记录条数-----------------"
# 执行查询数据源表的记录条数
source_table_query_result=`hive -e "select count(*) from
gmall.$source_table_name"`
# 取出全表数据源表的记录条数
source_table_count=${source_table_query_result:3}

echo "---------------dwd 层记录条数-----------------"
# 执行查询数据目标表的记录条数
target_table_query_result=`hive -e "select count(*) from
gmall.$target_table_name"`
# 取出全表数据目标表的记录条数
target_table_count=${target_table_query_result:3}

# 将所有数据检验结果插入到表中
hive -e "insert into datacheck.dwd_table_data_check
values('$do_date','gmall','$source_table_name','$source_column
','$target_table_name','$target_column','$consistent_data_coun
t','$source_table_count','$target_table_count')"
```

脚本参数注释：

第 1 个参数：传入时间分区参数(dt)

第 2 个参数：需要进行数据校验源表的表名(source_table_name)

第 3 个参数：需要校验的数据源表字段名称用逗号‘ ，’隔开，例如：col1,col2,col3

第 4 个参数：需要进行数据校验目标表的表名(target_table_name)

第 5 个参数：需要校验的数据目标表字段名称用逗号‘ ，’隔开，例如：col1,col2,col3

给脚本/opt/module/data_check/sh/increment_data_check_public.sh 赋权限：

```
[atguigu@hadoop102 sh]$ chmod 777 table_consistent_check_publi
c.sh
```

脚本执行示例：

```
[atguigu@hadoop102 sh]$ ./table_consistent_check_public.sh
2020-06-14 ods_base_province name dwd_dim_base_province
province_name
```

3.2 DWD 层各表检验

1. 业务数据表数据检验

① 优惠券信息表

```
数据源表：ods_coupon_info
源表字段：id,coupon_name,coupon_type,condition_amount,condition_num,activity_id,benefit_amount,benefit_discount,create_time,range_type,spu_id,tm_id,category3_id,limit_num,operate_time,expire_time
数据目标表：dwd_dim_coupon_info
目标表字段：id,coupon_name,coupon_type,condition_amount,condition_num,activity_id,benefit_amount,benefit_discount,create_time,range_type,spu_id,tm_id,category3_id,limit_num,operate_time,expire_time
分区：dt='2020-06-14'
```

数据检查脚本：

```
[atguigu@hadoop102 sh]$ pwd
/opt/module/data_check/sh
[atguigu@hadoop102 sh]$ /opt/module/data_check/sh/table_consistent_check_public.sh 2020-06-14 ods_coupon_info id,coupon_name,coupon_type,condition_amount,condition_num,activity_id,benefit_amount,benefit_discount,create_time,range_type,spu_id,tm_id,category3_id,limit_num,operate_time,expire_time dwd_dim_coupon_info id,coupon_name,coupon_type,condition_amount,condition_num,activity_id,benefit_amount,benefit_discount,create_time,range_type,spu_id,tm_id,category3_id,limit_num,operate_time,expire_time
```

② 订单明细事实表

```
数据源表：ods_order_detail
源表字段：id,order_id,user_id,sku_id,sku_name,order_price,sku_num,create_time
数据目标表：dwd_fact_order_detail
目标表字段：id,order_id,user_id,sku_id,sku_name,order_price,sku_num,create_time
分区：dt='2020-06-14'
```

数据检查脚本：

```
[atguigu@hadoop102 sh]$ pwd
/opt/module/data_check/sh
[atguigu@hadoop102 sh]$ /opt/module/data_check/sh/table_consistent_check_public.sh 2020-06-14 ods_order_detail id,order_id,user_id,sku_id,sku_name,order_price,sku_num,create_time dwd_fact_order_detail id,order_id,user_id,sku_id,sku_name,order_price,sku_num,create_time
```

③ 支付事实表

```
数据源表：ods_payment_info
源表字段：id,out_trade_no,order_id,user_id,alipay_trade_no,total_amount,subject,payment_type,payment_time
数据目标表：dwd_fact_payment_info
目标表字段：id,out_trade_no,order_id,user_id,alipay_trade_no,payment_amount,subject,payment_type,payment_time
分区：dt='2020-06-14'
```

数据检查脚本：

```
[atguigu@hadoop102 sh]$ pwd
/opt/module/data_check/sh
[atguigu@hadoop102 sh]$ /opt/module/data_check/sh/table_consistent_check_public.sh 2020-06-14 ods_payment_info id,out_trade_n
```

```
o,order_id,user_id,alipay_trade_no,total_amount,subject,paymen
t_type,payment_time dwd_fact_payment_info id,out_trade_no,orde
r_id,user_id,alipay_trade_no,payment_amount,subject,payment_ty
pe,payment_time
```

④ 退款事实表

```
数据源表：ods_order_refund_info
源表字段：id,user_id,order_id,sku_id,refund_type,refund_num,refu
nd_amount,refund_reason_type,create_time
数据目标表：dwd_fact_order_refund_info
目标表字段：id,user_id,order_id,sku_id,refund_type,refund_num,re
fund_amount,refund_reason_type,create_time
分区：dt='2020-06-14'
```

数据检查脚本：

```
[atguigu@hadoop102 sh]$ pwd
/opt/module/data_check/sh
[atguigu@hadoop102 sh]$ /opt/module/data_check/sh/table_consis
tent_check_public.sh 2020-06-14 ods_order_refund_info id,user_i
d,order_id,sku_id,refund_type,refund_num,refund_amount,refund_
reason_type,create_time dwd_fact_order_refund_info id,user_id,
order_id,sku_id,refund_type,refund_num,refund_amount,refund_re
ason_type,create_time
```

⑤ 评价事实表

```
数据源表：ods_comment_info
源表字段：id,user_id,sku_id,spu_id,order_id,appraise,create_time
数据目标表：dwd_fact_comment_info
目标表字段：id,user_id,sku_id,spu_id,order_id,appraise,create_ti
me
分区：dt='2020-06-14'
```

数据检查脚本：

```
[atguigu@hadoop102 sh]$ pwd
/opt/module/data_check/sh
[atguigu@hadoop102 sh]$ /opt/module/data_check/sh/table_consis
tent_check_public.sh 2020-06-14 ods_comment_info id,user_id,sku
_id,spu_id,order_id,appraise,create_time dwd_fact_comment_info
 id,user_id,sku_id,spu_id,order_id,appraise,create_time
```

⑥ 加购事实表

```
数据源表：ods_cart_info
源表字段：id,user_id,sku_id,cart_price,sku_num,sku_name,create_t
ime,operate_time,is_ordered,order_time
数据目标表：dwd_fact_cart_info
目标表字段：id,user_id,sku_id,cart_price,sku_num,sku_name,create
_time,operate_time,is_ordered,order_time
分区：dt='2020-06-14'
```

数据检查脚本：

```
[atguigu@hadoop102 sh]$ pwd
/opt/module/data_check/sh
[atguigu@hadoop102 sh]$ /opt/module/data_check/sh/table_consis
tent_check_public.sh 2020-06-14 ods_cart_info id,user_id,sku_i
d,cart_price,sku_num,sku_name,create_time,operate_time,is_orde
red,order_time dwd_fact_cart_info id,user_id,sku_id,cart_pric
e,sku_num,sku_name,create_time,operate_time,is_ordered,order_t
ime
```

⑦ 收藏事实表

```
数据源表：ods_favor_info
源表字段：id,user_id,sku_id,spu_id,is_cancel,create_time,cancel_
time
数据目标表：dwd_fact_favor_info
目标表字段：id,user_id,sku_id,spu_id,is_cancel,create_time,cance
l_time
分区：dt='2020-06-14'
```

数据检查脚本：

```
[atguigu@hadoop102 sh]$ pwd
/opt/module/data_check/sh
[atguigu@hadoop102 sh]$ /opt/module/data_check/sh/table_consis
tent_check_public.sh 2020-06-14 ods_favor_info id,user_id,sku_i
d,spu_id,is_cancel,create_time,cancel_time dwd_fact_favor_info
 id,user_id,sku_id,spu_id,is_cancel,create_time,cancel_time
```

2. DWD 层数据检查脚本

1）在路径/opt/module/data_check/sh 下创建 dwd 层数据一致性检验 shell 脚本

```
[atguigu@hadoop102 sh]$ vim dwd_data_check.sh
```

2）在脚本中编写如下内容：

```
#!/bin/bash
# 数据所在的日期分区
do_date=$1

/opt/module/data_check/sh/table_consistent_check_public.sh
$do_date ods_coupon_info
id,coupon_name,coupon_type,condition_amount,condition_num,acti
vity_id,benefit_amount,benefit_discount,create_time,range_type
,spu_id,tm_id,category3_id,limit_num,operate_time,expire_time
dwd_dim_coupon_info
id,coupon_name,coupon_type,condition_amount,condition_num,acti
vity_id,benefit_amount,benefit_discount,create_time,range_type
,spu_id,tm_id,category3_id,limit_num,operate_time,expire_time

/opt/module/data_check/sh/table_consistent_check_public.sh
$do_date ods_order_detail
id,order_id,user_id,sku_id,sku_name,order_price,sku_num,create
_time dwd_fact_order_detail
id,order_id,user_id,sku_id,sku_name,order_price,sku_num,create
_time

/opt/module/data_check/sh/table_consistent_check_public.sh
$do_date ods_payment_info
id,out_trade_no,order_id,user_id,alipay_trade_no,total_amount,
subject,payment_type,payment_time dwd_fact_payment_info
id,out_trade_no,order_id,user_id,alipay_trade_no,payment_amoun
t,subject,payment_type,payment_time

/opt/module/data_check/sh/table_consistent_check_public.sh
$do_date ods_order_refund_info
id,user_id,order_id,sku_id,refund_type,refund_num,refund_amoun
t,refund_reason_type,create_time dwd_fact_order_refund_info
id,user_id,order_id,sku_id,refund_type,refund_num,refund_amoun
t,refund_reason_type,create_time
```

```
/opt/module/data_check/sh/table_consistent_check_public.sh
$do_date ods_comment_info
id,user_id,sku_id,spu_id,order_id,appraise,create_time
dwd_fact_comment_info
id,user_id,sku_id,spu_id,order_id,appraise,create_time

/opt/module/data_check/sh/table_consistent_check_public.sh
$do_date ods_cart_info
id,user_id,sku_id,cart_price,sku_num,sku_name,create_time,oper
ate_time,is_ordered,order_time dwd_fact_cart_info
id,user_id,sku_id,cart_price,sku_num,sku_name,create_time,oper
ate_time,is_ordered,order_time

/opt/module/data_check/sh/table_consistent_check_public.sh
$do_date ods_favor_info
id,user_id,sku_id,spu_id,is_cancel,create_time,cancel_time
dwd_fact_favor_info
id,user_id,sku_id,spu_id,is_cancel,create_time,cancel_time
```

脚本参数注释：

第 1 个参数：传入时间分区参数(dt)

3）给脚本/opt/module/data_check/sh/dwd_data_check.sh 赋权限：

```
[atguigu@hadoop102 sh]$ chmod 777 dwd_data_check.sh
```

脚本执行示例：

```
[atguigu@hadoop102 sh]$ ./dwd_data_check.sh 2020-06-15
```

第 4 章 DWS 层数据校验

4.1 DWS 层数据质量校验

由于 DWS 层数据质量无法从一致性进行判断，只能通过表记录数以及空值记录数等维度判断， 所以在 DWS 将检验当日宽表记录数量是否在合理的范围之内，同时检验关键字段为空的记录的记录数量。

4.2 DWS 层数据校验脚本

1. 数据校验涉及表

每日会员行为
每日商品行为
每日活动行为
每日地区行为

2.数据质量校验脚本

1）在路径/opt/module/data_check/sh 下创建 ODS 层数据检查脚本

```
[atguigu@atguigu sh]$ pwd
/opt/module/data_check/sh
[atguigu@atguigu sh]$ vim dws_data_check.sh
```

在脚本中编写如下内容：

```
#!/bin/bash
```

```
data_date=$1

# 每日会员行为
/opt/module/data_check/sh/total_data_check_public.sh
$data_date dws_user_action_daycount
user_id,login_count,cart_count,order_count

# 每日商品行为
/opt/module/data_check/sh/total_data_check_public.sh
$data_date dws_sku_action_daycount
sku_id,order_count,order_num,order_amount,payment_count,paymen
t_num,payment_amount,refund_count

# 每日活动行为
/opt/module/data_check/sh/total_data_check_public.sh
$data_date dws_activity_info_daycount
id,activity_name,activity_type,start_time,end_time,create_time
,display_count

# 每日地区统计
/opt/module/data_check/sh/total_data_check_public.sh
$data_date dws_area_stats_daycount
id,province_name,area_code,iso_code,region_id,region_name,logi
n_count,order_count,order_amount,payment_count,payment_amount
```

2）给脚本/opt/module/data_check/sh/dws_data_check.sh 赋权限：

```
[atguigu@atguigu sh]$ chmod 777 dws_data_check.sh
```

3）执行/opt/module/data_check/sh/dws_data_check.sh，检验 2020-06-14 当日宽表数据质量。

```
[atguigu@atguigu sh]$ ./dws_data_check.sh 2020-16-14
```

第 5 章 DWT 层数据校验

5.1 DWT 层数据质量校验方法

在宽表阶段数据已经经过了一定的判断、过滤和变换等操作，因此在 DWT 层也将检验当日宽表记录数量是否在合理的范围之内，同时检验关键字段为空的记录的记录数量。

5.2 宽表校验脚本

1. 数据校验涉及表

设备主题宽表
会员主题宽表
商品主题宽表
活动主题宽表
地区主题宽表

2.数据质量校验脚本

1）在路径/opt/module/data_check/sh 下创建 ODS 层数据检查脚本

```
[atguigu@atguigu sh]$ pwd
/opt/module/data_check/sh
```

```
[atguigu@atguigu sh]$ vim dwt_data_check.sh
```

在脚本中编写如下内容：

```
#!/bin/bash
data_date=$1

# 全量检查
# 设备主题宽表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date dwt_uv_topic
mid_id,brand,model,login_date_first,login_date_last,login_day_
count,login_count

# 会员主题宽表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date dwt_user_topic
user_id,login_date_first,login_date_last,login_count,login_las
t_30d_count,order_date_first,order_date_last,order_count,order
_amount,order_last_30d_count,order_last_30d_amount,payment_dat
e_first,payment_date_last,payment_count,payment_amount,payment
_last_30d_count,payment_last_30d_amount

# 商品主题宽表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date dwt_sku_topic
sku_id,spu_id,order_last_30d_count,order_last_30d_num,order_la
st_30d_amount,order_count,order_num,order_amount

# 活动主题宽表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date dwt_activity_topic
id,activity_name,activity_type,start_time,end_time,create_time
,display_day_count,order_day_count,order_day_amount,payment_da
y_count,payment_day_amount,display_count,order_count,order_amo
unt,payment_count,payment_amount

# 地区主题宽表
/opt/module/data_check/sh/total_data_check_public.sh
$data_date
id,province_name,area_code,iso_code,region_id,region_name,logi
n_day_count,login_last_30d_count,order_day_count,order_day_amo
unt,order_last_30d_count,order_last_30d_amount,payment_day_cou
nt,payment_day_amount,payment_last_30d_count,payment_last_30d_
amount
```

2）给脚本/opt/module/data_check/sh/dwt_data_check.sh 赋权限：

```
[atguigu@atguigu sh]$ chmod 777 dwt_data_check.sh
```

3）执行/opt/module/data_check/sh/dwt_data_check.sh，检验 2020-06-15 当日宽表数据质量。

```
[atguigu@atguigu sh]$ ./dwt_data_check.sh 2020-16-15
```

第 6 章 ADS 层数据校验

数据仓库中 ADS 层数据是经过高度聚合计算的具体指标，因此无法从技术层面进行判

断数据是否健康。需要通过校对对各个指标的数值是否在合理的范围之内进行校验，进行定制化数据校验。由于 ADS 层涉及的需求无法一一涉及，因此在这针对一个需求进行分析。

1）在 Hive 中创建表数据质量校验记录表，记录数据校验的各个指标：

```
[atguigu@atguigu data_check]$ hive
```

创建数据库，用于存放数据质量校验的结果数据：

```
hive (default)> create database datacheck;
```

创建数据表，用于存放 ods 层的数据检验结果：

```
hive (datacheck)> create table datacheck.ads_table_data_check(
    data_date string comment '数据时间分区 dt',
    database_name string comment '库名',
    table_name string comment '表名',
    column_name string comment '指标名',
    healthy_value string comment '该指标合理值',
    now_value bigint comment '该指标当前值',
    is_healthy bigint comment '该指标是否合理：1 合理/0 不合理'
);
hive (datacheck)>quit;
```

3）在路径/opt/module/data_check/sh 下创建数据检验增量表通用 shell 脚本

```
[atguigu@hadoop102 sh]$ vim ads_data_check.sh
```

在脚本中编写如下内容：

```
#!/bin/bash
# 增量数据所在的日期分区
do_date=$1

hive -e "insert into datacheck.ads_table_data_check select
    temp.data_date,
    temp.database_name,
    temp.table_name,
    temp.column_name,
    temp.healthy_value,
    temp.new_mid_count,
    temp.is_healthy
from (
    select
        "$do_date" as data_date,
        "gmall" as database_name,
        "ads_new_mid_count" as table_name,
        "new_mid_count" as column_name,
        "大于 3" as healthy_value,
        new_mid_count,
        if(new_mid_count>3,1,0) as is_healthy
    from gmall.ads_new_mid_count
) as temp
"
```

脚本参数注释：

第一个参数：传入时间分区参数(dt)

给脚本/opt/module/data_check/sh/ads_data_check.sh 赋权限：

```
[atguigu@hadoop102 sh]$ chmod 777 ads_data_check.sh
```

脚本执行示例：

```
[atguigu@hadoop102 sh]$
./ads_data_check.sh 2020-06-14
```

第 7 章 数据质量之 Griffin

由于 Griffin 有着较为严重的版本依赖，因此无法在最新版本的数据仓库架构中兼容进去。但若是使用 2.x 版本的 Spark 和 hadoop，可通过如下案例进行 Griffin 的数据质量监控。文档中同时附上了 DWD 层与 ODS 层使用 Griffin 进行一致性数据检验的章节。文档如下：

质量监控之Griffin.docx