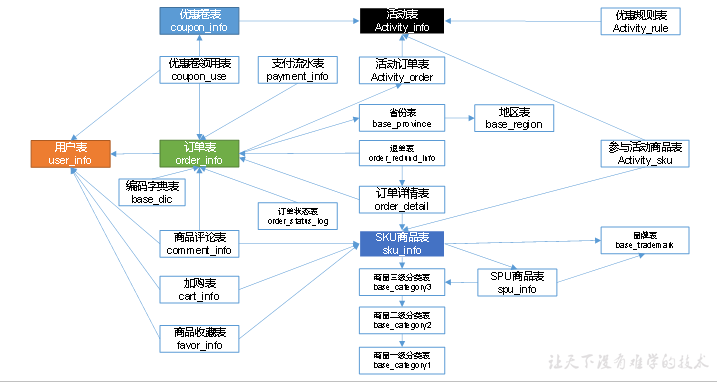
系统业务数据仓库



### 1.3.1 订单表（order\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 订单编号 | |
| consignee | 收货人 | |
| consignee\_tel | 收件人电话 | |
| final\_total\_amount | 总金额 | |
| order\_status | 订单状态 | |
| user\_id | 用户id | |
| delivery\_address | 送货地址 | |
| order\_comment | 订单备注 | |
| out\_trade\_no | 订单交易编号（第三方支付用) | |
| trade\_body | 订单描述(第三方支付用) | |
| create\_time | 创建时间 | |
| operate\_time | 操作时间 | |
| expire\_time | 失效时间 | |
| tracking\_no | 物流单编号 | |
| parent\_order\_id | 父订单编号 | |
| img\_url | 图片路径 | |
| province\_id | 地区 | |
| benefit\_reduce\_amount | 优惠金额 | |
| original\_total\_amount | 原价金额 | |
| feight\_fee | 运费 | |

### 1.3.2 订单详情表（order\_detail）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 订单编号 | |
| order\_id | 订单号 | |
| sku\_id | 商品id | |
| sku\_name | sku名称（冗余) | |
| img\_url | 图片名称（冗余) | |
| order\_price | 商品价格(下单时sku价格,未知是单价还是总价） | |
| sku\_num | 商品数量 | |
| create\_time | 创建时间 | |

### 1.3.3 SKU商品表（sku\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | skuId | |
| spu\_id | spuid | |
| price | 价格 | |
| sku\_name | 商品名称 | |
| sku\_desc | 商品描述 | |
| weight | 重量 | |
| tm\_id | 品牌id | |
| category3\_id | 品类id | |
| sku\_default\_img | 默认显示图片(冗余) | |
| create\_time | 创建时间 | |

### 1.3.4 用户表（user\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 用户id | |
| login\_name | 用户名称 | |
| nick\_name | 用户昵称 | |
| passwd | 用户密码 | |
| name | 姓名 | |
| phone\_num | 手机号 | |
| email | 邮箱 | |
| head\_img | 头像 | |
| user\_level | 用户级别 | |
| birthday | 生日 | |
| gender | 性别M男,F女 | |
| create\_time | 创建时间 | |
| operate\_time | 操作时间 | |

### 1.3.5 商品一级分类表（base\_category1）

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

### 1.3.6 商品二级分类表（base\_category2）

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

### 1.3.7 商品三级分类表（base\_category3）

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

### 1.3.8 支付流水表（payment\_info）

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

### 1.3.9 省份表（base\_province）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | id | |
| name | 省份名称 | |
| region\_id | 地区ID | |
| area\_code | 地区编码 | |
| iso\_code | 国际编码 | |

### 1.3.10 地区表（base\_region）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 大区id | |
| region\_name | 大区名称 | |

### 1.3.11 品牌表（base\_trademark）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| tm\_id | 品牌id | |
| tm\_name | 品牌名称 | |

### 1.3.12 订单状态表（order\_status\_log）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 编号 | |
| order\_id | 订单编号 | |
| order\_status | 订单状态 | |
| operate\_time | 操作时间 | |

### 1.3.13 SPU商品表（spu\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 商品id | |
| spu\_name | spu商品名称 | |
| description | 商品描述(后台简述） | |
| category3\_id | 三级分类id | |
| tm\_id | 品牌id | |

### 1.3.14 商品评论表（comment\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 编号 | |
| user\_id | 用户名称 | |
| sku\_id | 商品id | |
| spu\_id | spu\_id | |
| order\_id | 订单编号 | |
| appraise | 评价 1 好评 2 中评 3 差评 | |
| comment\_txt | 评价内容 | |
| create\_time | 创建时间 | |

### 1.3.15 退单表（order\_refund\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 编号 | |
| order\_id | 订单编号 | |
| sku\_id | skuid | |
| refund\_type | 退款类型 | |
| refund\_amount | 退款金额 | |
| refund\_reason\_type | 原因类型 | |
| refund\_reason\_txt | 原因内容 | |
| create\_time | 创建时间 | |

### 1.3.16 加购表（cart\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 编号 | |
| user\_id | 用户id | |
| sku\_id | SKU商品 | |
| cart\_price | 放入购物车时价格 | |
| sku\_num | 数量 | |
| img\_url | 图片文件 | |
| sku\_name | sku名称 (冗余) | |
| create\_time | 创建时间 | |
| operate\_time | 修改时间 | |
| is\_ordered | 是否已经下单 | |
| order\_time | 下单时间 | |

### 1.3.17 商品收藏表（favor\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 编号 | |
| user\_id | 用户名称 | |
| sku\_id | 商品id | |
| spu\_id | spu\_id | |
| is\_cancel | 是否已取消 0 正常 1 已取消 | |
| create\_time | 创建时间 | |
| cancel\_time | 修改时间 | |

### 1.3.18 优惠券领用表（coupon\_use）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 编号 | |
| coupon\_id | 购物券ID | |
| user\_id | 用户ID | |
| order\_id | 订单ID | |
| coupon\_status | 购物券状态 | |
| get\_time | 领券时间 | |
| using\_time | 使用时间 | |
| used\_time | 支付时间 | |
| expire\_time | 过期时间 | |

### 1.3.19 优惠券表（coupon\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 购物券编号 | |
| coupon\_name | 购物券名称 | |
| coupon\_type | 购物券类型 1 现金券 2 折扣券 3 满减券 4 满件打折券 | |
| condition\_amount | 满额数 | |
| condition\_num | 满件数 | |
| activity\_id | 活动编号 | |
| benefit\_amount | 减金额 | |
| benefit\_discount | 折扣 | |
| create\_time | 创建时间 | |
| range\_type | 范围类型 1、商品 2、品类 3、品牌 | |
| spu\_id | 商品id | |
| tm\_id | 品牌id | |
| category3\_id | 品类id | |
| limit\_num | 最多领用次数 | |
| operate\_time | 修改时间 | |
| expire\_time | 过期时间 | |

### 1.3.20 活动表（activity\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 活动id | |
| activity\_name | 活动名称 | |
| activity\_type | 活动类型 | |
| activity\_desc | 活动描述 | |
| start\_time | 开始时间 | |
| end\_time | 结束时间 | |
| create\_time | 创建时间 | |

### 1.3.21 活动订单关联表（activity\_order）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 编号 | |
| activity\_id | 活动id | |
| order\_id | 订单编号 | |
| create\_time | 发生日期 | |

### 1.3.22 优惠规则表（activity\_rule）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 编号 | |
| activity\_id | 活动id | |
| condition\_amount | 满减金额 | |
| condition\_num | 满减件数 | |
| benefit\_amount | 优惠金额 | |
| benefit\_discount | 优惠折扣 | |
| benefit\_level | 优惠级别 | |

### 1.3.23 编码字典表（base\_dic）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| dic\_code | 编号 | |
| dic\_name | 编码名称 | |
| parent\_code | 父编号 | |
| create\_time | 创建日期 | |
| operate\_time | 修改日期 | |

### 1.3.24 活动参与商品表（activity\_sku）（暂不导入）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| id | 编号 | |
| activity\_id | 活动id | |
| sku\_id | 满减金额 | |
| create\_time | 创建时间 | |

## 1.4 时间相关表

### 1.4.1 时间表（date\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| date\_id | 日期id | |
| week\_id | 周id | |
| week\_day | 周 | |
| day | 日 | |
| month | 月 | |
| quarter | 季度 | |
| year | 年 | |
| is\_workday | 是否是周末 | |
| holiday\_id | 假期id | |

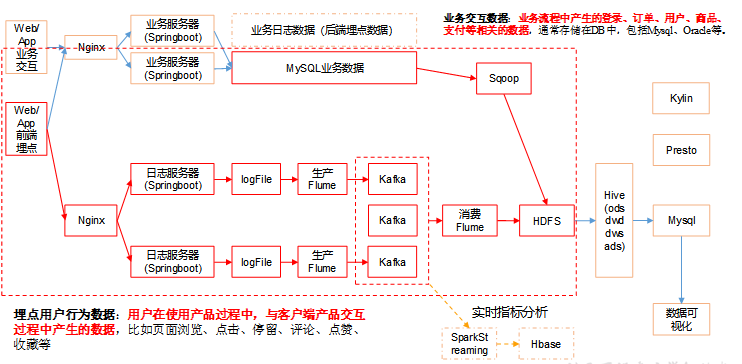
### 1.4.2 假期表（holiday\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| holiday\_id | 假期id | |
| holiday\_name | 假期名称 | |

### 1.4.3 假期年表（holiday\_year）

|  |  |  |
| --- | --- | --- |
| 标签 | | 含义 |
| holiday\_id | 假期id | |
| holiday\_name | 假期名称 | |
| start\_date\_id | 假期开始时间 | |
| end\_date\_id | 假期结束时间 | |

# 第2章 业务数据采集模块



## 2.1 MySQL安装

### 2.1.1 安装包准备

1）查看MySQL是否安装，如果安装了，卸载MySQL

（1）查看

[root@hadoop102 桌面]# rpm -qa|grep mysql

mysql-libs-5.1.73-7.el6.x86\_64

（2）卸载

[root@hadoop102 桌面]# rpm -e --nodeps mysql-libs-5.1.73-7.el6.x86\_64

2）解压mysql-libs.zip文件到当前目录

[root@hadoop102 software]# unzip mysql-libs.zip

[root@hadoop102 software]# ls

mysql-libs.zip

mysql-libs

3）进入到mysql-libs文件夹下

[root@hadoop102 mysql-libs]# ll

总用量 76048

-rw-r--r--. 1 root root 18509960 3月 26 2015 MySQL-client-5.6.24-1.el6.x86\_64.rpm

-rw-r--r--. 1 root root 3575135 12月 1 2013 mysql-connector-java-5.1.27.tar.gz

-rw-r--r--. 1 root root 55782196 3月 26 2015 MySQL-server-5.6.24-1.el6.x86\_64.rpm

### 2.1.2 安装MySql服务器

1）安装mysql服务端

[root@hadoop102 mysql-libs]# rpm -ivh MySQL-server-5.6.24-1.el6.x86\_64.rpm

2）查看产生的随机密码

[root@hadoop102 mysql-libs]# cat /root/.mysql\_secret

OEXaQuS8IWkG19Xs

3）查看mysql状态

[root@hadoop102 mysql-libs]# service mysql status

4）启动mysql

[root@hadoop102 mysql-libs]# service mysql start

### 2.1.3 安装MySql客户端

1）安装mysql客户端

[root@hadoop102 mysql-libs]# rpm -ivh MySQL-client-5.6.24-1.el6.x86\_64.rpm

2）链接mysql

[root@hadoop102 mysql-libs]# mysql -uroot -pOEXaQuS8IWkG19Xs

3）修改密码

mysql>SET PASSWORD=PASSWORD('000000');

4）退出mysql

mysql>exit

### 2.1.4 MySql中user表中主机配置

配置只要是root用户+密码，在任何主机上都能登录MySQL数据库。

1）进入mysql

[root@hadoop102 mysql-libs]# mysql -uroot -p000000

2）显示数据库

mysql>show databases;

3）使用mysql数据库

mysql>use mysql;

4）展示mysql数据库中的所有表

mysql>show tables;

5）展示user表的结构

mysql>desc user;

6）查询user表

mysql>select User, Host, Password from user;

7）修改user表，把Host表内容修改为%

mysql>update user set host='%' where host='localhost';

8）删除root用户的其他host

mysql>

delete from user where Host='hadoop102';

delete from user where Host='127.0.0.1';

delete from user where Host='::1';

9）刷新

mysql>flush privileges;

10）退出

mysql>quit;

## 2.2 Sqoop安装

### 2.2.1 下载并解压

1）下载地址：<http://mirrors.hust.edu.cn/apache/sqoop/1.4.6/>

2）上传安装包sqoop-1.4.6.bin\_\_hadoop-2.0.4-alpha.tar.gz到hadoop102的/opt/software路径中

3）解压sqoop安装包到指定目录，如：

[woaini@hadoop102 software]$ tar -zxf sqoop-1.4.6.bin\_\_hadoop-2.0.4-alpha.tar.gz -C /opt/module/

4）解压sqoop安装包到指定目录，如：

[woaini@hadoop102 module]$ mv sqoop-1.4.6.bin\_\_hadoop-2.0.4-alpha/ sqoop

### 2.2.2 修改配置文件

1) 进入到/opt/module/sqoop/conf目录，重命名配置文件

[woaini@hadoop102 conf]$ mv sqoop-env-template.sh sqoop-env.sh

2) 修改配置文件

[woaini@hadoop102 conf]$ vim sqoop-env.sh

增加如下内容

export HADOOP\_COMMON\_HOME=/opt/module/hadoop-2.7.2

export HADOOP\_MAPRED\_HOME=/opt/module/hadoop-2.7.2

export HIVE\_HOME=/opt/module/hive

export ZOOKEEPER\_HOME=/opt/module/zookeeper-3.4.10

export ZOOCFGDIR=/opt/module/zookeeper-3.4.10/conf

export HBASE\_HOME=/opt/module/hbase

### 2.2.3 拷贝JDBC驱动

1）进入到/opt/software/mysql-libs路径，解压mysql-connector-java-5.1.27.tar.gz到当前路径

[woaini@hadoop102 mysql-libs]$ tar -zxvf mysql-connector-java-5.1.27.tar.gz

2）进入到/opt/software/mysql-libs/mysql-connector-java-5.1.27路径，拷贝jdbc驱动到sqoop的lib目录下。

[woaini@hadoop102 mysql-connector-java-5.1.27]$ cp mysql-connector-java-5.1.27-bin.jar /opt/module/sqoop/lib/

### 2.2.4 验证Sqoop

我们可以通过某一个command来验证sqoop配置是否正确：

[woaini@hadoop102 sqoop]$ bin/sqoop help

出现一些Warning警告（警告信息已省略），并伴随着帮助命令的输出：

Available commands:

codegen Generate code to interact with database records

create-hive-table Import a table definition into Hive

eval Evaluate a SQL statement and display the results

export Export an HDFS directory to a database table

help List available commands

import Import a table from a database to HDFS

import-all-tables Import tables from a database to HDFS

import-mainframe Import datasets from a mainframe server to HDFS

job Work with saved jobs

list-databases List available databases on a server

list-tables List available tables in a database

merge Merge results of incremental imports

metastore Run a standalone Sqoop metastore

version Display version information

### 2.2.5 测试Sqoop是否能够成功连接数据库

[woaini@hadoop102 sqoop]$ bin/sqoop list-databases --connect jdbc:mysql://hadoop102:3306/ --username root --password 000000

出现如下输出：

information\_schema

metastore

mysql

oozie

performance\_schema

## 2.3 业务数据生成

### 2.3.1 连接MySQL

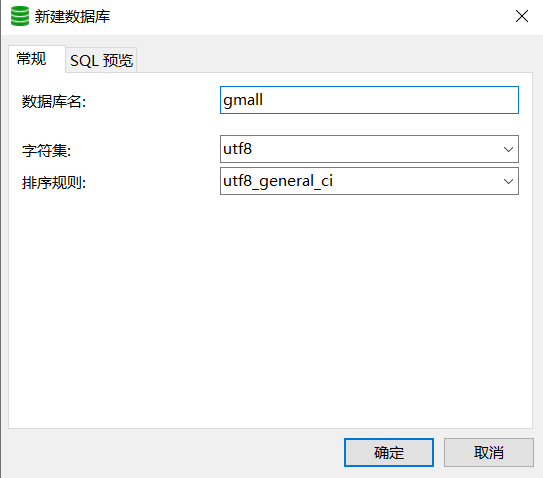
通过MySQL操作可视化工具SQLyog连接MySQL。



### 2.3.2 建表语句

1）通过SQLyog创建数据库gmall

2）设置数据库编码



3）导入数据库结构脚本（**gmall2020-03-16.sql**）

### 2.3.2 生成业务数据

1）在hadoop102的/opt/module/目录下创建db\_log文件夹

[woaini@hadoop102 module]$ mkdir db\_log/

2）把gmall-mock-db-2020-03-16-SNAPSHOT.jar和 application.properties上传到hadoop102的/opt/module/db\_log路径上。

3）根据需求修改application.properties相关配置

logging.level.root=info

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.url=jdbc:mysql://hadoop102:3306/gmall?characterEncoding=utf-8&useSSL=false&serverTimezone=GMT%2B8

spring.datasource.username=root

spring.datasource.password=000000

logging.pattern.console=%m%n

mybatis-plus.global-config.db-config.field-strategy=not\_null

#业务日期

mock.date=2020-03-10

#是否重置

mock.clear=1

#是否生成新用户

mock.user.count=50

#男性比例

mock.user.male-rate=20

#收藏取消比例

mock.favor.cancel-rate=10

#收藏数量

mock.favor.count=100

#购物车数量

mock.cart.count=10

#每个商品最多购物个数

mock.cart.sku-maxcount-per-cart=3

#用户下单比例

mock.order.user-rate=80

#用户从购物中购买商品比例

mock.order.sku-rate=70

#是否参加活动

mock.order.join-activity=1

#是否使用购物券

mock.order.use-coupon=1

#购物券领取人数

mock.coupon.user-count=10

#支付比例

mock.payment.rate=70

#支付方式 支付宝：微信 ：银联

mock.payment.payment-type=30:60:10

#评价比例 好：中：差：自动

mock.comment.appraise-rate=30:10:10:50

#退款原因比例：质量问题 商品描述与实际描述不一致 缺货 号码不合适 拍错 不想买了 其他

mock.refund.reason-rate=30:10:20:5:15:5:5

4）并在该目录下执行，如下命令，生成2020-03-10日期数据：

[woaini@hadoop102 db\_log]$ java -jar gmall-mock-db-2020-03-16-SNAPSHOT.jar

5）在配置文件application.properties中修改

mock.date=2020-03-11

mock.clear=0

6）再次执行命令，生成2020-03-11日期数据：

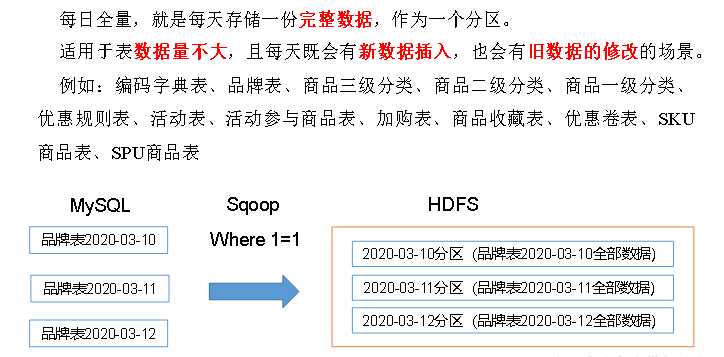
[woaini@hadoop102 db\_log]$ java -jar gmall-mock-db-2020-03-16-SNAPSHOT.jar

## 2.4 同步策略

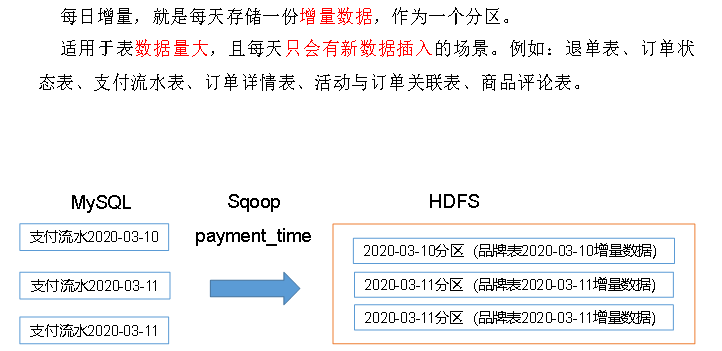
数据同步策略的类型包括：全量表、增量表、新增及变化表

* 全量表：存储完整的数据。
* 增量表：存储新增加的数据。
* 新增及变化表：存储新增加的数据和变化的数据。
* 特殊表：只需要存储一次。

### 2.4.1 全量同步策略



### 2.4.2 增量同步策略



### 2.4.3 新增及变化策略

每日新增及变化，就是存储创建时间和操作时间都是今天的数据。

适用场景为，**表的数据量大，既会有新增，又会有变化**。

例如：用户表、订单表、优惠卷领用表。

### 2.4.4 特殊策略

某些特殊的维度表，可不必遵循上述同步策略。

**1）客观世界维度**

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

**2）日期维度**

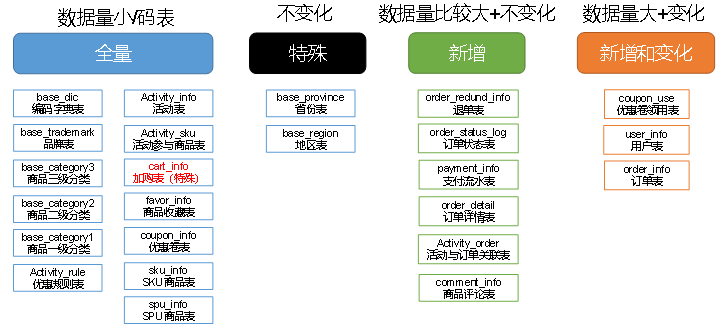
日期维度可以一次性导入一年或若干年的数据。

**3）地区维度**

省份表、地区表

## 2.5 业务数据导入HDFS

### 2.5.1 分析表同步策略



### 2.5.2 脚本编写

1）在/home/woaini/bin目录下创建

[woaini@hadoop102 bin]$ vim mysql\_to\_hdfs.sh

添加如下内容：

#! /bin/bash

sqoop=/opt/module/sqoop/bin/sqoop

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

if [[ -n "$2" ]]; then

do\_date=$2

fi

import\_data(){

$sqoop import \

--connect jdbc:mysql://hadoop102:3306/gmall \

--username root \

--password 000000 \

--target-dir /origin\_data/gmall/db/$1/$do\_date \

--delete-target-dir \

--query "$2 and \$CONDITIONS" \

--num-mappers 1 \

--fields-terminated-by '\t' \

--compress \

--compression-codec lzop \

--null-string '\\N' \

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

hadoop jar /opt/module/hadoop-2.7.2/share/hadoop/common/hadoop-lzo-0.4.20.jar com.hadoop.compression.lzo.DistributedLzoIndexer /origin\_data/gmall/db/$1/$do\_date

#建立索引和索引存放的目录

}

import\_order\_info(){

import\_data order\_info "select

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

from order\_info

where (date\_format(create\_time,'%Y-%m-%d')='$do\_date'

or date\_format(operate\_time,'%Y-%m-%d')='$do\_date')"

}

import\_coupon\_use(){

import\_data coupon\_use "select

id,

coupon\_id,

user\_id,

order\_id,

coupon\_status,

get\_time,

using\_time,

used\_time

from coupon\_use

where (date\_format(get\_time,'%Y-%m-%d')='$do\_date'

or date\_format(using\_time,'%Y-%m-%d')='$do\_date'

or date\_format(used\_time,'$Y-%m-%d')='$do\_date')"

}

import\_order\_status\_log(){

import\_data order\_status\_log "select

id,

order\_id,

order\_status,

operate\_time

from order\_status\_log

where date\_format(operate\_time,'%Y-%m-%d')='$do\_date'"

}

import\_activity\_order(){

import\_data activity\_order "select

id,

activity\_id,

order\_id,

create\_time

from activity\_order

where date\_format(create\_time,'%Y-%m-%d')='$do\_date'"

}

import\_user\_info(){

import\_data "user\_info" "select

id,

name,

birthday,

gender,

email,

user\_level,

create\_time,

operate\_time

from user\_info

where (DATE\_FORMAT(create\_time,'%Y-%m-%d')='$do\_date'

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

}

import\_order\_detail(){

import\_data order\_detail "select

od.id,

order\_id,

user\_id,

sku\_id,

sku\_name,

order\_price,

sku\_num,

od.create\_time

from order\_detail od

join order\_info oi

on od.order\_id=oi.id

where DATE\_FORMAT(od.create\_time,'%Y-%m-%d')='$do\_date'"

}

import\_payment\_info(){

import\_data "payment\_info" "select

id,

out\_trade\_no,

order\_id,

user\_id,

alipay\_trade\_no,

total\_amount,

subject,

payment\_type,

payment\_time

from payment\_info

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

}

import\_comment\_info(){

import\_data comment\_info "select

id,

user\_id,

sku\_id,

spu\_id,

order\_id,

appraise,

comment\_txt,

create\_time

from comment\_info

where date\_format(create\_time,'%Y-%m-%d')='$do\_date'"

}

import\_order\_refund\_info(){

import\_data order\_refund\_info "select

id,

user\_id,

order\_id,

sku\_id,

refund\_type,

refund\_num,

refund\_amount,

refund\_reason\_type,

create\_time

from order\_refund\_info

where date\_format(create\_time,'%Y-%m-%d')='$do\_date'"

}

import\_sku\_info(){

import\_data sku\_info "select

id,

spu\_id,

price,

sku\_name,

sku\_desc,

weight,

tm\_id,

category3\_id,

create\_time

from sku\_info where 1=1"

}

import\_base\_category1(){

import\_data "base\_category1" "select

id,

name

from base\_category1 where 1=1"

}

import\_base\_category2(){

import\_data "base\_category2" "select

id,

name,

category1\_id

from base\_category2 where 1=1"

}

import\_base\_category3(){

import\_data "base\_category3" "select

id,

name,

category2\_id

from base\_category3 where 1=1"

}

import\_base\_province(){

import\_data base\_province "select

id,

name,

region\_id,

area\_code,

iso\_code

from base\_province

where 1=1"

}

import\_base\_region(){

import\_data base\_region "select

id,

region\_name

from base\_region

where 1=1"

}

import\_base\_trademark(){

import\_data base\_trademark "select

tm\_id,

tm\_name

from base\_trademark

where 1=1"

}

import\_spu\_info(){

import\_data spu\_info "select

id,

spu\_name,

category3\_id,

tm\_id

from spu\_info

where 1=1"

}

import\_favor\_info(){

import\_data favor\_info "select

id,

user\_id,

sku\_id,

spu\_id,

is\_cancel,

create\_time,

cancel\_time

from favor\_info

where 1=1"

}

import\_cart\_info(){

import\_data cart\_info "select

id,

user\_id,

sku\_id,

cart\_price,

sku\_num,

sku\_name,

create\_time,

operate\_time,

is\_ordered,

order\_time

from cart\_info

where 1=1"

}

import\_coupon\_info(){

import\_data coupon\_info "select

id,

coupon\_name,

coupon\_type,

condition\_amount,

condition\_num,

activity\_id,

benefit\_amount,

benefit\_discount,

create\_time,

range\_type,

spu\_id,

tm\_id,

category3\_id,

limit\_num,

operate\_time,

expire\_time

from coupon\_info

where 1=1"

}

import\_activity\_info(){

import\_data activity\_info "select

id,

activity\_name,

activity\_type,

start\_time,

end\_time,

create\_time

from activity\_info

where 1=1"

}

import\_activity\_rule(){

import\_data activity\_rule "select

id,

activity\_id,

condition\_amount,

condition\_num,

benefit\_amount,

benefit\_discount,

benefit\_level

from activity\_rule

where 1=1"

}

import\_base\_dic(){

import\_data base\_dic "select

dic\_code,

dic\_name,

parent\_code,

create\_time,

operate\_time

from base\_dic

where 1=1"

}

case $1 in

"order\_info")

import\_order\_info

;;

"base\_category1")

import\_base\_category1

;;

"base\_category2")

import\_base\_category2

;;

"base\_category3")

import\_base\_category3

;;

"order\_detail")

import\_order\_detail

;;

"sku\_info")

import\_sku\_info

;;

"user\_info")

import\_user\_info

;;

"payment\_info")

import\_payment\_info

;;

"base\_province")

import\_base\_province

;;

"base\_region")

import\_base\_region

;;

"base\_trademark")

import\_base\_trademark

;;

"activity\_info")

import\_activity\_info

;;

"activity\_order")

import\_activity\_order

;;

"cart\_info")

import\_cart\_info

;;

"comment\_info")

import\_comment\_info

;;

"coupon\_info")

import\_coupon\_info

;;

"coupon\_use")

import\_coupon\_use

;;

"favor\_info")

import\_favor\_info

;;

"order\_refund\_info")

import\_order\_refund\_info

;;

"order\_status\_log")

import\_order\_status\_log

;;

"spu\_info")

import\_spu\_info

;;

"activity\_rule")

import\_activity\_rule

;;

"base\_dic")

import\_base\_dic

;;

"first")

import\_base\_category1

import\_base\_category2

import\_base\_category3

import\_order\_info

import\_order\_detail

import\_sku\_info

import\_user\_info

import\_payment\_info

import\_base\_province

import\_base\_region

import\_base\_trademark

import\_activity\_info

import\_activity\_order

import\_cart\_info

import\_comment\_info

import\_coupon\_use

import\_coupon\_info

import\_favor\_info

import\_order\_refund\_info

import\_order\_status\_log

import\_spu\_info

import\_activity\_rule

import\_base\_dic

;;

"all")

import\_base\_category1

import\_base\_category2

import\_base\_category3

import\_order\_info

import\_order\_detail

import\_sku\_info

import\_user\_info

import\_payment\_info

import\_base\_trademark

import\_activity\_info

import\_activity\_order

import\_cart\_info

import\_comment\_info

import\_coupon\_use

import\_coupon\_info

import\_favor\_info

import\_order\_refund\_info

import\_order\_status\_log

import\_spu\_info

import\_activity\_rule

import\_base\_dic

;;

esac

说明1：

[ -n 变量值 ] 判断变量的值，是否为空

-- 变量的值，非空，返回true

-- 变量的值，为空，返回false

说明2：

查看date命令的使用，[woaini@hadoop102 ~]$ date --help

2）修改脚本权限

[woaini@hadoop102 bin]$ chmod 777 mysql\_to\_hdfs.sh

3）初次导入

[woaini@hadoop102 bin]$ mysql\_to\_hdfs.sh first 2020-03-10

4）每日导入

[woaini@hadoop102 bin]$ mysql\_to\_hdfs.sh all 2020-03-11

### 2.5.3 项目经验

Hive中的Null在底层是以“\N”来存储，而MySQL中的Null在底层就是Null，为了保证数据两端的一致性。在导出数据时采用--input-null-string和--input-null-non-string两个参数。导入数据时采用--null-string和--null-non-string。

# 第3章 数据环境准备

## 3.1 安装Hive2.3

1）上传apache-hive-2.3.0-bin.tar.gz 到/opt/software目录下，并解压到/opt/module

[woaini@hadoop102 software]$ tar -zxvf apache-hive-2.3.6-bin.tar.gz -C /opt/module/

2）修改apache-hive-2.3.6-bin名称为hive

[woaini@hadoop102 module]$ mv apache-hive-2.3.6-bin hive

3）将Mysql的mysql-connector-java-5.1.27-bin.jar拷贝到/opt/module/hive/lib/

[woaini@hadoop102 module]$ cp /opt/software/mysql-libs/mysql-connector-java-5.1.27/mysql-connector-java-5.1.27-bin.jar /opt/module/hive/lib/

4）在/opt/module/hive/conf路径上，创建hive-site.xml文件

[woaini@hadoop102 conf]$ vim hive-site.xml

添加如下内容

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://hadoop102:3306/metastore?createDatabaseIfNotExist=true</value>

<description>JDBC connect string for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

<description>Driver class name for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

<description>username to use against metastore database</description>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>000000</value>

<description>password to use against metastore database</description>

</property>

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/user/hive/warehouse</value>

<description>location of default database for the warehouse</description>

</property>

<property>

<name>hive.cli.print.header</name>

<value>true</value>

</property>

<property>

<name>hive.cli.print.current.db</name>

<value>true</value>

</property>

<property>

<name>hive.metastore.schema.verification</name>

<value>false</value>

</property>

<property>

<name>datanucleus.schema.autoCreateAll</name>

<value>true</value>

</property>

<property>

<name>hive.metastore.uris</name>

<value>thrift://hadoop102:9083</value>

</property>

</configuration>

注意：hive安装在哪个服务器节点，thrift://hadoop102:9083中的主机名就更换为相应的主机名。

3）启动服务

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

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

注意：hive2.x版本需要启动两个服务metastore和hiveserver2，否则会报错Exception in thread "main" java.lang.RuntimeException: org.apache.hadoop.hive.ql.metadata.HiveException: java.lang.RuntimeException: Unable to instantiate org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClient

4）服务启动完毕后在启动Hive

[woaini@hadoop102 hive]$ bin/hive

## 3.2 Hive集成引擎Tez

Tez是一个Hive的运行引擎，性能优于MR。为什么优于MR呢？看下图。



用Hive直接编写MR程序，假设有四个有依赖关系的MR作业，上图中，绿色是Reduce Task，云状表示写屏蔽，需要将中间结果持久化写到HDFS。

Tez可以将多个有依赖的作业转换为一个作业，这样只需写一次HDFS，且中间节点较少，从而大大提升作业的计算性能。

**3.2.1 安装包准备**

1）下载tez的依赖包：<http://tez.apache.org>

2）拷贝apache-tez-0.9.1-bin.tar.gz到hadoop102的/opt/software目录

[woaini@hadoop102 software]$ ls

apache-tez-0.9.1-bin.tar.gz

3）将apache-tez-0.9.1-bin.tar.gz上传到HDFS的/tez目录下。

[woaini@hadoop102 conf]$ hadoop fs -mkdir /tez

[woaini@hadoop102 conf]$ hadoop fs -put /opt/software/apache-tez-0.9.1-bin.tar.gz/ /tez

4）解压缩apache-tez-0.9.1-bin.tar.gz

[woaini@hadoop102 software]$ tar -zxvf apache-tez-0.9.1-bin.tar.gz -C /opt/module

5）修改名称

[woaini@hadoop102 module]$ mv apache-tez-0.9.1-bin/ tez-0.9.1

**3.2.2 集成Tez**

1）进入到Hive的配置目录：/opt/module/hive/conf

[woaini@hadoop102 conf]$ pwd

/opt/module/hive/conf

2）在Hive的/opt/module/hive/conf下面创建一个tez-site.xml文件

[woaini@hadoop102 conf]$ pwd

/opt/module/hive/conf

[woaini@hadoop102 conf]$ vim tez-site.xml

添加如下内容

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>tez.lib.uris</name>

<value>${fs.defaultFS}/tez/apache-tez-0.9.1-bin.tar.gz</value>

</property>

<property>

<name>tez.use.cluster.hadoop-libs</name>

<value>true</value>

</property>

<property>

<name>tez.history.logging.service.class</name>

<value>org.apache.tez.dag.history.logging.ats.ATSHistoryLoggingService</value>

</property>

</configuration>

2）在hive-env.sh文件中添加tez环境变量配置和依赖包环境变量配置

[woaini@hadoop102 conf]$ mv hive-env.sh.template hive-env.sh

[woaini@hadoop102 conf]$ vim hive-env.sh

添加如下配置

# Set HADOOP\_HOME to point to a specific hadoop install directory

export HADOOP\_HOME=/opt/module/hadoop-2.7.2

# Hive Configuration Directory can be controlled by:

export HIVE\_CONF\_DIR=/opt/module/hive/conf

# Folder containing extra libraries required for hive compilation/execution can be controlled by:

export TEZ\_HOME=/opt/module/**tez-0.9.1** #是你的tez的解压目录

export TEZ\_JARS=""

for jar in `ls $TEZ\_HOME |grep jar`; do

export TEZ\_JARS=$TEZ\_JARS:$TEZ\_HOME/$jar

done

for jar in `ls $TEZ\_HOME/lib`; do

export TEZ\_JARS=$TEZ\_JARS:$TEZ\_HOME/lib/$jar

done

export HIVE\_AUX\_JARS\_PATH=/opt/module/hadoop-2.7.2/share/hadoop/common/**hadoop-lzo-0.4.20.jar**$TEZ\_JARS

3）在hive-site.xml文件中添加如下配置，更改hive计算引擎

<property>

<name>hive.execution.engine</name>

<value>tez</value>

</property>

**3.2.3 测试**

1）启动Hive

[woaini@hadoop102 hive]$ bin/hive

2）创建表

hive (default)> create table student(

id int,

name string);

3）向表中插入数据

hive (default)> insert into student values(1,"zhangsan");

4）如果没有报错就表示成功了

hive (default)> select \* from student;

1 zhangsan

### 3.2.4 注意事项

1）运行Tez时检查到用过多内存而被NodeManager杀死进程问题：

Caused by: org.apache.tez.dag.api.SessionNotRunning: TezSession has already shutdown. Application application\_1546781144082\_0005 failed 2 times due to AM Container for appattempt\_1546781144082\_0005\_000002 exited with exitCode: -103

For more detailed output, check application tracking page:http://hadoop103:8088/cluster/app/application\_1546781144082\_0005Then, click on links to logs of each attempt.

Diagnostics: Container [pid=11116,containerID=container\_1546781144082\_0005\_02\_000001] is running beyond virtual memory limits. Current usage: 216.3 MB of 1 GB physical memory used; 2.6 GB of 2.1 GB virtual memory used. Killing container.

这种问题是从机上运行的Container试图使用过多的内存，而被NodeManager kill掉了。

[摘录] The NodeManager is killing your container. It sounds like you are trying to use hadoop streaming which is running as a child process of the map-reduce task. The NodeManager monitors the entire process tree of the task and if it eats up more memory than the maximum set in mapreduce.map.memory.mb or mapreduce.reduce.memory.mb respectively, we would expect the Nodemanager to kill the task, otherwise your task is stealing memory belonging to other containers, which you don't want.

**2）解决方法：**

（1）关掉虚拟内存检查，修改yarn-site.xml，

<property>

<name>yarn.nodemanager.vmem-check-enabled</name>

<value>false</value>

</property>

（2）修改后一定要分发，并重新启动hadoop集群。

[woaini@hadoop102 hadoop]$ xsync yarn-site.xml