尚硅谷大数据项目之电商数仓（业务数据采集平台）

(作者：尚硅谷大数据研发部)

版本：V6.2.0

第 1 章 电商业务简介

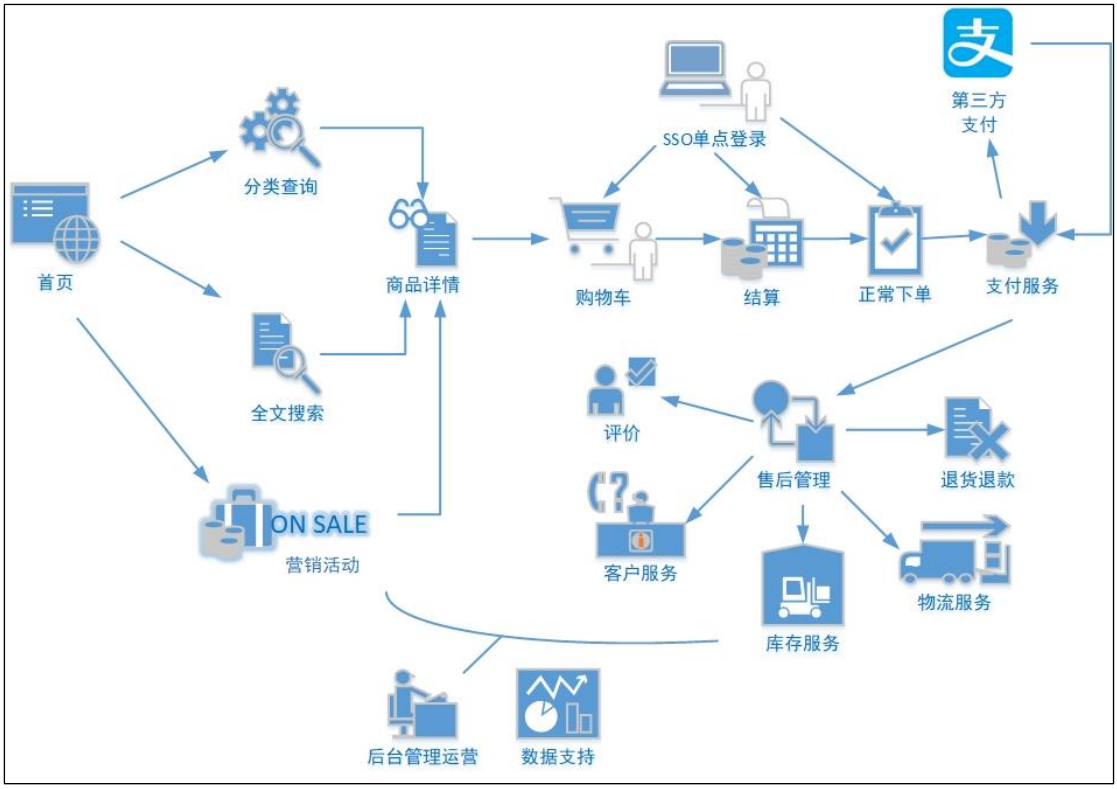
1.1 电商业务流程

电商的业务流程可以以一个普通用户的浏览足迹为例进行说明，用户点开电商首页开始浏览，可能会通过分类查询也可能通过全文搜索寻找自己中意的商品，这些商品无疑都是存储在后台的管理系统中的。

当用户寻找到自己中意的商品，可能会想要购买，将商品添加到购物车后发现需要登录，登录后对商品进行结算，这时候购物车的管理和商品订单信息的生成都会对业务数据库产生影响，会生成相应的订单数据和支付数据。

订单正式生成之后，还会对订单进行跟踪处理，直到订单全部完成。

电商的主要业务流程包括用户前台浏览商品时的商品详情的管理，用户商品加入购物车进行支付时用户个人中心&支付服务的管理，用户支付完成后订单后台服务的管理，这些流程涉及到了十几个甚至几十个业务数据表，甚至更多。



1.2 电商常识（SKU、SPU）

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

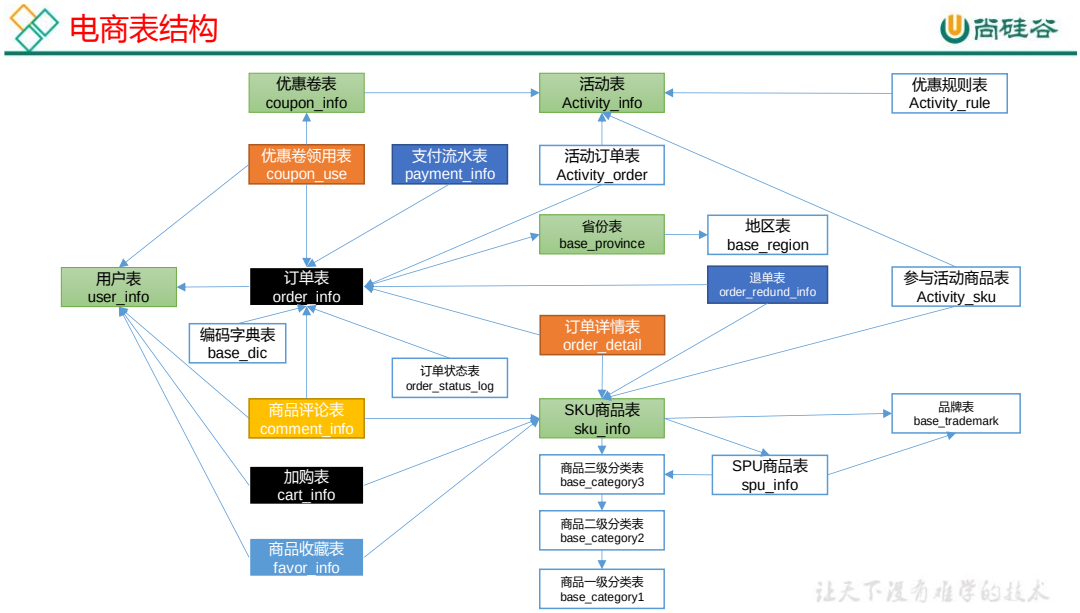
SPU（Standard Product Unit）：是商品信息聚合的最小单位，是一组可复用、易检索的标准化信息集合。

例如：iPhoneX 手机就是 SPU。一台银色、128G 内存的、支持联通网络的 iPhoneX，就是 SKU。

SPU 表示一类商品。好处就是：可以共用商品图片，海报、销售属性等。

1.3 电商业务表结构

本电商数仓系统涉及到的业务数据表结构关系。这 24 个表以订单表、用户表、SKU 商品表、活动表和优惠券表为中心，延伸出了优惠券领用表、支付流水表、活动订单表、订单详情表、订单状态表、商品评论表、编码字典表退单表、SPU 商品表等，用户表提供用户的详细信息，支付流水表提供该订单的支付详情，订单详情表提供订单的商品数量等情况，商品表给订单详情表提供商品的详细信息。本次讲解只以此 24 个表为例，实际项目中，业务数据库中表格远远不止这些。



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	运费金额

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）
sku_num	商品数量
create_time	创建时间
source_type	来源类型
source_id	来源编号

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	真实姓名
phone_num	手机号
email	邮箱
head_img	头像
user_level	用户级别

birthday	生日
gender	性别：男=M，女=F
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 id
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 id
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
user_id	用户 id
order_id	订单编号
sku_id	sku_id
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
source_type	来源类型
source_id	来源编号

1.3.17 商品收藏表（favor_info）

标签	含义
id	编号
user_id	用户 id
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	收藏时间
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	优惠券状态
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	优惠券编号
coupon_name	优惠券名称

coupon_type	优惠券类型 1 现金券 2 折扣券 3 满减券 4 满件打折券
condition_amount	满减金额
condition_num	满减件数
activity_id	活动编号
benefit_amount	优惠金额
benefit_discount	优惠折扣
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）

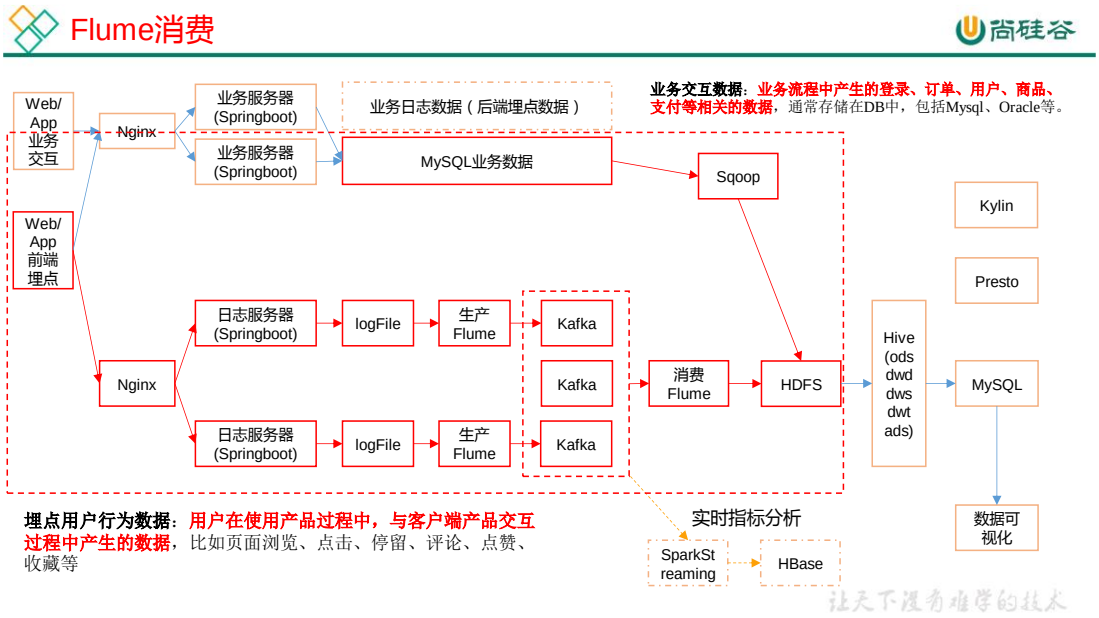
标签	含义

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 参与活动商品表（activity_sku）（暂不导入）

标签	含义
id	编号
activity_id	活动 id
sku_id	sku_id
create_time	创建时间

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）将安装包和 JDBC 驱动上传到/opt/software，共计 6 个

```
01_mysql-community-common-5.7.16-1.el7.x86_64.rpm
02_mysql-community-libs-5.7.16-1.el7.x86_64.rpm
03_mysql-community-libs-compat-5.7.16-1.el7.x86_64.rpm
04_mysql-community-client-5.7.16-1.el7.x86_64.rpm
05_mysql-community-server-5.7.16-1.el7.x86_64.rpm
mysql-connector-java-5.1.27-bin.jar
```

2）如果是虚拟机按照如下步骤执行

（1）卸载自带的 Mysql-libs（如果之前安装过 mysql，要全都卸载掉）

```
[atguigu@hadoop102 software]$ rpm -qa | grep -i -E
```

```
mysql\|mariadb | xargs -n1 sudo rpm -e --nodeps
```

3）如果是阿里云服务器按照如下步骤执行

（1）卸载 MySQL 依赖，虽然机器上没有装 MySQL，但是这一步不可少

```
[atguigu@hadoop102 software]# sudo yum remove mysql-libs
```

（2）下载依赖并安装

```
[atguigu@hadoop102 software]# sudo yum install libaio
[atguigu@hadoop102 software]# sudo yum -y install autoconf
```

2.1.2 安装 MySQL

1）安装 mysql 依赖

```
[atguigu@hadoop102 software]$ sudo rpm -ivh 01_mysql-community-common-5.7.16-1.el7.x86_64.rpm
[atguigu@hadoop102 software]$ sudo rpm -ivh 02_mysql-community-libs-5.7.16-1.el7.x86_64.rpm
[atguigu@hadoop102 software]$ sudo rpm -ivh 03_mysql-community-libs-compat-5.7.16-1.el7.x86_64.rpm
```

2）安装 mysql-client

```
[atguigu@hadoop102 software]$ sudo rpm -ivh 04_mysql-community-client-5.7.16-1.el7.x86_64.rpm
```

3）安装 mysql-server

```
[atguigu@hadoop102 software]$ sudo rpm -ivh 05_mysql-community-server-5.7.16-1.el7.x86_64.rpm
```

注意：如果报如下错误，这是由于 yum 安装了旧版本的 GPG keys 所造成，从 rpm 版本 4.1 后，在安装或升级软件包时会自动检查软件包的签名。

```
warning:     05_mysql-community-server-5.7.16-1.el7.x86_64.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY
error: Failed dependencies:
libaio.so.1()(64bit)   is   needed   by   mysql-community-server-5.7.16-1.el7.x86_64
```

解决办法

```
[atguigu@hadoop102 software]$ sudo rpm -ivh 05_mysql-community-server-5.7.16-1.el7.x86_64.rpm --force --nodeps
```

4）启动 mysql

```
[atguigu@hadoop102 software]$ sudo systemctl start mysqld
```

5）查看 mysql 密码

```
[atguigu@hadoop102 software]$ sudo cat /var/log/mysqld.log | grep password
```

2.1.3 配置 MySQL

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）用刚刚查到的密码进入 mysql（如果报错，给密码加单引号）

```
[atguigu@hadoop102 software]$ mysql -uroot -p'password'
```

2）设置复杂密码（由于 mysql 密码策略，此密码必须足够复杂）

```
mysql> set password=password("Qs23=zs32");
```

3）更改 mysql 密码策略

```
mysql> set global validate_password_length=4;
mysql> set global validate_password_policy=0;
```

4）设置简单好记的密码

```
mysql> set password=password("000000");
```

5）进入 msyql 库

```
mysql> use mysql
```

6）查询 user 表

```
mysql> select user, host from user;
```

7）修改 user 表，把 Host 表内容修改为%

```
mysql> update user set host="%" where user="root";
```

8）刷新

```
mysql> flush privileges;
```

9）退出

```
mysql> quit;
```

2.2 业务数据生成

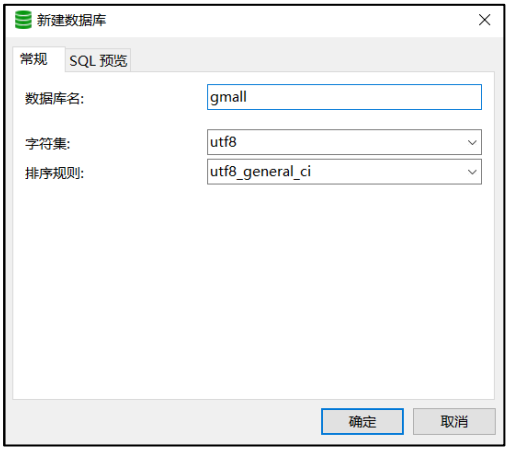
2.2.1 连接 MySQL

通过 MySQL 操作可视化工具 SQLyog 连接 MySQL。

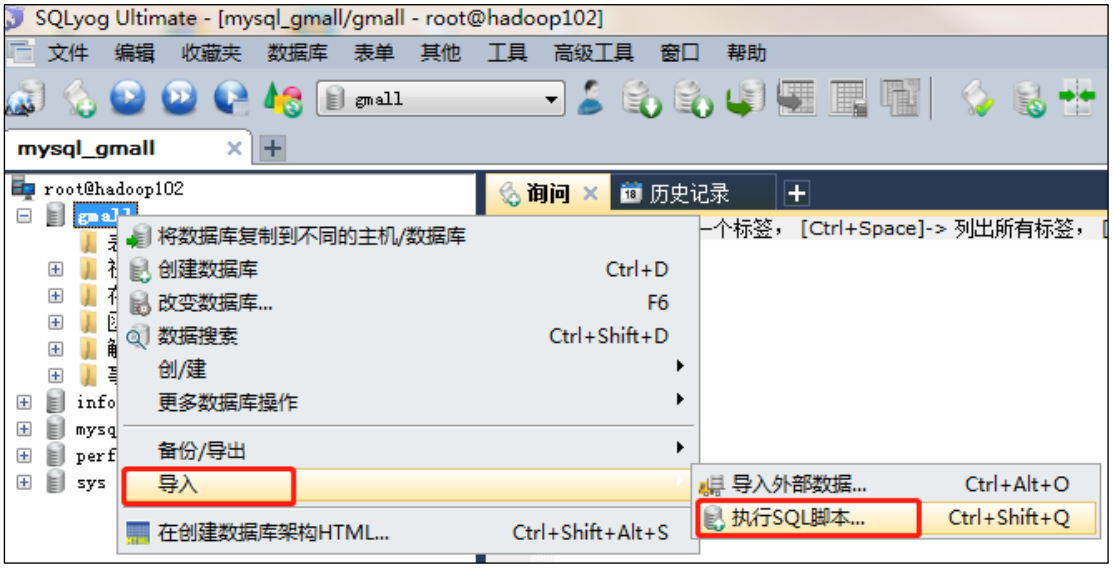
2.2.2 建表语句

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（**gmall2020-04-01.sql**）



注意：完成后，要记得右键，刷新一下对象浏览器，就可以看见数据库中的表了。

2.2.3 生成业务数据

1）在 hadoop102 的/opt/module/目录下创建 db_log 文件夹

```
[atguigu@hadoop102 module]$ mkdir db_log/
```

2）把 gmall2020-mock-db-2020-04-01.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=2020-06-14
#是否重置
mock.clear=1

#生成新用户数量
mock.user.count=1000
#男性比例
mock.user.male-rate=20
#用户数据变化概率
mock.user.update-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=30
#每个商品最多购物个数
mock.cart.sku-maxcount-per-cart=3
#购物车来源  用户查询，商品推广，智能推荐，促销活动
mock.cart.source-type-rate=60:20:10:10

#用户下单比例
mock.order.user-rate=95
#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=1000

#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 2020-06-14 日期数据：

```
[atguigu@hadoop102 db_log]$ java -jar gmall2020-mock-db-2020-04-01.jar
```

5）在配置文件 application.properties 中修改

mock.date=2020-06-15

mock.clear=0

6）再次执行命令，生成 2020-06-15 日期数据：

```
[atguigu@hadoop102 db_log]$ java -jar gmall2020-mock-db-2020-04-01.jar
```

2.2.4 业务数据建模

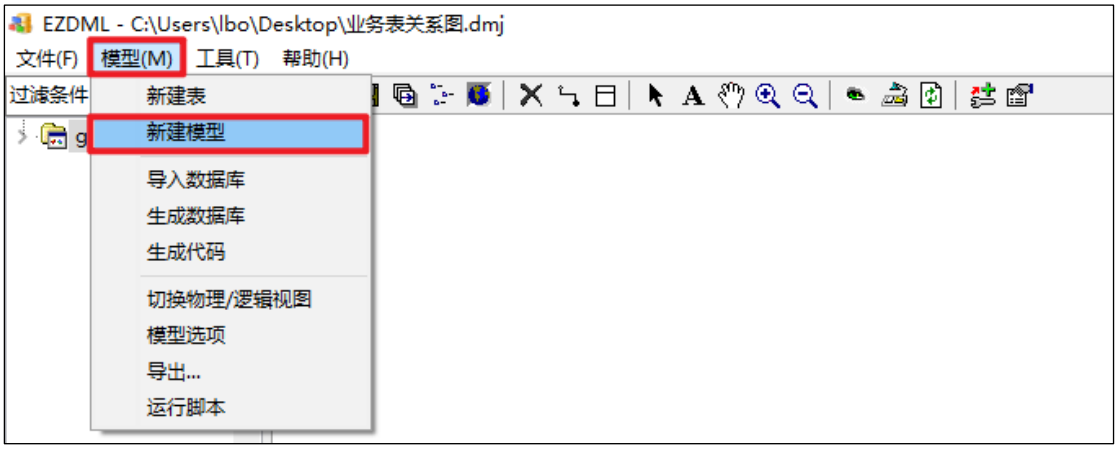
可借助 EZDML 这款数据库设计工具，来辅助我们梳理复杂的业务表关系。

1）下载地址

http://www.ezdml.com/download_cn.html

2）使用说明

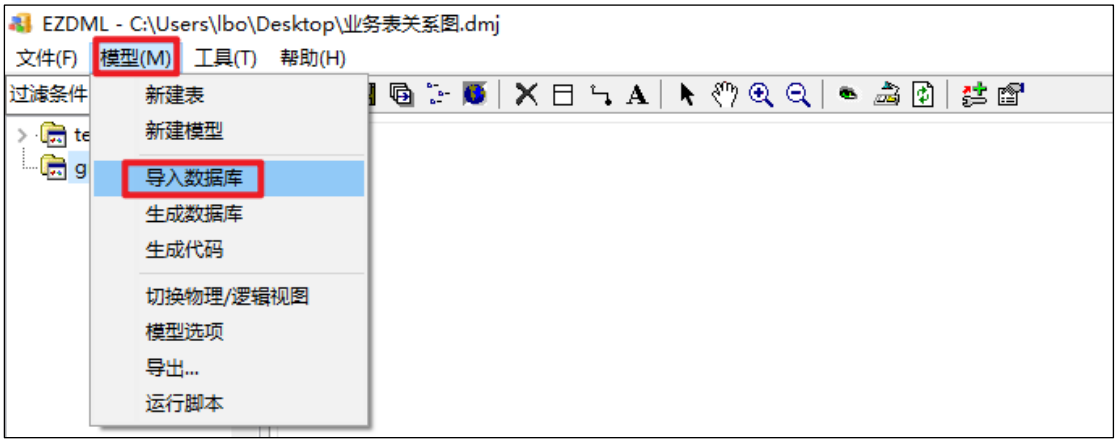
（1）新建模型



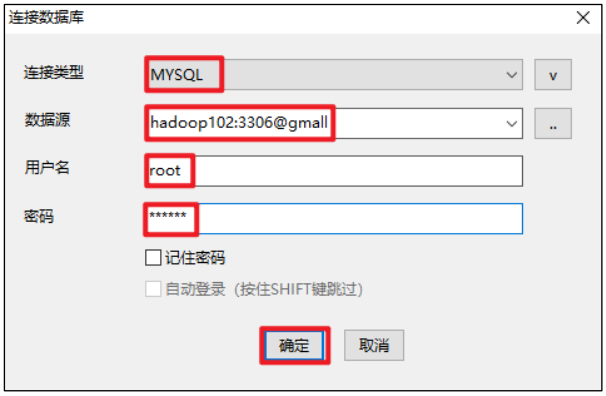
（2）命名模型

（3）点击图标，选中模型

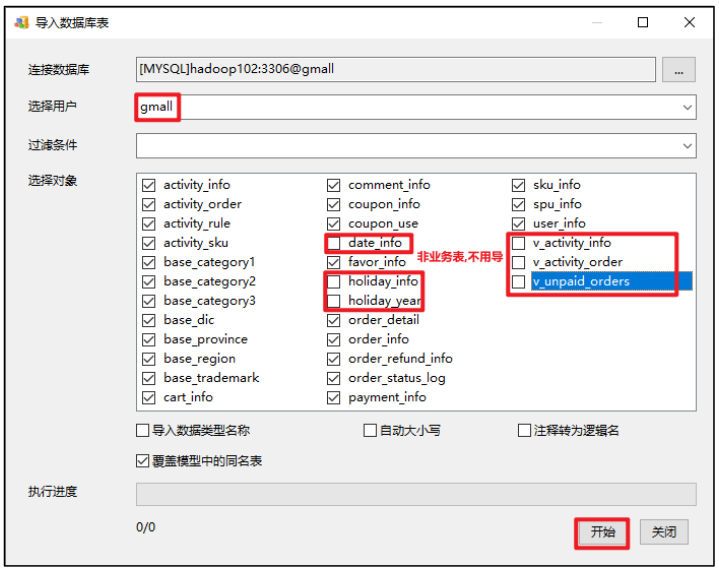
（4）导入数据库



（5）配置数据库连接

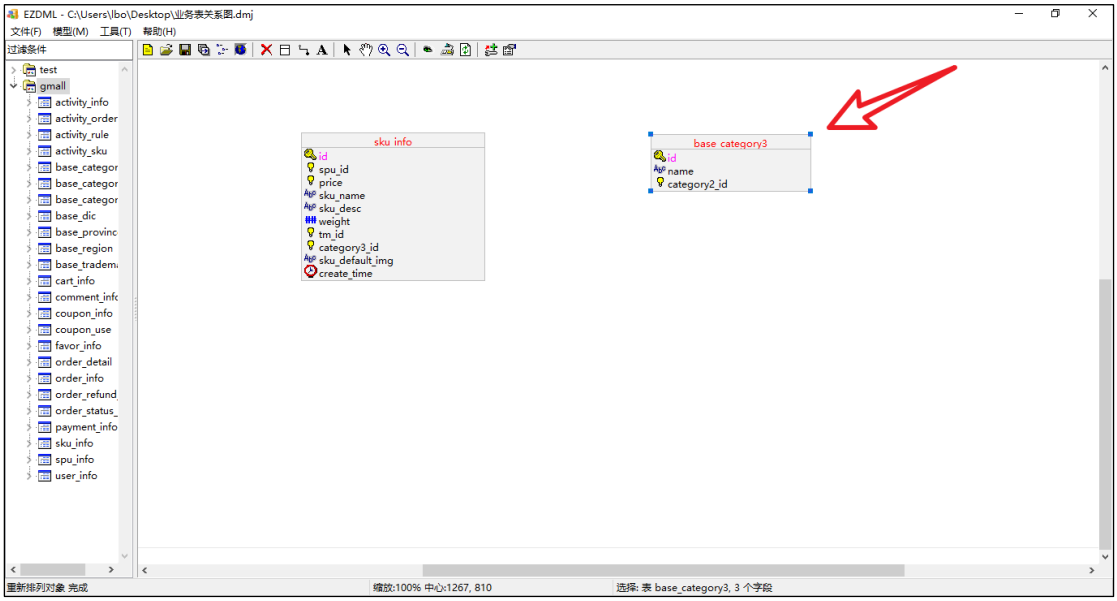


（6）选择导入的表

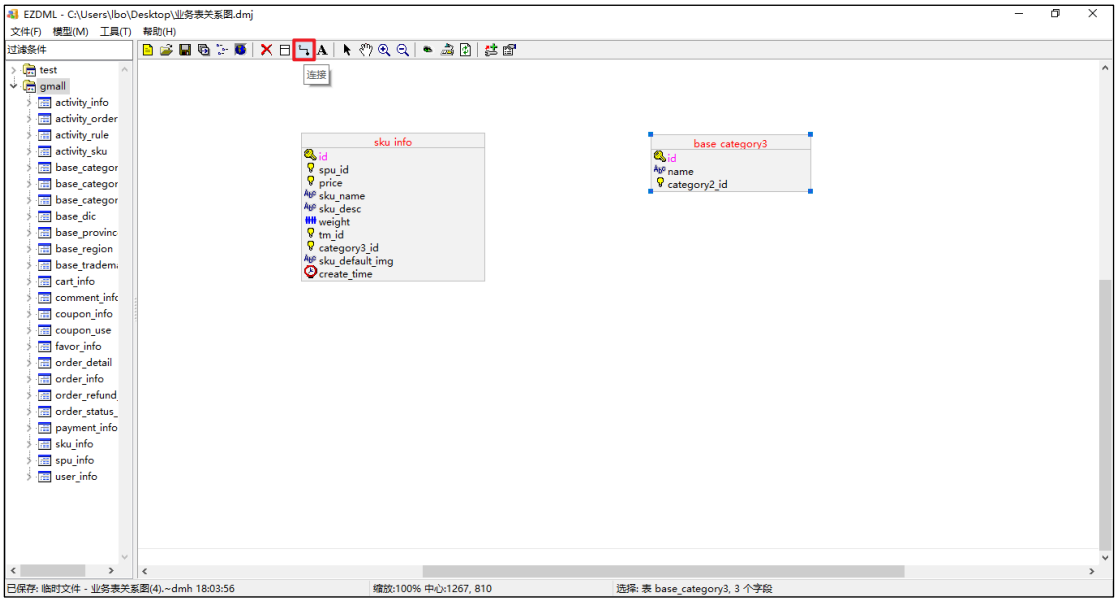


（7）建立表关系

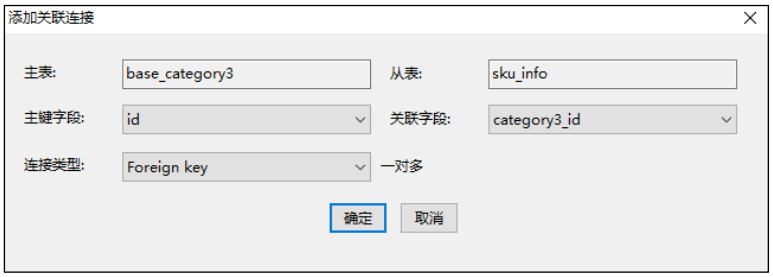
第一步：点击选中主表（主键所在的表）



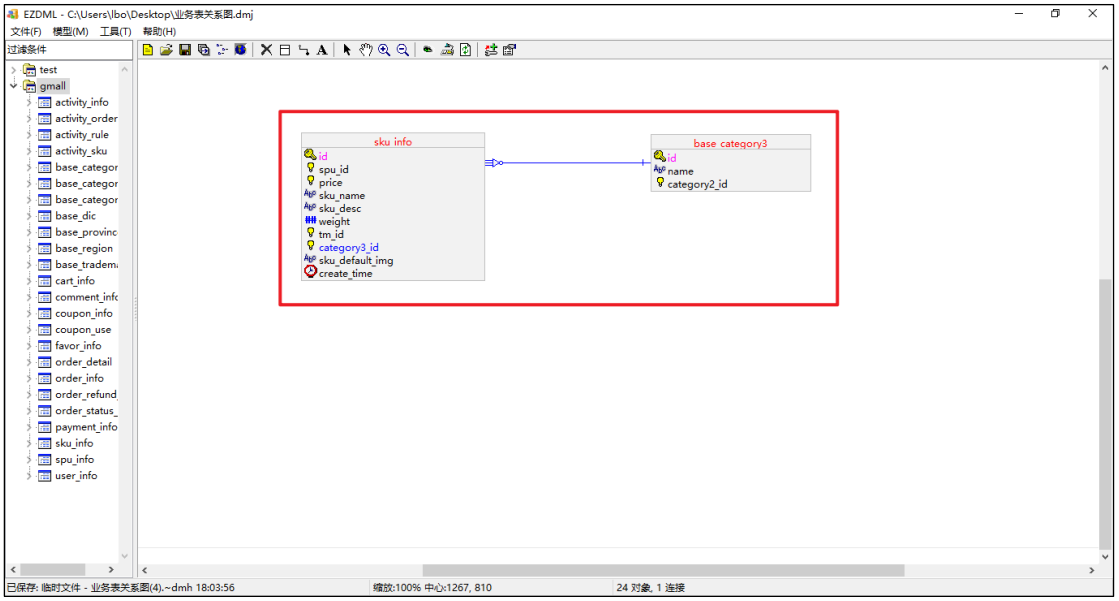
第二步：点击连接按钮



第三步：点击从表，配置连接条件

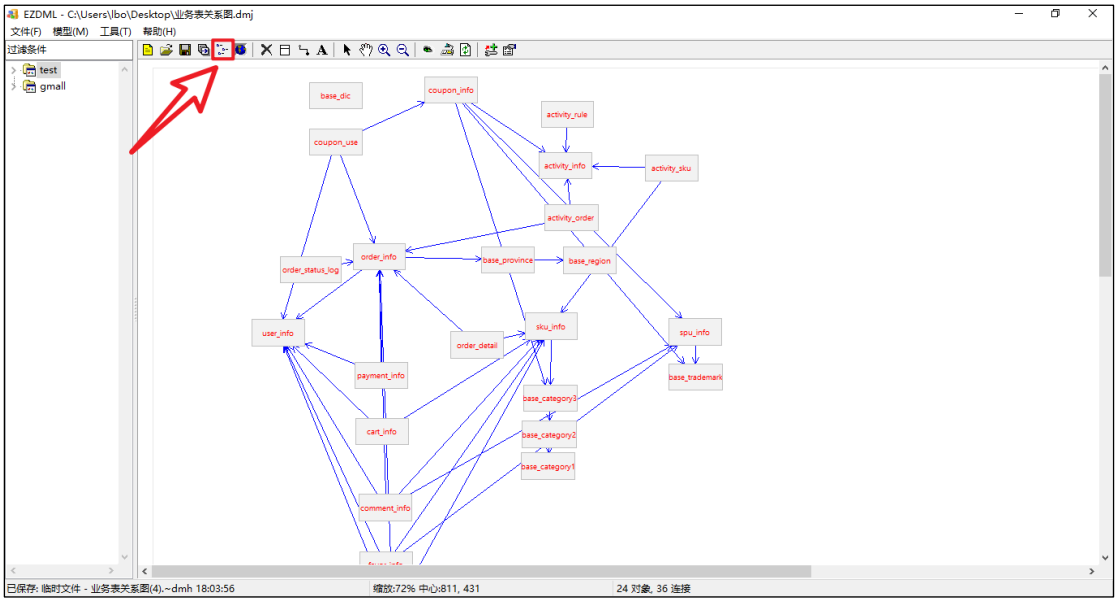


第四步：效果展示



3）使用技巧

（1）缩略图



（2）热键

按住 shift 键，用鼠标点击表，进行多选，可实现批量移动

按住 ctrl 键，用鼠标圈选表，也可进行多选，实现批量移动

2.3 Sqoop 安装

2.3.1 下载并解压

1）sqoop 官网地址：http://sqoop.apache.org/docs/1.4.7/index.html

2）下载地址：http://mirrors.hust.edu.cn/apache/sqoop/1.4.7/

3）上传安装包 sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz 到 hadoop102 的/opt/software 路径中

4）解压 sqoop 安装包到指定目录，如：

```
[atguigu@hadoop102 software]$ tar -zxvf sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz -C /opt/module/
```

5）修改名称为 sqoop：

```
[atguigu@hadoop102 module]$ mv sqoop-1.4.7.bin__hadoop-2.6.0/ sqoop
```

2.3.2 修改配置文件

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

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

2）修改配置文件

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

增加如下内容

```
export HADOOP_COMMON_HOME=/opt/module/hadoop-3.1.3
export HADOOP_MAPRED_HOME=/opt/module/hadoop-3.1.3
export HIVE_HOME=/opt/module/hive
```

```
export ZOOKEEPER_HOME=/opt/module/zookeeper-3.5.7
export ZOOCFGDIR=/opt/module/zookeeper-3.5.7/conf
```

2.3.3 拷贝 JDBC 驱动

1）将 mysql-connector-java-5.1.48.jar 上传到/opt/software 路径

2）进入到/opt/software/路径，拷贝 jdbc 驱动到 sqoop 的 lib 目录下。

```
[atguigu@hadoop102 software]$ cp /opt/software/mysql-connector-java-5.1.48.jar /opt/module/sqoop/lib/
```

2.3.4 验证 Sqoop

我们可以通过某一个 command 来验证 sqoop 配置是否正确：

```
[atguigu@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
```

2.3.5 测试 Sqoop 是否能够成功连接数据库

```
[atguigu@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
```

2.3.6 Sqoop 基本使用

将 mysql 中 user_info 表数据导入到 hdfs 的/test 路径

```
bin/sqoop import \
--connect jdbc:mysql://hadoop102:3306/gmall \
--username root \
--password 000000 \
--table user_info \
--columns id,login_name \
```

```
--where "id>=10 and id<=30" \
--target-dir /test \
--delete-target-dir \
--fields-terminated-by '\t' \
--num-mappers 2 \
--split-by id
```

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、特殊表

- 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。

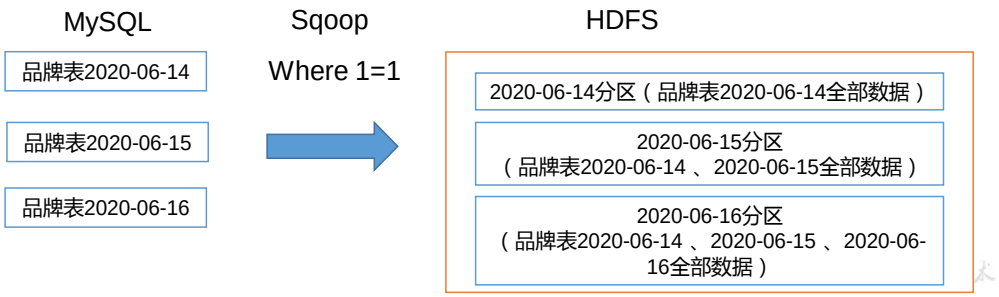
2.4.1 全量同步策略

全量同步策略

每日全量，就是每天存储一份**完整数据**，作为一个分区。

适用于表**数据量不大**，且每天既会有**新数据插入**，也会有**旧数据的修改**的场景。

例如：编码字典表、品牌表、商品三级分类、商品二级分类、商品一级分类、优惠规则表、活动表、活动参与商品表、加购表、商品收藏表、优惠卷表、SKU商品表、SPU商品表

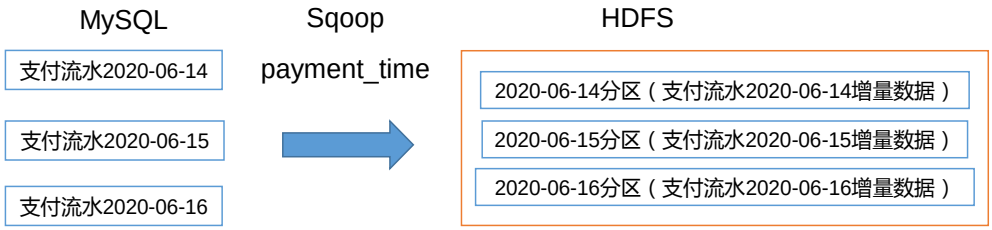


2.4.2 增量同步策略

增量同步策略

每日增量，就是每天存储一份增量数据，作为一个分区。

适用于表数据量大，且每天只会有新数据插入的场景。例如：退单表、订单状态表、支付流水表、订单详情表、活动与订单关联表、商品评论表。



2.4.3 新增及变化策略

新增及变化同步策略

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。

trade_body	create_time	operate_time
小米（MI）电视 55英寸曲面4K智能WiFi网络液晶电视机4S L55M5-AQ 小米电	2020-06-14 08:37:45	2020-06-15 08:39:34
北纯精制黄小米（小黄米 月子米 小米粥 粗粮杂粮 大米伴侣）2.18kg等1件	2020-06-14 08:37:45	2020-06-15 08:39:34
Apple iPhoneXSMax (A2104) 256GB 深空灰色 移动联通电信4G手机 双卡双待	2020-06-14 08:37:45	2020-06-15 08:39:34
小米（MI） 小米路由器4 双千兆路由器 无线家用穿墙1200M高速双频wifi 千	2020-06-14 08:37:45	2020-06-14 08:37:45
迪奥（Dior）烈艳蓝金唇膏/口红 珊瑚粉 ACTRICE 028号 3.5g等2件商品	2020-06-14 08:37:45	2020-06-14 08:37:45
北纯精制黄小米（小黄米 月子米 小米粥 粗粮杂粮 大米伴侣）2.18kg等3件	2020-06-14 08:37:45	2020-06-14 08:37:45
Apple iPhoneXSMax (A2104) 256GB 深空灰色 移动联通电信4G手机 双卡双待	2020-06-14 08:37:45	2020-06-14 08:37:45
迪奥（Dior）烈艳蓝金唇膏/口红 珊瑚粉 ACTRICE 028号 3.5g等2件商品	2020-06-14 08:37:45	2020-06-14 08:37:45
荣耀10青春版 幻彩渐变 2400万AI自拍 全网通版4GB+64GB 渐变蓝 移动联通	2020-06-14 08:37:45	2020-06-14 08:37:45
小米（MI） 小米路由器4 双千兆路由器 无线家用穿墙1200M高速双频wifi 千	2020-06-15 08:39:33	2020-06-15 08:39:34
迪奥（Dior）烈艳蓝金唇膏/口红 珊瑚粉 ACTRICE 028号 3.5g等4件商品	2020-06-15 08:39:33	2020-06-15 08:39:34

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

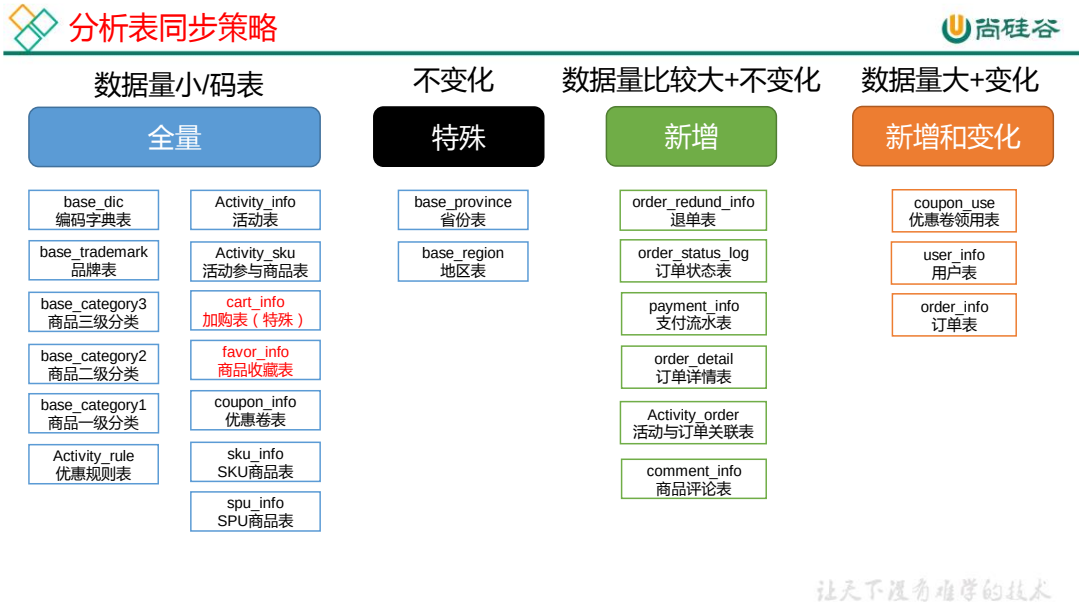
省份表、地区表

2.5 业务数据导入 HDFS

2.5.1 分析表同步策略

在生产环境，个别小公司，为了简单处理，所有表全量导入。

中大型公司，由于数据量比较大，还是严格按照同步策略导入数据。



2.5.2 脚本编写

1）在/home/atguigu/bin 目录下创建

```
[atguigu@hadoop102 bin]$ vim mysql_to_hdfs.sh
```

添加如下内容：

```
#! /bin/bash

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

if [ -n "$2" ] ;then
    do_date=$2
else
    do_date=`date -d '-1 day' +%F`
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-3.1.3/share/hadoop/common/hadoop-lzo-0.4.20.jar
com.hadoop.compression.lzo.DistributedLzoIndexer
/origin_data/gmall/db/$1/$do_date
}

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,
                          source_type,
                          source_id
                        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,
                      source_type,
                      source_id
                    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，[atguigu@hadoop102 ~]$ date --help

2）修改脚本权限

```
[atguigu@hadoop102 bin]$ chmod 777 mysql_to_hdfs.sh
```

3）初次导入

```
[atguigu@hadoop102 bin]$ mysql_to_hdfs.sh first 2020-06-14
```

4）每日导入

```
[atguigu@hadoop102 bin]$ mysql_to_hdfs.sh all 2020-06-15
```

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 Hive 安装部署

1）把 apache-hive-3.1.2-bin.tar.gz 上传到 linux 的/opt/software 目录下

2）解压 apache-hive-3.1.2-bin.tar.gz 到/opt/module/目录下面

```
[atguigu@hadoop102 software]$ tar -zxvf /opt/software/apache-hive-3.1.2-bin.tar.gz -C /opt/module/
```

3）修改 apache-hive-3.1.2-bin.tar.gz 的名称为 hive

```
[atguigu@hadoop102 software]$ mv /opt/module/apache-hive-3.1.2-bin/ /opt/module/hive
```

4）修改/etc/profile.d/my_env.sh，添加环境变量

```
[atguigu@hadoop102 software]$ sudo vim /etc/profile.d/my_env.sh
```

添加内容

```
#HIVE_HOME
export HIVE_HOME=/opt/module/hive
export PATH=$PATH:$HIVE_HOME/bin
```

重启 Xshell 对话框或者 source 一下 /etc/profile.d/my_env.sh 文件，使环境变量生效

```
[atguigu@hadoop102 software]$ source /etc/profile.d/my_env.sh
```

5）解决日志 Jar 包冲突，进入/opt/module/hive/lib 目录

```
[atguigu@hadoop102 lib]$ mv log4j-slf4j-impl-2.10.0.jar log4j-slf4j-impl-2.10.0.jar.bak
```

3.2 Hive 元数据配置到 MySQL

3.2.1 拷贝驱动

将 MySQL 的 JDBC 驱动拷贝到 Hive 的 lib 目录下

```
[atguigu@hadoop102 lib]$ cp /opt/software/mysql-connector-java-5.1.48.jar /opt/module/hive/lib/
```

3.2.2 配置 Metastore 到 MySQL

在$HIVE_HOME/conf 目录下新建 hive-site.xml 文件

```
[atguigu@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?useSSL=false</value>
    </property>

    <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.mysql.jdbc.Driver</value>
    </property>

    <property>
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>root</value>
    </property>

    <property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>000000</value>
    </property>

    <property>
        <name>hive.metastore.warehouse.dir</name>
        <value>/user/hive/warehouse</value>
    </property>

    <property>
        <name>hive.metastore.schema.verification</name>
        <value>false</value>
    </property>

    <property>
    <name>hive.server2.thrift.port</name>
    <value>10000</value>
    </property>

    <property>
```

```
        <name>hive.server2.thrift.bind.host</name>
        <value>hadoop102</value>
    </property>

    <property>

<name>hive.metastore.event.db.notification.api.auth</name>
        <value>false</value>
    </property>

    <property>
        <name>hive.cli.print.header</name>
        <value>true</value>
    </property>

    <property>
        <name>hive.cli.print.current.db</name>
        <value>true</value>
    </property>
</configuration>
```

3.3 启动 Hive

3.3.1 初始化元数据库

1）登陆 MySQL

```
[atguigu@hadoop102 conf]$ mysql -uroot -p000000
```

2）新建 Hive 元数据库

```
mysql> create database metastore;
mysql> quit;
```

3）初始化 Hive 元数据库

```
[atguigu@hadoop102 conf]$ schematool -initSchema -dbType mysql
-verbose
```

3.3.2 启动 hive 客户端

1）启动 Hive 客户端

```
[atguigu@hadoop102 hive]$ bin/hive
```

2）查看一下数据库

```
hive (default)> show databases;
OK
database_name
default
```