目录

**[一 数仓基础 1](#_Toc25664)**

[1.1 数仓的定义 1](#_Toc8407)

[1.2 数据仓库和数据库关系 4](#_Toc29852)

[1.3 数据仓库架构 6](#_Toc17248)

[1.4 OLAP VS. OLTP 7](#_Toc1228)

**[二 数据仓库案例 9](#_Toc21390)**

[2.1 环境准备 9](#_Toc6571)

[2.2 数仓分层 14](#_Toc11736)

[2.3 数仓开发规范 16](#_Toc1203)

[2.4 数仓模型 17](#_Toc12589)

[2.5 建立物理模型 18](#_Toc5898)

[2.6 建库、装载数据 18](#_Toc26695)

[2.7 数据仓库初始化 21](#_Toc24068)

[2.8 ETL过程 23](#_Toc10911)

[2.9 ETL自动化 42](#_Toc15560)

**[三 数据集市 43](#_Toc23053)**

[3.1 数据集市概念 43](#_Toc5107)

[3.2 集市分类 44](#_Toc19028)

[3.3 集市和主题的区别 45](#_Toc28522)

**[四 项目需求 46](#_Toc2605)**

[4.1 项目需求分析 46](#_Toc13251)

[4.2 项目描述 47](#_Toc16785)

[4.3 业务系统说明 47](#_Toc20458)

[4.4 业务需求指标 48](#_Toc14150)

**[五 项目建设实战 51](#_Toc23785)**

[5.1 技术选型 51](#_Toc24586)

[5.2 建立物理模型 51](#_Toc9471)

[5.3 ETL过程 60](#_Toc11206)

[5.4 项目ETL完整代码 66](#_Toc5230)

[5.5 开发数据仓库的分析应用 89](#_Toc10674)

**[六 知识点扩展 184](#_Toc27039)**

[6.1 概念模型 184](#_Toc8794)

[6.2 逻辑模型 190](#_Toc26517)

[6.3 事实表 201](#_Toc32466)

[6.4 维度表 202](#_Toc9410)

[6.5 数据粒度 204](#_Toc20137)

# 一 数仓基础

## 1.1 数仓的定义

### 1.1.1 数仓的发展历程

1. 萌芽阶段。数据仓库概念最早可追溯到20世纪70年代，MIT的研究员致力于研究一种优化的技术架构，该架构试图将业务处理系统和分析系统分开，即将业务处理和分析处理分为不同层次，针对各自的特点采取不同的架构设计原则，MIT的研究员认为这两种信息处理的方式具有显著差别，以至于必须采取完全不同的架构和设计方法。但受限于当时的信息处理能力，这个研究仅仅停留在理论层面。
2. 探索阶段。20世纪80年代中后期，DEC公司结合MIT的研究结论，建立了TA2（Technical Architecture2）规范，该规范定义了分析系统的四个组成部分：数据获取、数据访问、目录和用户服务。这是系统架构的一次重大转变，第一次明确提出分析系统架构并将其运用于实践。
3. 雏形阶段。1988年，为解决全企业集成问题，IBM公司第一次提出了信息仓库（InformationWarehouse）的概念，并称之为VITAL规范（VirtuallyIntegrated Technical Architecture Lifecycle）。VITAL定义了85种信息仓库组件，包括PC、图形化界面、面向对象的组件以及局域网等。至此，数据仓库的基本原理、技术架构以及分析系统的主要原则都已确定，数据仓库初具雏形。
4. 确立阶段。1991年Bill Inmon（比尔·恩门）出版了他的第一本关于数据仓库的书《Building the Data Warehouse》，标志着数据仓库概念的确立。该书指出，数据仓库(DataWarehouse)是一个面向主题的(Subject Oriented)、集成的(Integrated)、相对稳定的(Non-Volatile)、反映历史变化的(Time Variant)数据集合，用于支持管理决策(Decision-Making Support)。该书还提供了建立数据仓库的指导意见和基本原则。凭借着这本书，Bill Inmon被称为数据仓库之父。

### 1.1.2 数仓的定义

数据仓库即Data Warehouse，简称DW，主要研究和解决从数据中获取信息的问题，为企业所有级别的决策制定过程，提供所有类型数据支持的战略集合。本质上，数据仓库试图提供一种从操作型系统到决策支持系统的数据流架构模型。主要是解决多重数据复制带来的高成本问题。在有数仓之前，需要大量的冗余数据来支撑多个决策支持系统，尽管每个系统服务于不同的用户，但是这些系统经常需要大量相同的数据。决策支持系统Decision support System，即DDS，是用于支持业务或组织决策活动的信息系统，服务于组织管理、运营和规划管理层（通常是中层或高级管理层），帮助人们对可能快速变化并且不容易预测结果的问题做出决策。数据仓库就是为决策系统提供数据支持的。

William H.Inmon（比尔·恩门）：数据仓库是一个面向主题的、集成的、非易失的且随时间变化的数据集合，用于支持管理人员的决策。

### 1.1.3 数仓的四大特征

1. 数据仓库的数据是面向主题的
2. 数据仓库的数据是集成的
3. 数据仓库的数据是非易失的
4. 数据仓库的数据是随时间不断变化的

#### 1.1.3.1 面向主题

主题（Subject）：特定的数据分析领域与目标。

主题是根据分析的要求来确定的。这与按照数据处理或应用的要求来组织数据是不同的。如在生产企业中，同样是材料供应，在操作型数据库系统中，人们所关心的是怎样更方便和更快捷地进行材料供应的业务处理；而在进行分析处理时，人们就应该关心材料的不同采购渠道和材料供应是否及时，以及材料质量状况等。

数据仓库是面向分析、决策人员的主观要求的，不同的用户有不同的要求，同一个用户的要求也会随时间而经常变化，因此，数据仓库中的主题有时会因用户主观要求的变化而变化的。

**面向主题划分如下：**

数据仓库面向在数据模型中已经定义好的公司的主要主题领域。典型的主题领域包括顾客、产品、订单和财务或是其他某项事务或活动。  
  
基本主题：  
教育机构：学生、讲师、班主任、课程等  
电商行业：运营、流量、价值、商品、市场、风控、销售等

#### 1.1.3.2 集成

* 集成性是指数据仓库中数据必须是一致的。数据仓库的数据是从原有的分散的多个数据库、数据文件和数据段中抽取来的，数据来源可能既有内部数据又有外部数据。
* 数据仓库中的数据是为分析服务的，而分析需要多种广泛的不同数据源以便进行比较、鉴别，因此数据仓库中的数据必须从多个数据源中获取，这些数据源包括多种类型数据库、文件系统以及Internet网上数据等，它们通过数据集成而形成数据仓库中的数据。
* **集成的方法：**
  + 统一：消除不一致的现象
  + 综合：对原有数据进行综合和计算
* **集成需要考虑的问题：**
  + 数据格式
  + 计量单位
  + 数据代码含义混乱
  + 数据名称混乱

#### 1.1.3.3 非易失

* 数据仓库中的数据是经过抽取而形成的分析型数据，不具有原始性，主要供企业决策分析之用，执行的主要是‘查询’操作，一般情况下不执行‘更新’操作。同时，一个稳定的数据环境也有利于数据分析操作和决策的制订。
* 面向应用的事务数据库需要对数据进行频繁的插入、更新操作，而对于数据仓库中数据的操作仅限于数据的初始导入和记录查询。

#### 1.1.3.4 随时间不断变化

* 数据仓库以维的形式对数据进行组织，时间维是数据仓库中很重要的一个维度。并且数据仓库中的数据时间跨度大，从几年甚至到几十年，称为历史数据。
* 数据仓库中的数据必须以一定时间段为单位进行统一更新。
* 数据变化方式:
* –不断增加新的数据内容  
  –不断删去旧的数据内容  
  –更新与时间有关的综合数据

数据的生命周期与行业、自己本身的需求有关，比如金融业“在设计银行数据保存周期策略时，最常用的经验法则是7年和13个月规则”  
 基础数据区里面通过历史表（拉链表）来保存重要信息的历史数据，一般客户类、账户类等信息要保留7年，交易类流水类信息要保留至少13个月以上。除此之外，重要代码、主数据也要通过历史表保存历史。

根据业务决定数据的生命周期，比如电商交易频繁的可能是2年，保险行业交易比较少的5年。太老的数据对于数据分析没多大作用，你想10年前的电商交易数据对于现在的电商能有多大帮助，价格、产品、用户都已经完全不同了

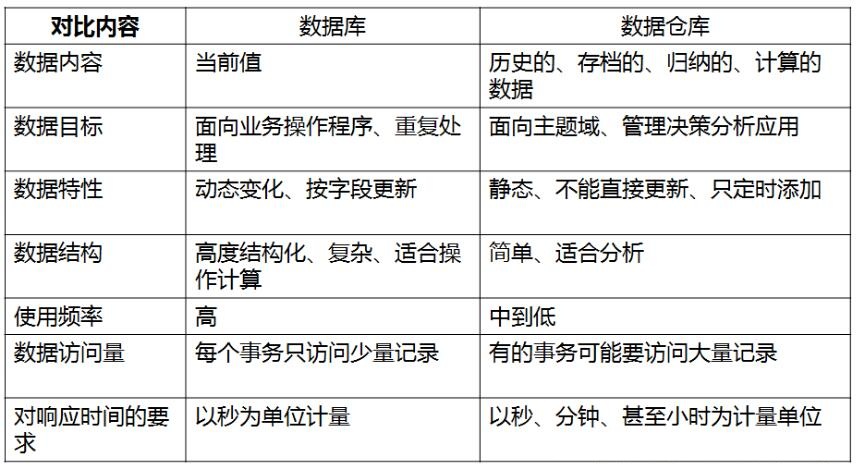
如果数据仓库是否仅用于分析的话（我看好多地方建立的数据仓库仅用于统计分析，对于数据挖掘基本都没有用），如果有大量的数据挖掘的话，那么数据多些对于结果越精确。（当然，前提是你的历史数据质量不太差的情况下）

现在存储设备越来越便宜，如果不是数据量很惊人的话，一般是不用删除或导出的，因为导出后是需要管理的。

## 1.2 数据仓库和数据库关系

1. 数仓主要用于解决企业级的数据分析问题。
2. 数据库是为捕获和存储数据而设计，数据仓库是为分析数据而设计。
3. 数据库是面向事务设计的，属于操作型。数据仓库是面向分析，面向主题设计的，即信息是按主题进行组织的，属于分析型。
4. 数据库设计是尽量避免冗余，一般采用符合范式的规则来设计，数据仓库在设计是有意引入冗余，采用反范式的方式来设计。
5. 数据库是数据仓库的基础，数据库较小，而数据仓库较大，通常一个数据仓库中的数据来源于多个数据库的异构。
6. 数据不一样。

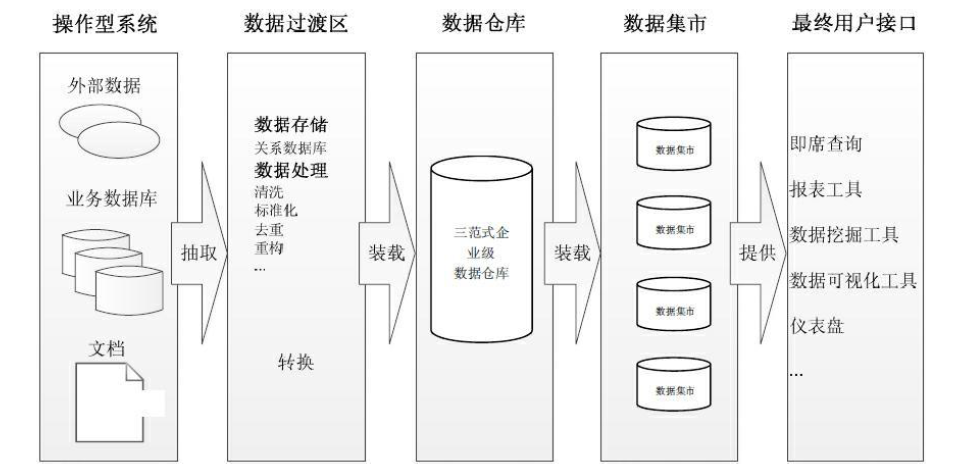
对比如下：



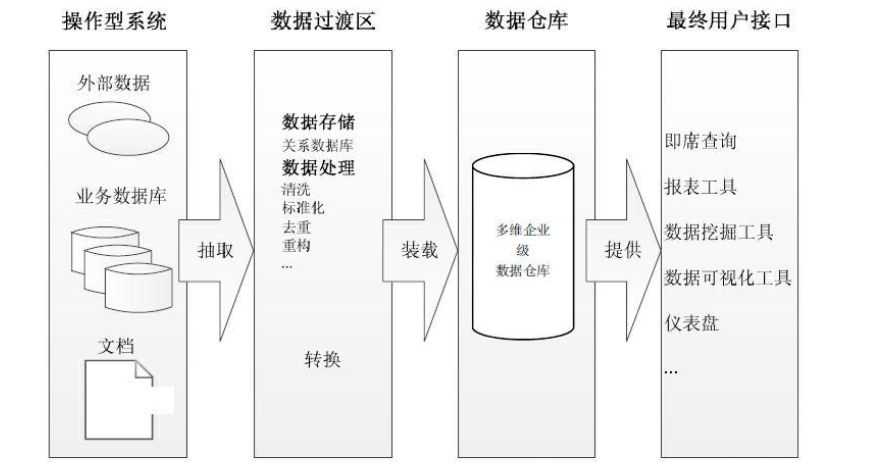


## 1.3 数据仓库架构

inmon架构



kimball架构



inmon架构和kimball架构的区别就是inmon的数据仓库是三范式企业级数据仓库，kimball的数据库时多维企业级数据仓库。

## 1.4 OLAP VS. OLTP



联机事务处理OLTP（on-line transaction processing） 主要是执行基本日常的事务处理，比如数据库记录的增删查改。比如在银行的一笔交易记录，就是一个典型的事务。

联机分析处理OLAP（On-Line Analytical Processing） 是数据仓库系统的主要应用，支持复杂的分析操作，侧重决策支持，并且提供直观易懂的查询结果。典型的应用就是复杂的动态的报表系统。

OLTP的特点一般有：

1.实时性要求高。我记得之前上大学的时候，银行异地汇款，要隔天才能到账，而现在是分分钟到账的节奏，说明现在银行的实时处理能力大大增强。   
  
2.数据量不是很大，生产库上的数据量一般不会太大，而且会及时做相应的数据处理与转移。   
  
3.交易一般是确定的，比如银行存取款的金额肯定是确定的，所以OLTP是对确定性的数据进行存取   
  
4.高并发，并且要求满足ACID原则。比如两人同时操作一个银行卡账户，比如大型的购物网站秒杀活动时上万的QPS请求。

OLAP的特点一般有：

1.实时性要求不是很高，比如最常见的应用就是天级更新数据，然后出对应的数据报表。   
  
2.数据量大，因为OLAP支持的是动态查询，所以用户也许要通过将很多数据的统计后才能得到想要知道的信息，例如时间序列分析等等，所以处理的数据量很大;   
  
3.OLAP系统的重点是通过数据提供决策支持，所以查询一般都是动态，自定义的。所以在OLAP中，维度的概念特别重要。一般会将用户所有关心的维度数据，存入对应数据平台。

总结：

OLTP即联机事务处理，就是我们经常说的关系数据库，增删查改就是我们经常应用的东西，这是数据库的基础；TPCC(Transaction Processing Performance Council)属于此类。

OLAP即联机分析处理，是数据仓库的核心部心，所谓数据仓库是对于大量已经由OLTP形成的数据的一种分析型的数据库，用于处理商业智能、决策支持等重要的决策信息；数据仓库是在数据库应用到一定程序之后而对历史数据的加工与分析，读取较多，更新较少，TPCH属于此类。

随着大数据时代的到来，对于OLAP，列存储模式或者说nosql模式比传统意义的行存储模式可能更具优势。

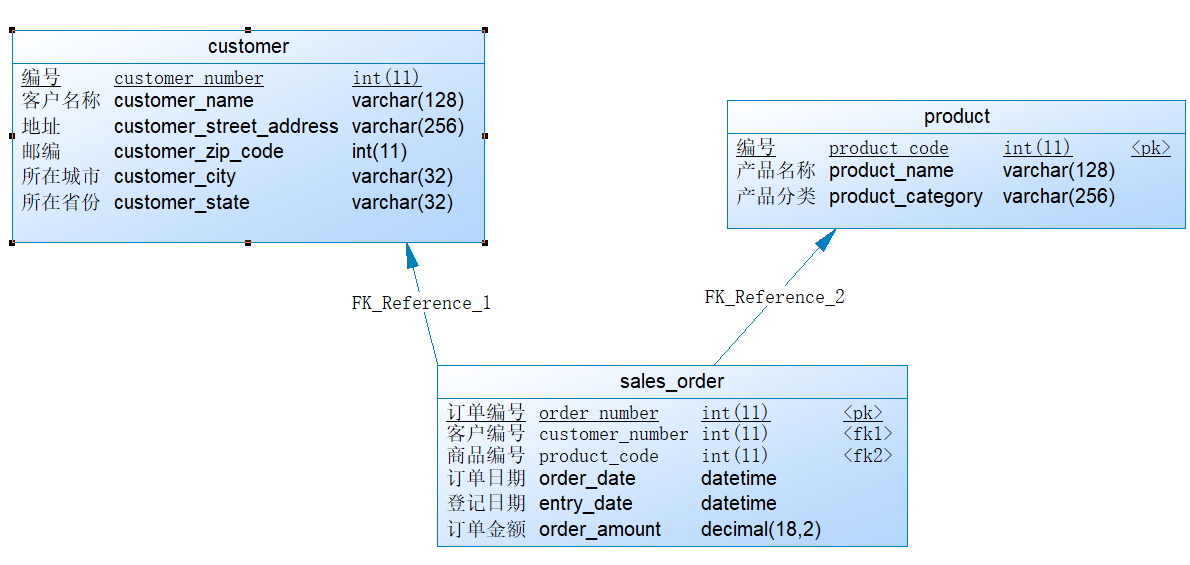
联机事务处理【OLTP Online Transaction Processing】  
 联机事务处理，表示事务性非常高的系统，一般都是高可用的在线系统，以小的事务以及小的查询为主，以传统的关系型数据库为主要应用，主要是基本的、日常的事务处理，主要为业务数据，例如银行交易  
 OLTP系统最容易出现瓶颈的地方就是CPU与磁盘子系统。 OLTP比较常用的设计与优化方式为Cache技术与B-tree索引技术.OLTP 系统是一个数据块变化非常频繁，SQL 语句提交非常频繁的系统.

联机分析处理【OLAP Online Analytical Processing】：  
 联机分析处理，有的时候也叫DSS决策支持系统，就是我们说的数据仓库，重点主要是面向分析，支持复杂的分析操作，侧重决策支持，并且提供直观易懂的查询结果。典型的应用就是复杂的动态的报表系统。会产生大量的查询，一般很少涉及增删改。在这样的系统中，语句的执行量不是考核标准，因为一条语句的执行时间可能会非常长，读取的数据也非常多。所以，在这样的系统中，考核的标准往往是磁盘子系统的吞吐量（带宽），如能达到多少MB/s的流量。  
 在OLAP系统中，常使用分区技术、并行技术。 分区技术在OLAP系统中的重要性主要体现在数据库管理上，分区技术对性能上的影响，它可以使得一些大表的扫描变得很快（只扫描单个分区）。另外，如果分区结合并行的话，也可以使得整个表的扫描会变得很快。总之，分区主要的功能是管理上的方便性，它并不能绝对保证查询性能的提高，有时候分区会带来性能上的提高，有时候会降低

# 二 数据仓库案例

## 2.1 环境准备

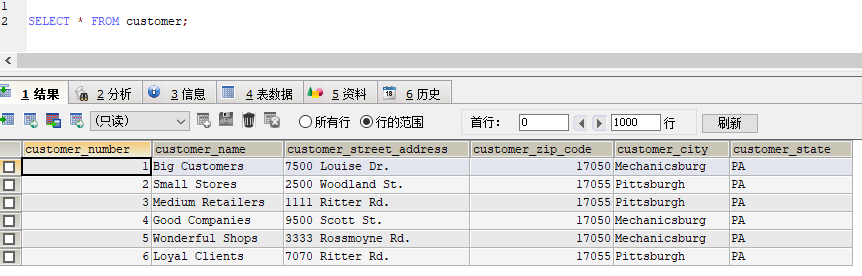
源系统是mysql库，数据模型如下

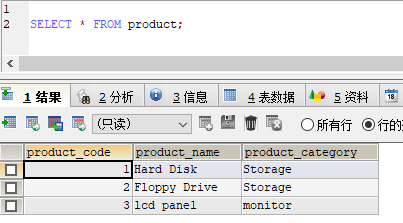


建表语句如下：

/\*==============================================================\*/  
/\* DBMS name: MySQL 5.0 \*/  
/\* Created on: 2018/11/23 1:09:10 \*/  
/\*==============================================================\*/  
  
CREATE DATABASE IF NOT EXISTS sales\_source DEFAULT CHARSET utf8 COLLATE utf8\_general\_ci;   
  
USE sales\_source;  
  
DROP TABLE IF EXISTS customer;  
  
DROP TABLE IF EXISTS product;  
  
DROP TABLE IF EXISTS sales\_order;  
  
/\*==============================================================\*/  
/\* Table: customer \*/  
/\*==============================================================\*/  
CREATE TABLE customer  
(  
 customer\_number INT(11) NOT NULL AUTO\_INCREMENT,  
 customer\_name VARCHAR(128) NOT NULL,  
 customer\_street\_address VARCHAR(256) NOT NULL,  
 customer\_zip\_code INT(11) NOT NULL,  
 customer\_city VARCHAR(32) NOT NULL,  
 customer\_state VARCHAR(32) NOT NULL,  
 PRIMARY KEY (customer\_number)  
);  
  
/\*==============================================================\*/  
/\* Table: product \*/  
/\*==============================================================\*/  
CREATE TABLE product  
(  
 product\_code INT(11) NOT NULL AUTO\_INCREMENT,  
 product\_name VARCHAR(128) NOT NULL,  
 product\_category VARCHAR(256) NOT NULL,  
 PRIMARY KEY (product\_code)  
);  
  
/\*==============================================================\*/  
/\* Table: sales\_order \*/  
/\*==============================================================\*/  
CREATE TABLE sales\_order  
(  
 order\_number INT(11) NOT NULL AUTO\_INCREMENT,  
 customer\_number INT(11) NOT NULL,  
 product\_code INT(11) NOT NULL,  
 order\_date DATETIME NOT NULL,  
 entry\_date DATETIME NOT NULL,  
 order\_amount DECIMAL(18,2) NOT NULL,  
 PRIMARY KEY (order\_number)  
);  
  
/\*==============================================================\*/  
/\* insert data \*/  
/\*==============================================================\*/  
  
INSERT INTO customer  
( customer\_name  
, customer\_street\_address  
, customer\_zip\_code  
, customer\_city  
, customer\_state  
 )  
VALUES  
 ('Big Customers', '7500 Louise Dr.', '17050',  
 'Mechanicsburg', 'PA')  
, ( 'Small Stores', '2500 Woodland St.', '17055',  
 'Pittsburgh', 'PA')  
, ('Medium Retailers', '1111 Ritter Rd.', '17055',  
 'Pittsburgh', 'PA'  
)  
, ('Good Companies', '9500 Scott St.', '17050',  
 'Mechanicsburg', 'PA')  
, ('Wonderful Shops', '3333 Rossmoyne Rd.', '17050',  
 'Mechanicsburg', 'PA')  
, ('Loyal Clients', '7070 Ritter Rd.', '17055',  
 'Pittsburgh', 'PA')  
;   
   
INSERT INTO product(product\_name,product\_category) VALUES  
('Hard Disk','Storage'),  
('Floppy Drive','Storage'),  
('lcd panel','monitor')  
;  
  
DROP PROCEDURE IF EXISTS usp\_generate\_order\_data;  
DELIMITER //  
CREATE PROCEDURE usp\_generate\_order\_data()  
BEGIN  
  
 DROP TABLE IF EXISTS tmp\_sales\_order;  
 CREATE TABLE tmp\_sales\_order AS SELECT \* FROM sales\_order WHERE 1=0;  
 SET @start\_date := UNIX\_TIMESTAMP('2020-1-1');  
 SET @end\_date := UNIX\_TIMESTAMP('2020-1-31');  
 SET @i := 1;  
 WHILE @i<=10000 DO  
 SET @customer\_number := FLOOR(1+RAND()\*6);  
 SET @product\_code := FLOOR(1+RAND()\* 3);  
 SET @order\_date := FROM\_UNIXTIME(@start\_date+RAND()\*(@end\_date-@start\_date));  
 SET @amount := FLOOR(1000+RAND()\*9000);  
 INSERT INTO tmp\_sales\_order VALUES (@i,@customer\_number,@product\_code,@order\_date,@order\_date,@amount);  
 SET @i := @i +1;  
 END WHILE;  
 TRUNCATE TABLE sales\_order;  
 INSERT INTO sales\_order  
 SELECT NULL,customer\_number,product\_code,order\_date,entry\_date,order\_amount  
 FROM tmp\_sales\_order;  
 COMMIT;  
 DROP TABLE tmp\_sales\_order;  
END //  
  
  
CALL usp\_generate\_order\_data();

建完库后的表和数据如下







## 2.2 数仓分层

### 2.2.1 分层概念

数据仓库更多代表的是一种对数据的管理和使用的方式，它是一整套包括了etl、调度、建模在内的完整的理论体系流程。数据仓库在构建过程中通常都需要进行分层处理。业务不同，分层的技术处理手段也不同。

分层的主要原因是在管理数据的时候，能对数据有一个更加清晰的掌控。详细来讲，主要有下面几个原因：

* 清晰数据结构
* 每一个数据分层都有它的作用域，这样我们在使用表的时候能更方便地定位和理解。
* 数据血缘追踪
* 简单来说，我们最终给业务呈现的是一个能直接使用业务表，但是它的来源有很多，如果有一张来源表出问题了，我们希望能够快速准确地定位到问题，并清楚它的危害范围。
* 减少重复开发
* 规范数据分层，开发一些通用的中间层数据，能够减少极大的重复计算。
* 把复杂问题简单化
* 将一个复杂的任务分解成多个步骤来完成，每一层只处理单一的步骤，比较简单和容易理解。而且便于维护数据的准确性，当数据出现问题之后，可以不用修复所有的数据，只需要从有问题的步骤开始修复。
* 屏蔽原始数据的异常
* 屏蔽业务的影响，不必改一次业务就需要重新接入数据

### 2.2.2 常见分层

数仓的常见分层一般为3层，分别为：数据操作层、数据仓库层和数据集市层。当然根据研发人员经验或者业务，可以分为更多不同的层，只要能达到流程清晰、方便查数即可。

ODS:

Operate data store，操作数据存储，是最接近数据源中数据的一层，数据源中的数据，经过抽取、洗净、传输，也就说传说中的ETL之后，装入本层。本层的数据，总体上大多是按照源头业务系统的分类方式而分类的。

例如这一层可能包含的数据表可为：人口表（包含每个人的身份证号、姓名、性别、年龄、住址等）、机场登机记录（包含乘机人身份证号、航班号、乘机日期、起飞城市等）、银联的刷卡信息表（包含银行卡号、刷卡地点、刷卡时间、刷卡金额等）、银行账户表（包含银行卡号、持卡人身份证号等）等等一系列原始的业务数据。这里我们可以看到，这一层面的数据还具有鲜明的业务数据库的特征，甚至还具有一定的关系数据库中的数据范式的组织形式。

但是，这一层面的数据却不完全等同于原始数据。在源数据装入这一层时，根据业务不同，可能会进行诸如去噪（例如去掉明显偏离正常水平的银行刷卡信息）、去重（例如银行账户信息、公安局人口信息中均含有人的姓名，但是只保留一份即可）、提脏（例如有的人的银行卡被盗刷，在十分钟内同时有两笔分别在中国和日本的刷卡信息，这便是脏数据）、业务提取、单位统一、砍字段（例如用于支撑前端系统工作，但是在数据挖掘中不需要的字段）、业务判别等多项工作。

ODS层数据的来源方式：

* 业务库
* 经常会使用sqoop来抽取，比如我们每天定时抽取一次。在实时方面，可以考虑用canal监听mysql的binlog，实时接入即可。
* 埋点日志
* 线上系统会打入各种日志，这些日志一般以文件的形式保存，我们可以选择用flume定时抽取，也可以用用spark streaming或者storm来实时接入，当然，kafka也会是一个关键的角色。
* 其它数据源
* 不同的业务其它数据源不一样,比如第三方数据。

DW:

Data warehouse，数据仓库层。在这里，从ODS层中获得的数据按照主题建立各种数据模型。

例如以研究人的旅游消费为主题的数据集中,便可以结合航空公司的登机出行信息，以及银联系统的刷卡记录，进行结合分析，产生数据集。在这里，我们需要了解四个概念：维（dimension）、事实（Fact）、指标（Index）和粒度（ Granularity）。

DM:

该层主要是提供数据产品和数据分析使用的数据，一般会存放在es、mysql等系统中供线上系统使用，也可能会存在Hive或者Druid中供数据分析和数据挖掘使用。 比如我们经常说的报表数据，或者说那种大宽表，一般就放在这里。

注意:

每个分层不是必须要用ODS、DW、DM等字样来标识，可以随便起名字，只要统一这一层是什么类型数据，名字符合知名知意即可。

### 2.2.3 维度层DIM

针对存在度量值的表创建维度表，来从各个角度描述事实表。而一个项目中可能有很多张维度表，那我们可以选择将这些维度表放到单独的一个库中，形成维度库，即可看成是维度层，意在让数仓数据清晰明了。

## 2.3 数仓开发规范

1 数据库命名  
 数据命名规则：数仓层\_业务方式 如 ods\_release/sda\_release  
  
2 数仓各层对应数据库  
 ods/sda层 -> sda/ods\_业务（原始数据）  
 dw层 -> dw\_业务 （主题库）  
 dim层 -> dim\_维度 （维表库）  
 dm层 -> dm\_业务（集市库）  
 middle层 -> mid\_业务（中间库）  
 临时数据 -> temp（临时库）  
  
3 表命名  
 (3-1) 数据库表命名规则：  
 原始层表：数仓层\_来源类型\_业务 如 ods\_01\_xxx  
 其他表：数仓层\_业务 如 dw\_xxx  
   
 如果业务名称较长可以简写 如 ods\_01\_xxx\_xx\_xx ods\_01\_xx  
  
 (3-2) 数据来源代码(sda层)  
 01 -> hdfs数据  
 02 -> mysql数据  
 03 -> redis数据  
 04 -> mongodb数据  
 05 -> tidb数据  
  
如  
 ods\_release.ods\_01\_release 投放数据  
 ods\_release.ods\_02\_user 注册用户表(业务表：存于MYSQL)  
  
 dw\_release.dw\_customer 目标客户主题表  
 dm\_release.dm\_customer\_stat 目标客户统计表

## 2.4 数仓模型

星型模式是维度模型最简单的形式，也是比较常用的模型，我们的案例采用星型模型。所谓星型模型就是以一个事实表为中心，周围围绕多个维度表。

星型模型将业务分为事实和维度。

事实是业务数据的度量值，比如销售额、销售数量等，它记录了特定事件的量化指标，一般是度量值和指向维表的外键组成。事实表的粒度级别通常会设计的比较低。

事实表有三种类型:

事务事实表:最低粒度级别的事实表,记录原始的操作型事件. Dwd

快照事实表:记录给定时间点的事实,如月底账户余额

累积事实表:记录给定时间点的聚合事实,如当月的销售金额.

维度是对事实数据属性的描述，如日期，省份，地区等,维度表的数据量通常不大。

常用的维度表有:

日期维度表,每个数据仓库都需要一个日期维度表。

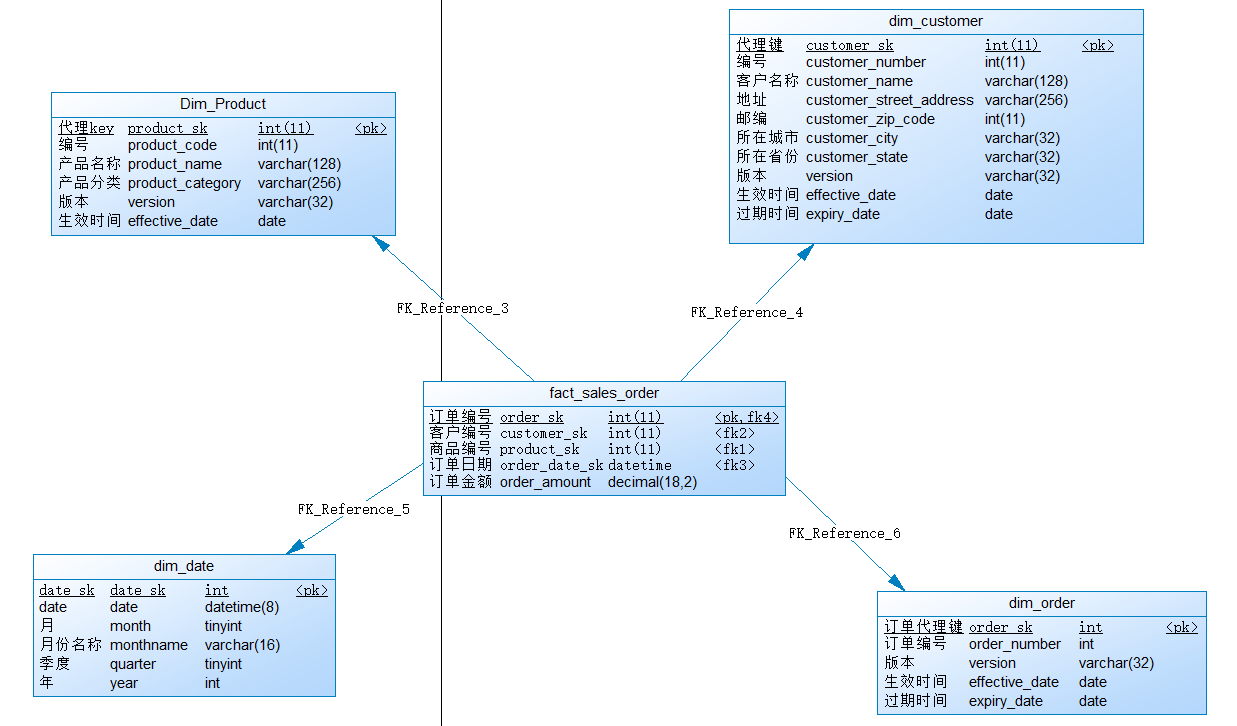
地理维度表:描述位置信息的数据,如国家,省份,城市,区县,邮编等

产品维度表:描述产品及其属性

人员维度表:描述人员相关信息,部门员工表等

范围维度表:描述分段数据的信息等,比如信用等级

## 2.5 建立物理模型



## 2.6 建库、装载数据

连接hive

创建rds库

创建表，脚本如下

create database sales\_rds;  
/\*==============================================================\*/  
/\* Table: customer \*/  
/\*==============================================================\*/  
CREATE TABLE sales\_rds.customer  
(  
 customer\_number INT ,  
 customer\_name VARCHAR(128) ,  
 customer\_street\_address VARCHAR(256) ,  
 customer\_zip\_code INT ,  
 customer\_city VARCHAR(32) ,  
 customer\_state VARCHAR(32)   
);  
  
/\*==============================================================\*/  
/\* Table: product \*/  
/\*==============================================================\*/  
CREATE TABLE sales\_rds.product  
(  
 product\_code INT,  
 product\_name VARCHAR(128) ,  
 product\_category VARCHAR(256)   
);  
  
/\*==============================================================\*/  
/\* Table: sales\_order \*/  
/\*==============================================================\*/  
CREATE TABLE sales\_rds.sales\_order  
(  
 order\_number INT ,  
 customer\_number INT,  
 product\_code INT ,  
 order\_date timestamp ,  
 entry\_date timestamp ,  
 order\_amount DECIMAL(18,2)   
)  
row format delimited fields terminated by '\t'  
  
create database if not exists dw;  
use dw;  
  
create table dim\_Product  
(  
 product\_sk int ,  
 product\_code int ,  
 product\_name varchar(128),  
 product\_category varchar(256),  
 version varchar(32),  
 effective\_date date,  
 expiry\_date date  
)  
clustered by (product\_sk ) into 8 buckets  
stored as orc tblproperties('transactional'='true');  
  
  
/\*==============================================================\*/  
/\* Table: dim\_customer \*/  
/\*==============================================================\*/  
create table dim\_customer  
(  
 customer\_sk int ,  
 customer\_number int ,  
 customer\_name varchar(128),  
 customer\_street\_address varchar(256),  
 customer\_zip\_code int,  
 customer\_city varchar(32),  
 customer\_state varchar(32),  
 version varchar(32),  
 effective\_date date,  
 expiry\_date date  
)  
clustered by (customer\_sk ) into 8 buckets  
stored as orc tblproperties('transactional'='true');  
  
/\*==============================================================\*/  
/\* Table: dim\_date \*/  
/\*==============================================================\*/  
create table dw.dim\_date  
(  
 date\_sk int ,  
 date date,  
 month tinyint,  
 month\_name varchar(16),  
 quarter tinyint,  
 year int  
) row format delimited fields terminated by ','  
stored as textfile;  
  
/\*==============================================================\*/  
/\* Table: dim\_order \*/  
/\*==============================================================\*/  
create table dim\_order  
(  
 order\_sk int ,  
 order\_number int,  
 version varchar(32),  
 effective\_date date,  
 expiry\_date date  
)  
clustered by (order\_sk ) into 8 buckets  
stored as orc tblproperties('transactional'='true');  
;  
  
/\*==============================================================\*/  
/\* Table: fact\_sales\_order \*/  
/\*==============================================================\*/  
create table fact\_sales\_order  
(  
 order\_sk int ,  
 customer\_sk int ,  
 product\_sk int ,  
 order\_date\_sk int ,  
 order\_amount decimal(18,2)  
)  
partitioned by (order\_date string)  
clustered by (order\_sk) into 8 buckets  
row format delimited fields terminated by '\t'   
stored as orc tblproperties('transactional'='true');

注意,dw库的表除dim\_date存储格式是textfile外,其他的表均为orc,并且设置表属性支持事务

虽然表在模型设计中设置了主键,但是在hive中没有主键,外键唯一性约束,非空约束这些概念

## 2.7 数据仓库初始化

### 2.7.1 装载日期维度表

日期维度是数据仓库中一个很重要的角色,一般来说每个数据仓库都要有日期维度,数据仓库存储的是历史数据。分析中一个很重要的指标是基于时间的一个业务的变化趋势,因此,数据仓库中都包含这个维度。

* 从源数据库装载日期

DROP TABLE IF EXISTS `dim\_date`;  
CREATE TABLE `dim\_date` (  
 `date\_sk` int(11) NOT NULL AUTO\_INCREMENT,  
 `date` varchar(20) DEFAULT NULL,  
 `month` int(11) DEFAULT NULL,  
 `month\_name` varchar(50) DEFAULT NULL,  
 `quarter` int(11) DEFAULT NULL,  
 `year` int(11) DEFAULT NULL,  
 PRIMARY KEY (`date\_sk`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
DELIMITER //  
CREATE PROCEDURE USP\_Load\_Dim\_Date(dt\_start DATE,dt\_end DATE)  
BEGIN  
WHILE dt\_start<=dt\_end DO  
 INSERT INTO dim\_date (`date`,`month`,`month\_name`,`quarter`,`year`)  
 VALUES (dt\_start,MONTH(dt\_start),MONTHNAME(dt\_start),QUARTER(dt\_start),YEAR(dt\_start));  
 SET dt\_start =ADDDATE(dt\_start,1);  
END WHILE;  
COMMIT;  
END //  
  
CALL USP\_Load\_Dim\_Date('2010-1-1','2050-1-1');

* 直接为hive数仓生成日期表

我们需要写一个shell脚本,生成日期数据文件,上传到hdfs对应的日期维度表的目录即可

#!/bin/bash  
date1="$1"  
date2="$2"  
tempdate=`date -d "$date1" +%F`  
tempdateSec=`date -d "$date2" +%s`  
enddateSec=`date -d "$date2" +$s`  
min=1  
#max=`expr \( $enddateSec - $tempdateSec \) / \( 24 \\* 60 \\* 60 \) + 1`  
max=14611  
cat /datas >./dim\_date.csv  
  
while [ $min -le $max ]   
do  
 month=`date -d "$tempdate" +%m`  
 month\_name=`date -d "$tempdate" +%B`  
 quarter=`echo $month | awk '{print int(($0-1)/3 +1 }'`  
 year=`date -d "$tempdate" +%Y`  
 echo ${min}","${tempdate}","${month}","${month\_name}","${quarter}","${year} >> ./dim\_date.csv  
 tempdate=`date -d "+$min day $date1" +%F`  
 tempdateSec=`date -d "+min day $date1" +%s`  
 min=`expr $min + 1`  
done

脚本保存为generate*dim*date.sh,执行脚本

sh ./generate\_dim\_date.sh '2020-1-1' '2050-1-1'

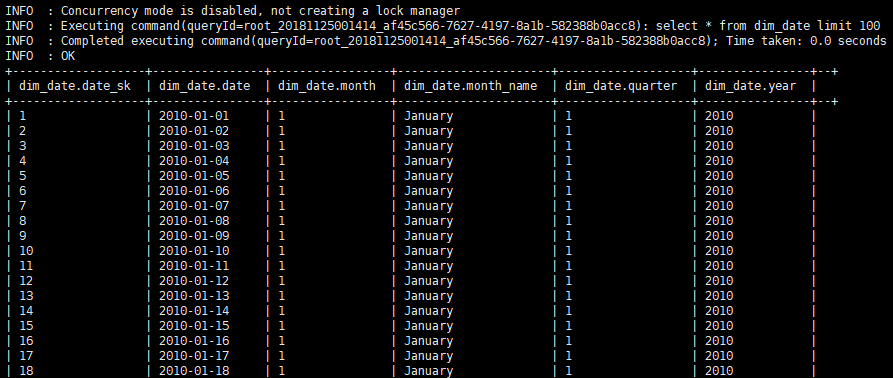
然后再hive中创建日期维度表

CREATE TABLE IF NOT EXISTS dw.dim\_date(  
 date\_sk int,  
 date string,  
 month int,  
 month\_name string,  
 quarter int,  
 year int  
)  
row format delimited fields terminated by ','  
stored as textfile  
;

将生成的文件上传到hdfs下面/hive/warehose/dw.db/dim\_date/

hdfs dfs -put -f dim\_date.csv /user/hive/warehouse/dw.db/dim\_date  
  
## 或者在hive客户端执行  
load data local inpath 'dim\_date.csv' into table dw.dim\_date;

进入hive查看一下表 select \* from dim\_date limit 100;



至此这个简单的数仓模型就搭建完成了。

## 2.8 ETL过程

### 2.8.1 Extract数据抽取

数据仓库模型搭建完成后,下一步要做的工作就是数据仓库中数据的填充,也就是我们的ETL过程,ETL可以说是数据仓库系统中最基础也是最重要的环节.我们前面讲过数据仓库的数据来源是业务系统，业务系统可能同时使用多种数据库系统，这些系统在物理上各自独立，在逻辑上有相互联系，数据仓库要从中捕获变化的数据，实现增量数据抽取。

#### 2.8.1.1 抽取方式

从源数据导入到数据仓库或者贴源层有两种方式，从源数据拉取数据（pull）和请求源数据推送到数据仓库（push）。一般来讲，后一种方式需要增加业务系统的功能才能进行推送，这个在现实情况中不大行的通，一方面影响业务系统的性能，另一方面增加开发者的工作量，理论上讲，数据仓库不应该要求对源系统做任何的改造，因此一般都采用拉取数据的方式。

#### 2.8.1.2 抽取类型

确定了数据的抽取方式，还需要确定数据的抽取类型，即全量抽取还是增量抽取。如果数据量小并且容易处理，一般采用全量抽取即可。这种方式适合基础编码类的数据源，最典型的是我们通常所说的数据字典类的码表，在数据仓库中就是维表。如果数据量很大，就只能抽取变化的源数据，即最后一次抽取以来发生了变化的数据。

#### 2.8.1.3 抽取方案

* 导出为文本方式

以文件文件的形式交换数据是一种可行的通用方法，一般需要将数据源库中的数据导出成预定的义好的分隔符，比如逗号分隔，然后用hadoop的dfs命令将文件上传到hive表对应目录中，或者使用hive的load data local inpath 语句将数据装载的目标表中。

大多数据数据库都提供导出的工具或命令。比如mysql中，可以使用 select … into outfile

select \* into outfile /tmp/t1.txt’ fields terminated by ',' from t1;

如果不带查询条件或者导出整个库，可以使用mysqldump命令行工具，可以一次导出多个表，多个库或所有库的数据库。

mysqldump -uroot -p123456 –databases test > /tmp/test.sql

* sqoop抽取数据

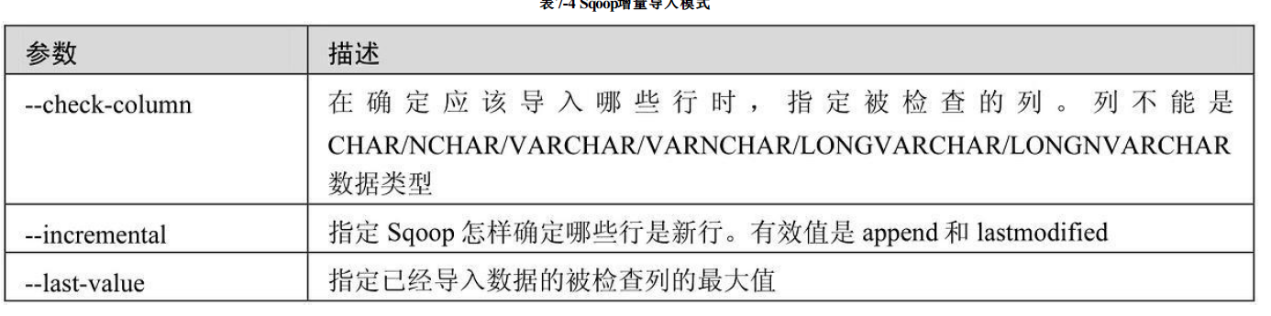
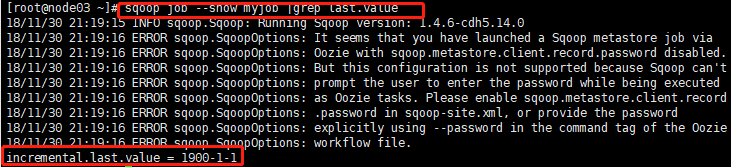
Sqoop是用来实现结构型数据（如关系数据库）和Hadoop之间进行数据迁移的工具,我们可以使用sqoop来对我们的销售案例进行数据抽取。

|  |  |  |  |
| --- | --- | --- | --- |
| 源数据表 | RDS库中的表(ODS) | DW库中的表 | 抽取模式 |
| customer | customer | dim\_customer | 整体拉取 |
| product | product | dim\_product | 整体拉取 |
| sales\_order | sales\_order | fact*sales*order dim\_order | 增量拉取 |

1. 全量导入

* 对于 customer、 product这两个表采用整体拉取的方式抽取数据。ETL通常是按一个固定的时间间隔,周期性定时执行,因此对于整体拉取的方式而言,每次导入的数据需要覆盖上次导入的数据。
* sqoop import \  
  --connect jdbc:mysql://192.168.137.77:3306/sales\_source \  
  --username root --password 123456 \  
  --table customer \  
  --hive-import \  
  --hive-table sales\_rds.customer \  
  --hive-overwrite \  
  --target-dir temp  
  ;  
    
  sqoop import \  
  --connect jdbc:mysql://192.168.137.77:3306/sales\_source \  
  --username root --password 123456 \  
  --table product \  
  --hive-import \  
  --hive-table sales\_rds.product \  
  --hive-overwrite \  
  --target-dir temp  
  ;
* Sqop中提供了 hive-overwrite参数实现覆盖导入。 hive-overwrite的另一个作用是提供了一个幂等操作的选择。所谓幂等操作指的是其执行任意多次所产生的影响均与次执行的影响相同。这样就能在导入失败或修复bug后可以再次执行该操作,而不用担心重复执行会对系统造成数据混乱。具体Sqoop命令如上,其中  
  hive-import参数表示向hive表导入,hive-  
  table参数指定目标hive库表。
* **注意**
* 1) 这个脚本里面是里面的命令没有加路径，需要将sqoop的路径配到环境变量中
* 2) 要设置target-dir不然会报目录存在的错误
* 3) 出错的解决方案：
* Caused by: java.lang.ClassNotFoundException: org.apache.hadoop.hive.conf.HiveConf
* 复制hive lib 包下面的hive-common-1.1.0-cdh5.13.2.jar 至sqoop lib 包下
* Caused by: java.lang.ClassNotFoundException: org.apache.hadoop.hive.shims.ShimLoader
* 拷贝 hive lib 包下 hive-exec-1.1.0-cdh5.13.2.jar 至 sqoop 的lib包下

1. 增量导入

* Sqoop提供增量导入模式，用于只导入新的数据行, Sqoop支持两种类型的增量导入：append和lastmodified。可以使用--incremental参数指定增量导入的类型。
* 
* Sqoop支持两种类型的增量导入：append和lastmodified。可以使用--incremental参数指定增量导入的类型。
* sqoop job \  
  --create myjob -- import \  
  --connect jdbc:mysql://192.168.137.77:3306/sales\_source \  
  --username root \  
  --password 123456 \  
  --query "select \* from sales\_order where \$CONDITIONS" \  
  --hive-import \  
  --hive-database sales\_rds \  
  --hive-table sales\_order \  
  --split-by order\_number \  
  --fields-terminated-by '\t' \  
  --lines-terminated-by '\n' \  
  --target-dir /hive/warehouse/sales\_rds.db/sales\_order \  
  --hive-delims-replacement ' ' \  
  --incremental append \  
  --check-column entry\_date \  
  --password-file /sqoop/pwd/sqoopPWD.pwd  
  --last-value '1900-1-1'  
  ;
* 查看sqoop的job
* sqoop job --list  
  sqoop job --show myjob
* sqoop执行任务的时候会提示执行密码，需要建立密码文件并指定--password-file 文件，文件需要存放在hdfs上，权限为400
* echo -n "123456" >sqoopPWD.pwd  
  hdfs dfs -mkdir -p /sqoop/pwd  
  hdfs dfs -put sqoopPWD.pwd /sqoop/pwd  
  hdfs dfs -chmod 400 /sqoop/pwd/sqoopPWD.pwd  
  将sqoop的job中去掉--password增加  
   --password-file /sqoop/pwd/sqoopPWD.pwd  
    
    
    
  sqoop job --create myjob -- import --connect jdbc:mysql://192.168.137.77:3306/sales\_source \  
  --username root \  
  --password-file /sqoop/pwd/sqoopPWD.pwd \  
  --query "select \* from sales\_order where \$CONDITIONS" \  
  --hive-import \  
  --hive-database sales\_rds \  
  --hive-table sales\_order \  
  --split-by order\_number \  
  --fields-terminated-by '\t' \  
  --lines-terminated-by '\n' \  
  --target-dir /hive/warehouse/sales\_rds.db/sales\_order \  
  --hive-delims-replacement ' ' \  
  --incremental append \  
  --check-column entry\_date \  
  --last-value '1900-1-1' \
* 查看此时作业中保存的last-value，结果如下所示。
* 
* 执行作业
* sqoop job --exec myjob
* 因为last-value的值为'1900-01-01'，所以这次会导入全部数据。执行完成后，查看作业，看看last-value值。
* 往源库增加两条数据。
* USE sales\_source;  
  SET @customer\_number := FLOOR(1 + RAND()\* 6);  
  SET @product\_code := FLOOR(1 + RAND()\* 2);  
  SET @order\_date := '2018-11-23' ;  
  SET @amount := FLOOR(100003 + RAND()\* 9000);  
  INSERT INTO sales\_order  
  VALUES(100003,@customer\_number,@product\_code,@order\_date,@order\_date,@amount);  
    
  SET @customer\_number := FLOOR(1 + RAND()\* 6);  
  SET@product\_code := FLOOR(1 + RAND()\* 2);  
  SET @order\_date := '2018-11-24';  
  SET @amount := FLOOR(1000 + RAND()\* 9000);  
  INSERT INTO sales\_order  
  VALUES(100004,@customer\_number,@product\_code,@order\_date,@order\_date,@amount);  
  COMMIT;
* 再次执行sqoop作业，这次只会导入比上次last-value大的数据。

### 2.8.2 Transform数据转换

数据转换是将数据进行重构以及标准化，消除数据的不一致，处理缺失数据，转换最主要的任务就是数据清洗。数据清洗是对数据进行重新审查和校验的过程，目的在于删除重复信息，纠正存在的错误，并提供数据一致性。

数据在采集或进行外部接口调用过程中可能会产生不符合要求的“脏数据”，下面介绍下对这些脏数据的产生原因及处理  
  
1 格式内容问题产生的原因  
 \* 不同数据源采集而来的数据内容和格式定义不一致  
 \* 时间、日期格式不一致清洗 根据实际情况，把时间/日期数据库转换成统一的表示方式。  
 \* 数据类型不符清洗  
  
2 逻辑错误清洗  
 \* 数据重复清洗  
 \* 数据不完全相同，但从业务角度看待数据是同一个数据，如页面埋点时，进入页面和退出页面都会上报一次数据，只有时间不一样，其他字段相同，在统计pv/uv时应该进行去重。  
 \* 矛盾内容的修正 如身份证号可以验证我们的年龄，然后我们可以根据字段的数据来源，判断哪个字段提供的信息更为可靠来做修正  
  
3 缺失值的清洗  
 \* 数据值缺失是数据分析中经常遇到的问题之一。  
 \* 造成原因 有些对象的某个或某些属性是不可用的，信息被遗漏，或者实时性高还未来得及做出判断  
 \* 删除元组 这种只适合在对象有多个属性缺失值、被删除的含缺失值的对象与初始数据集的数据量相比非常小的情况下非常有效  
 \* 数据填充用一定的值去填充空值，从而使信息表完备化  
 \* 去空处理 null-string '\\N'  
   
4 不符合业务需求（或挖掘分析需求）的数据  
 \* 越界数据 如温度 采集如果为10000已经是设备故障等原因导致的，但对后续分析没有意义；视频或文章的播放或查阅时长，不同公司有不同的行业要求，假如要求时长>5m认为是又意义的数据

### 2.8.3 Load数据装载

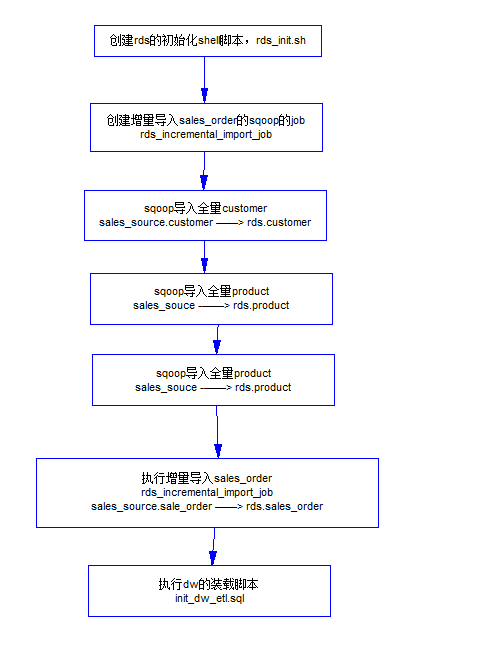
数据仓库搭建完成后，有一些预装载的操作先进行，比如时间维度以及其他不从源数据获取的一些数据进行预装载，接着就是初始装载和定期装载。

#### 2.8.3.1 初始装载

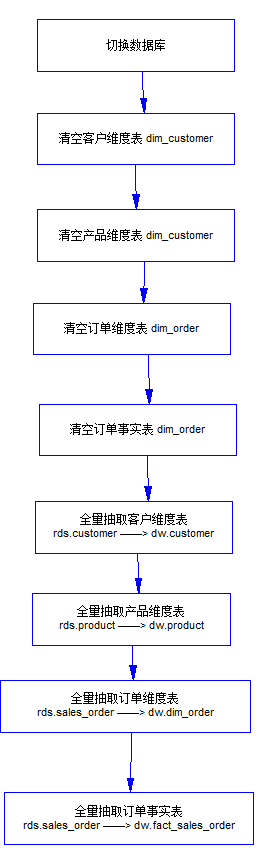
编写数据源映射关系表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 业务表 | 源数据 | 源类型 | 贴源层 | 类型 | 数仓 | 类型 |
| 客户表 | customer | mysql表 | rds.customer | Hive表 | dw.dim\_customer | hive表 |
| 产品表 | product | mysql表 | rds. product | Hive表 | dw.dim\_product | Hive表 |
| 销售订单 | Sales\_order | mysql表 | rds.sales\_order | Hive表 | dw.dim*order dw.fact*sales\_order | Hive表 |

装载流程



dw装载子流程，init*dw*etl.sql



脚本编写

* 初始化dw库的脚本init*dw*etl.sql

USE dw;  
-- 清空表  
TRUNCATE TABLE dim\_customer;  
TRUNCATE TABLE dim\_product;  
TRUNCATE TABLE dim\_order;  
TRUNCATE TABLE fact\_sales\_order;  
  
-- 装载客户维度表  
INSERT INTO dim\_customer (customer\_sk,customer\_number,customer\_name,customer\_street\_address,customer\_zip\_code,customer\_city,customer\_state,`version`,effective\_date,expiry\_date)  
SELECT  
 t1.customer\_number,  
 t1.customer\_number,   
 t1.customer\_name,   
 t1.customer\_street\_address,  
 t1.customer\_zip\_code,   
 t1.customer\_city,   
 t1.customer\_state,   
 1,  
 '2018-01-01',   
 '2050-01-01'  
FROM sales\_rds.customer t1;  
   
-- 装载产品维度表  
INSERT INTO dim\_product (product\_sk,product\_code,product\_name,product\_category,`version`,effective\_date,expiry\_date)  
SELECT t1.product\_code,  
 product\_code,   
 product\_name,   
 product\_category,   
 1,  
 '2016-03-01',   
 '2050-01-01'  
FROM sales\_rds.product t1;  
   
-- 装载订单维度表  
INSERT INTO dim\_order(order\_sk,order\_number,`version`,effective\_date,expiry\_date)  
SELECT t1.order\_number,  
 order\_number,   
 1,   
 order\_date,   
 '2050-01-01'  
FROM sales\_rds.sales\_order t1;  
   
-- 装载销售订单事实表  
INSERT INTO fact\_sales\_order  
SELECT order\_sk, customer\_sk, product\_sk, date\_sk, order\_amount  
FROM sales\_rds.sales\_order a  
JOIN dim\_order b ON a.order\_number = b.order\_number  
JOIN dim\_customer c ON a.customer\_number = c.customer\_number  
JOIN dim\_product d ON a.product\_code = d.product\_code  
JOIN dim\_date e ON to\_date(a.order\_date) = e.`date`;

* 装载数据总的shell脚本init*all*etl.sh

#!/bin/bash  
# 建立Sqoop增量导入作业，以entry\_date作为检查列，初始的last-value是1900-1-1  
sqoop job --create myjob -- import --connect jdbc:mysql://192.168.137.77:3306/sales\_source \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--query "select \* from sales\_order where \$CONDITIONS" \  
--hive-import \  
--hive-database sales\_rds \  
--hive-table sales\_order \  
--split-by order\_number \  
--fields-terminated-by '\t' \  
--lines-terminated-by '\n' \  
--target-dir /hive/warehouse/sales\_rds.db/sales\_order \  
--hive-delims-replacement ' ' \  
--incremental append \  
--check-column entry\_date \  
--last-value '1900-1-1' \  
# 首次抽取，将全部数据导入RDS库  
sqoop import --connect jdbc:mysql://192.168.137.77:3306/sales\_source --username root --password 123456 --table customer --hive-import --hive-table sales\_rds.customer --hive-overwrite --target-dir temp  
sqoop import --connect jdbc:mysql://192.168.137.77:3306/sales\_source --username root --password 123456 --table product --hive-import --hive-table sales\_rds.product --hive-overwrite --target-dir temp  
# 清空rds层的订单表  
hive -e "TRUNCATE TABLE sales\_rds.sales\_order"  
# 执行rds层的增量抽取  
sqoop job --exec myjob  
# 装载dw层  
hive -f ./init\_dw\_etl.sql

检验数据

装载数据以后要检查一下数据的正确性，源数据的条数，抽取的条数,拿customer举例

select count(1) from sales\_source.customer  
select count(1) from rds.customer  
select count(1) from dw.dim\_customer

如果条数一致可以选几条数据查看是否都存在，一般条数一致基本上就问题不大，如果条数不一致，就需要查看差异在什么地方，查找出错的原因，调整脚本。

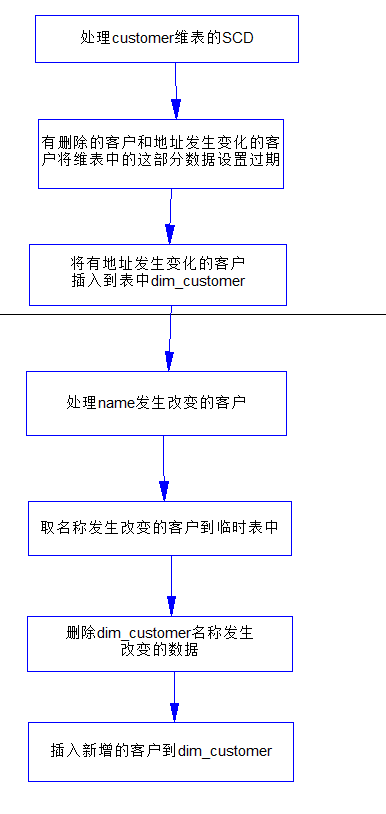
#### 2.8.3.2 定期装载

定期装载涉及表

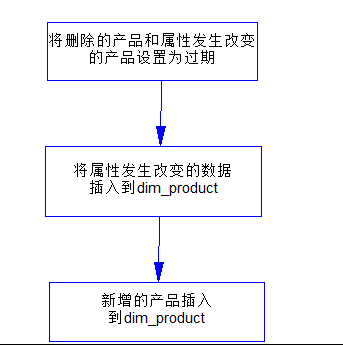
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 源 | Rds | 数仓 | 抽取模式 | 维度处理 |
| customer | customer | dim\_customer | 全量拉取 | address上SCD2，name上SCD1 |
| product | product | dim\_product | 全量拉取 | 均SCD2 |
| sales\_order | sales\_order | dim\_order | 每天CDC | 唯一订单号 |
| Fact*sales*order | 每天CDC | N/A |  |  |
| N/A | N/A | dim\_date | N/A | 预装载 |

处理流程

* 处理dim\_customer



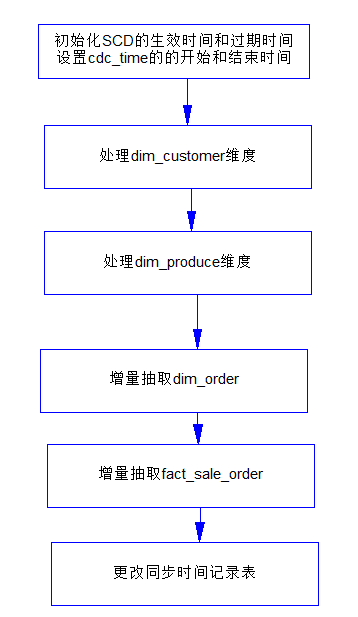
* 处理dim\_product



处理订单维度表dim\_order,直接抽取增量数据插入即可

处理订单事实表fact*sales*order,直接抽取即可。

* 整体流程图如下



脚本编写schedule*daily*etl.sql

-- 设置scd的生效时间和过期时间  
SET hivevar:cur\_date = CURRENT\_DATE();   
SET hivevar:pre\_date = DATE\_ADD(${hivevar:cur\_date},-1);  
SET hivevar:max\_date = CAST('2050-01-01' AS DATE);  
  
-- 设置cdc的开始结束日期  
INSERT overwrite TABLE sales\_rds.cdc\_time  
SELECT end\_date, ${hivevar:cur\_date} FROM sales\_rds.cdc\_time;  
  
-- 装载customer维度  
-- 获取源数据中被删除的客户和地址发生改变的客户，将这些数据设置为过期时间，即当前时间的前一天  
UPDATE dim\_customer  
SET expiry\_date = ${hivevar:pre\_date}  
WHERE dim\_customer.customer\_sk IN(SELECT  
 a.customer\_sk  
 FROM (SELECT  
 customer\_sk,  
 customer\_number,  
 customer\_street\_address  
 FROM dim\_customer  
 WHERE expiry\_date = ${hivevar:max\_date}) a  
 LEFT JOIN sales\_rds.customer b ON a.customer\_number = b.customer\_number  
 WHERE b.customer\_number IS NULL  
 OR a.customer\_street\_address <> b.customer\_street\_address);  
  
-- 将有地址变化的插入到dim\_customer表，如果有相同数据存在有不过期的数据则不插入  
INSERT INTO dim\_customer  
SELECT t1.customer\_number,  
 t1.customer\_number,  
 t1.customer\_name,  
 t1.customer\_street\_address,  
 t1.customer\_zip\_code,  
 t1.customer\_city,  
 t1.customer\_state,  
 t1.version,  
 t1.effective\_date,  
 t1.expiry\_date  
FROM(SELECT  
 t2.customer\_number customer\_number,  
 t2.customer\_name customer\_name,  
 t2.customer\_street\_address customer\_street\_address,  
 t2.customer\_zip\_code,  
 t2.customer\_city,  
 t2.customer\_state,  
 t1.version + 1 `version`,  
 ${hivevar:pre\_date} effective\_date,  
 ${hivevar:max\_date} expiry\_date  
FROM dim\_customer t1  
INNER JOIN sales\_rds.customer t2 ON t1.customer\_number = t2.customer\_number  
 AND t1.expiry\_date = ${hivevar:pre\_date}  
LEFT JOIN dim\_customer t3 ON t1.customer\_number = t3.customer\_number  
 AND t3.expiry\_date = ${hivevar:max\_date}  
WHERE t1.customer\_street\_address <> t2.customer\_street\_address   
 AND t3.customer\_sk IS NULL  
) t1;  
   
  
-- 处理customer\_name列上的scd1，覆盖  
-- 不进行更新，将源数据中的name列有变化的数据提取出来，放入临时表  
-- 将 dim\_couster中这些数据删除、  
-- 将临时表中的数据插入  
DROP TABLE IF EXISTS tmp;  
CREATE TABLE tmp AS  
SELECT a.customer\_sk,  
 a.customer\_number,  
 b.customer\_name,  
 a.customer\_street\_address,  
 a.customer\_zip\_code,  
 a.customer\_city,  
 a.customer\_state,  
 a.version,  
 a.effective\_date,  
 a.expiry\_date  
FROM dim\_customer a   
JOIN sales\_rds.customer b ON a.customer\_number = b.customer\_number   
Where a.customer\_name <> b.customer\_name ;  
-- 删除数据   
DELETE FROM  
dim\_customer WHERE  
dim\_customer.customer\_sk IN (SELECT customer\_sk FROM tmp);  
  
-- 插入数据  
INSERT INTO dim\_customer   
SELECT \* FROM tmp;  
  
  
-- 处理新增的customer记录  
INSERT INTO dim\_customer  
SELECT t1.customer\_number,  
 t1.customer\_number,  
 t1.customer\_name,  
 t1.customer\_street\_address,  
 t1.customer\_zip\_code,  
 t1.customer\_city,  
 t1.customer\_state,  
 1,  
 ${hivevar:pre\_date},  
 ${hivevar:max\_date}  
FROM( SELECT t1.\*   
 FROM sales\_rds.customer t1   
 LEFT JOIN dim\_customer t2 ON t1.customer\_number = t2.customer\_number  
WHERE t2.customer\_sk IS NULL ) t1;  
  
-- 装载product维度  
-- 取源数据中删除或者属性发生变化的产品，将对应  
UPDATE dim\_product  
SET expiry\_date = ${hivevar:pre\_date}  
WHERE dim\_product.product\_sk IN(SELECT a.product\_sk  
 FROM(SELECT product\_sk,  
 product\_code,  
 product\_name,  
 product\_category  
 FROM dim\_product   
 WHERE expiry\_date = ${hivevar:max\_date}) a   
 LEFT JOIN sales\_rds.product b ON a.product\_code = b.product\_code  
 WHERE b.product\_code IS NULL   
 OR (a.product\_name <> b.product\_name OR a.product\_category <> b.product\_category));  
   
-- 处理product\_name、product\_category列上scd2的新增行  
INSERT INTO dim\_product  
SELECT ORDER BY t1.product\_code,  
 t1.product\_code,  
 t1.product\_name,  
 t1.product\_category,  
 t1.version,  
 t1.effective\_date,  
 t1.expiry\_date  
FROM( SELECT t2.product\_code product\_code,  
 t2.product\_name product\_name,  
 t2.product\_category product\_category,  
 t1.version + 1 `version`,  
 ${hivevar:pre\_date} effective\_date,  
 ${hivevar:max\_date} expiry\_date  
FROM dim\_product t1  
INNER JOIN sales\_rds.product t2 ON t1.product\_code = t2.product\_code  
 AND t1.expiry\_date = ${hivevar:pre\_date}  
LEFT JOIN dim\_product t3 ON t1.product\_code = t3.product\_code   
 AND t3.expiry\_date = ${hivevar:max\_date}  
WHERE(t1.product\_name <> t2.product\_name   
 OR t1.product\_category <> t2.product\_category)   
 AND t3.product\_sk IS NULL  
) t1;  
   
-- 处理新增的 product 记录  
INSERT INTO dim\_product  
SELECT t1.product\_code,  
 t1.product\_code,  
 t1.product\_name,  
 t1.product\_category,  
 1,  
 ${hivevar:pre\_date},  
 ${hivevar:max\_date}  
FROM( SELECT t1.\*   
 FROM sales\_rds.product t1   
 LEFT JOIN dim\_product t2 ON t1.product\_code = t2.product\_code  
 WHERE t2.product\_sk IS NULL  
 ) t1;  
  
-- 装载order维度  
INSERT INTO dim\_order  
SELECT t1.order\_number,  
 t1.order\_number,  
 t1.version,  
 t1.effective\_date,  
 t1.expiry\_date  
FROM( SELECT order\_number order\_number,  
 1 `version`,  
 order\_date effective\_date,  
 '2050-01-01' expiry\_date  
 FROM sales\_rds.sales\_order, rds.cdc\_time  
 WHERE entry\_date >= last\_load AND entry\_date < current\_load ) t1;  
  
  
-- 装载销售订单事实表  
INSERT INTO sales\_fact\_sales\_order  
SELECT order\_sk,  
 customer\_sk,  
 product\_sk,  
 date\_sk,  
 order\_amount  
FROM sales\_rds.sales\_order a,  
 dim\_order b,  
 dim\_customer c,  
 dim\_product d,  
 date\_dim e,  
 rds.cdc\_time f  
WHERE a.order\_number = b.order\_number  
 AND a.customer\_number = c.customer\_number  
 AND a.order\_date >= c.effective\_date  
 AND a.order\_date < c.expiry\_date  
 AND a.product\_code = d.product\_code  
 AND a.order\_date >= d.effective\_date  
 AND a.order\_date < d.expiry\_date  
 AND to\_date(a.order\_date) = e.date  
 AND a.entry\_date >= f.start\_date   
 AND a.entry\_date < f.end\_date ;

shell脚本 schedule\_daily.sh

#!/bin/bash  
# 整体拉取rds层customer、product表数据  
sqoop import --connect jdbc:mysql://192.168.25.120:3306/sales\_source --username root --password 123456 --table customer --hive-import --hive-table sales\_rds.customer --hive-overwrite --target-dir temp  
  
sqoop import --connect jdbc:mysql://192.168.25.120:3306/sales\_source --username root --password 123456 --table product --hive-import --hive-table sales\_rds.product --hive-overwrite --target-dir temp  
  
# 执行增量导入  
sqoop job --exec myjob  
# 调用 schedule\_daily\_etl.sql 文件执行定期装载  
hive -f ./schedule\_daily\_etl.sql

检验数据

当任务开始跑以后,要验证几天的数据，看数据是否正确，如果数据存在问题，可能是脚本存在bug，需要做调整

## 2.9 ETL自动化

上面我们已经把每天需要增量跑的任务编写完了，这个任务需要每天定时去执行，一般来讲需要制定一个任务计划表，这个任务来记录所有的任务的执行周期，执行时间，负责人，联系方式等信息，当一个项目很大，有很多的任务，每个任务的调度周期和时间可能是不一样的，比如有日结的数据，也有月结的数据，日结是每天要进行汇总计算的，月结是每月月底要进行计算的，可能日结是项目组中的一个人写的，月结是令一个人写的，为保证整体项目的在试运行期间的平稳运行，一般要有这么一个表格，当项目有问题时，找相应的负责人。

我们的示例比较简单，只有一个调度任务

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能模块 | 调度文件 | 调度周期 | 负责人 | 联系方式 |
| 增量抽取数据 | schedule\_daily.sh | 每天凌晨2点 | 张三 | 13912345678 |

关于调度的执行周期，一般选在业务量的低峰，上线也是一样，在业务量最低的时候去进行操作，这个时候如果出现问题，进行回滚，业务影响不大，所以一般选择在凌晨2-4点之间，关于上线也多说一句，一般来讲公司有几个环境，开发环境，测试环境和生产环境。一般我们在开发的时候使用的数据库连的都是开发环境，开发环境一般性能不高，数据库里面的数据也是样本数据，量很小，只是为了开发使用，测试环境一般是为了模拟上线用的，预上线就是上线到测试环境，测试环境一般来讲跟生产环境是相同的，性能跟生产环境不能说一模一样，至少也是要差不多，开发人员开发完成后需要预上线到测试环境，测试人员就开始对系统进行功能测试和压力测试，并且编写测试报告，测试报告通过，才能正式上线，上线需要跟运维提邮件申请，各负责人进行审批，运维人员进行上线，上线以后，测试人员需要进行拨测，一般来说上线的时候测试和开发以及运维都需要在现场，一旦发生问题，马上回滚。上线通常在周二和周四的晚上12点以后，避开业务的高峰期，一般选择凌晨2点左右，如果发生回滚，在4点以前搞不定，这次上线就失败了，底下去修复bug，等下次上线周期。回滚机制是必须要有的，如果上线失败导致业务出现大量的问题，这个时候就发生了生产事故，事故一般会定级，当级别比较高的时候，会进行邮件的通报，甚至全公司通报，每个审批的负责人都要进行相应的处理比如扣绩效罚款等，开发人员也会相应的处罚但是没有负责人承担的重，所以其实一旦发生问题程序员就要跑路真实况发生的几率并不大，除非是毁灭性的不可恢复的，否则发生问题各级的负责人会受到处罚，还落不到程序员头上，但是负责人被处罚了，部门可能会秋后算账，你的年终绩效可能要受到影响，这就看老大的处事风格了，大家了解一下即可。

定时执行任务最简单的就是使用crontab，我们使用crontab命令来创建一个任务，定期运行脚本。

# 修改文件属性为可执行  
chmod 755 root/ schedule\_daily.sh  
# 编辑crontab文件内容  
crontab -e  
# 打开编辑页面，插入一行，指定每天2点执行定期装载作业，然后保存退出  
0 2 \* \* \* /root/ schedule\_daily.sh

也可以使用azkaban这样的框架进行可视化的这种工作流方式进行任务调度。

# 三 数据集市

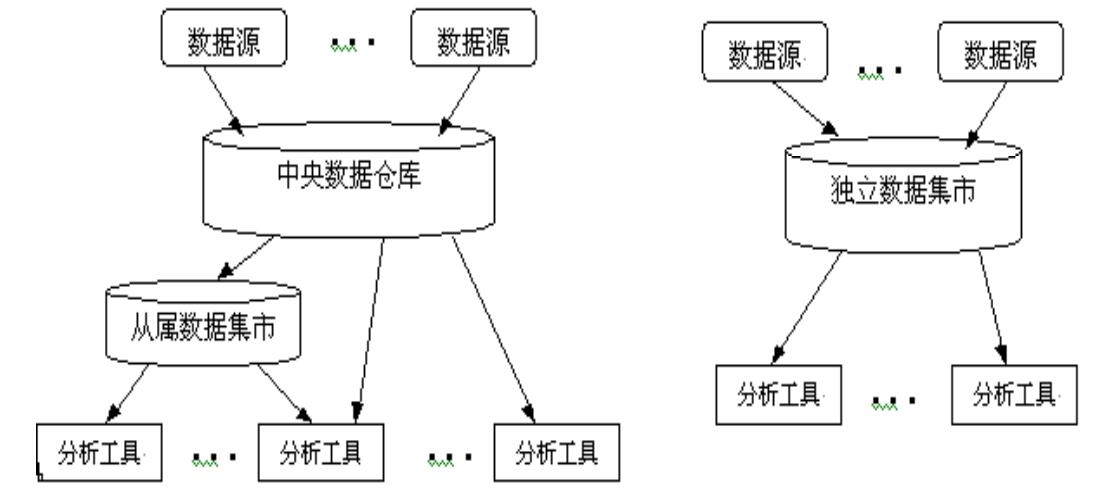
## 3.1 数据集市概念

* 建立数据集市的原因
* 数据仓库是一种反映主题的全局性数据组织。但是，全局性数据仓库往往太大，在实际应用中将它们按部门或个人分别建立反映各个子主题的局部性数据组织,它们即是数据集市。因此，有时我们也称它为部门数据仓库。
* 数据集市：是按照主题域组织的数据集合，用于支持部门级的决策。
* 主题域的概念
* 主题域通常是联系较为紧密的数据主题的集合。可以根据业务的关注点，将这些数据主题划分到不同的主题域。主题域的确定必须由最终用户和数据仓库的设计人员共同完成。  
    
  主题边界的划分应该按照以下规则来进行定义划分。   
  首先数据仓库中逻辑模型根据业务划分为多个主题域,主题域下面会涉及具体的实体表,维表以及关系实体,这些划分可以按照下面规则来进行划分。   
  a:每个主题域包含一个主要业务概念;   
  b:每个主题域包含一个主要交易业务概念,用一个或几个核心实体来表述。   
  c:主题域与主题域之间的核心实体不能重叠,核心实体间的关系实体则可以出现在两个主题域内;   
  d:每个主题域中包含几个关键的核心实体,且这几个核心实体间具有直接的关联关系。   
    
  主题域的另一种定义是：对某个主题进行分析后确定的主题的边界。分析主题域，确定要装载到数据仓库的主题是信息打包技术的第一步。而在进行数据仓库设计时，一般是一次先建立一个主题或企业全部主题中的一部分，因此在大多数数据仓库的设计过程中都有一个主题域的选择过程。主题域的确定必须由最终用户和数据仓库的设计人员共同完成。
* 例：在有关商品销售的数据仓库中可以建立多 个不同主题的数据集市：
  + 商品采购数据集市
  + 库房使用数据集市
  + 商品销售数据集市

## 3.2 集市分类

* 按照数据获取来源：
  + 独立型：直接从操作型环境获取数据
  + 从属型：从企业级数据仓库获取数据

从属集市和独立集市如下图：



* 集市建设途径分为如下两种：
  + 从 全局数据仓库 到 数据集市
  + 从 数据集市 到 全局数据仓库

## 3.3 集市和主题的区别

* 数据仓库与数据集市的关系类似于传统关系数据库系统中的基表与视图的关系。
* 数据集市的数据来自数据仓库，它是数据仓库中数据的一个部分与局部，是一个数据的再抽取与组织的过程。
* 具体区别如下：
* 

由于数据集市仅仅是数据仓库的某一部分，实施难度大大降低，并且能够满足公司内部部分业务部门的迫切需求，在初期获得了较大成功。但随着数据集市的不断增多，这种架构的缺陷也逐步显现。公司内部独立建设的数据集市由于遵循不同的标准和建设原则，以致多个数据集市的数据混乱和不一致。这就是数据孤岛（百度百科），也叫信息孤岛。

“企业发展到一定阶段，出现多个事业部，每个事业部都有各自数据，事业部之间的数据往往都各自存储，各自定义。每个事业部的数据就像一个个孤岛一样无法(或者极其困难)和企业内部的其他数据进行连接互动。”我们把这样的情况称为数据孤岛。简单说就是数据间缺乏关联性，数据库彼此无法兼容。

专业人士把数据孤岛分为物理性和逻辑性两种。物理性的数据孤岛指的是，数据在不同部门相互独立存储，独立维护，彼此间相互孤立，形成了物理上的孤岛。逻辑性的数据孤岛指的是，不同部门站在自己的角度对数据进行理解和定义，使得一些相同的数据被赋予了不同的含义，无形中加大了跨部门数据合作的沟通成本。

解决问题的方法只能是回归到数据仓库最初的基本建设原则上来。1998年，Inmon提出了新的BI架构CIF（CorporationInformation Factory，企业信息工厂），新架构在不同架构层次上采用不同的构件来满足不同的业务需求。

# 四 项目需求

## 4.1 项目需求分析

参考文档：高效运营支撑平台设计文档.docx

## 4.2 项目描述

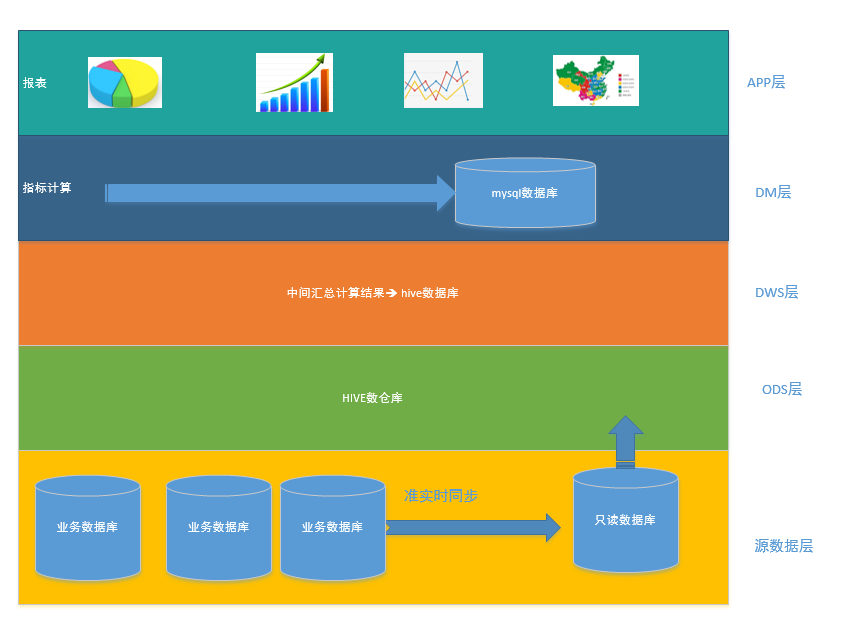
本平台是外卖行业类的自营业务的运营分析平台的一部分,通过对业务数据的分析,获得用户主题和订单的相关分析情况,生成用户的访问模型和用户的订单行为模型,对访问时段和下单区域分布进行分析和报表展示。

## 4.3 业务系统说明

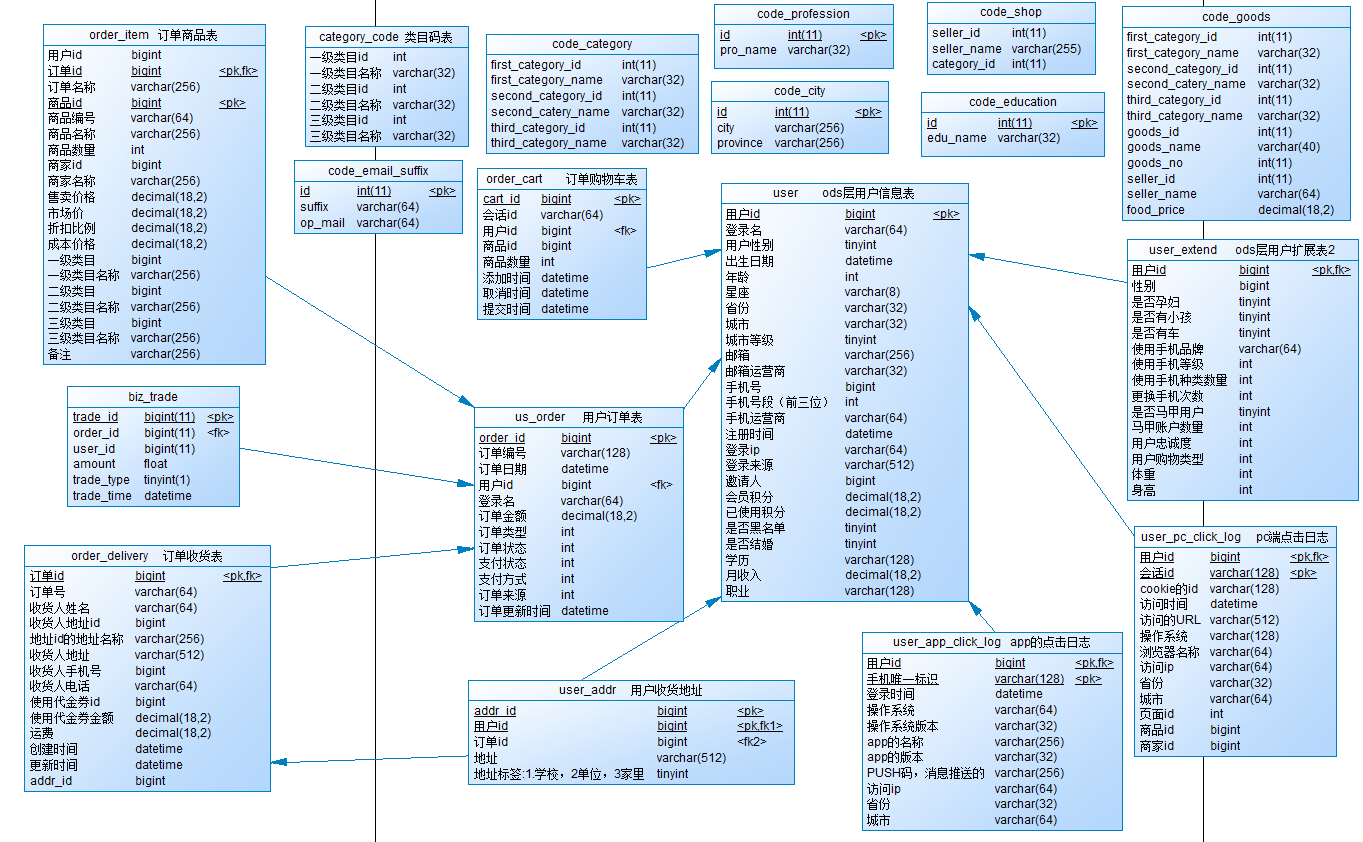
用户使用浏览器(PC端)和app对外卖平台进行访问,记录用户的注册信息,通过调研平台和数据挖掘记录用户的扩展信息,业务系统对用户的每次点击和下单信息记录到数据库中,并提供一台从库将主库的业务数据实时同步到从库中,同时分配对从库的只读访问权限,运营平台通过sqoop将数据抽取到运营分析平台,运营分析平台进行指标的计算,将结果存入到mysql中,将结果展示到报表。

业务数据存储在mysql中，可在mysql客户端执行qfbap.sql脚本，脚本中有event事件，可自动生成用户信息及业务数据，亦可手动执行 执行脚本.txt 中的脚本，分别生成用户信息、订单信息以及用户在pcc端和app端的点击日志信息。

具体架构如下



业务表数据模型



## 4.4 业务需求指标

#### 4.4.1 用户访问模型

1 生成用户基本宽表user\_basic

将用户的基本信息表和用户的扩展表进行合并形成用户宽表.

2 根据用户的访问日志信息,统计以下指标,结果存入用户访问模型表user\_visit

#### 4.4.2 用户订单模型

根据用户的订单业务，统计用户下单行为，生成用户订单模型表user\_order

#### 4.4.3 访问情况时段统计图

将指定的日期的每个时间区间的访问统计情况进行图形展示,横坐标是时间点,纵坐标是访问次数

#### 4.4.4 当天访问用户top20

以柱状图的形式显示当天访问用户的top20,横坐标为用户登录名,纵坐标为pv量

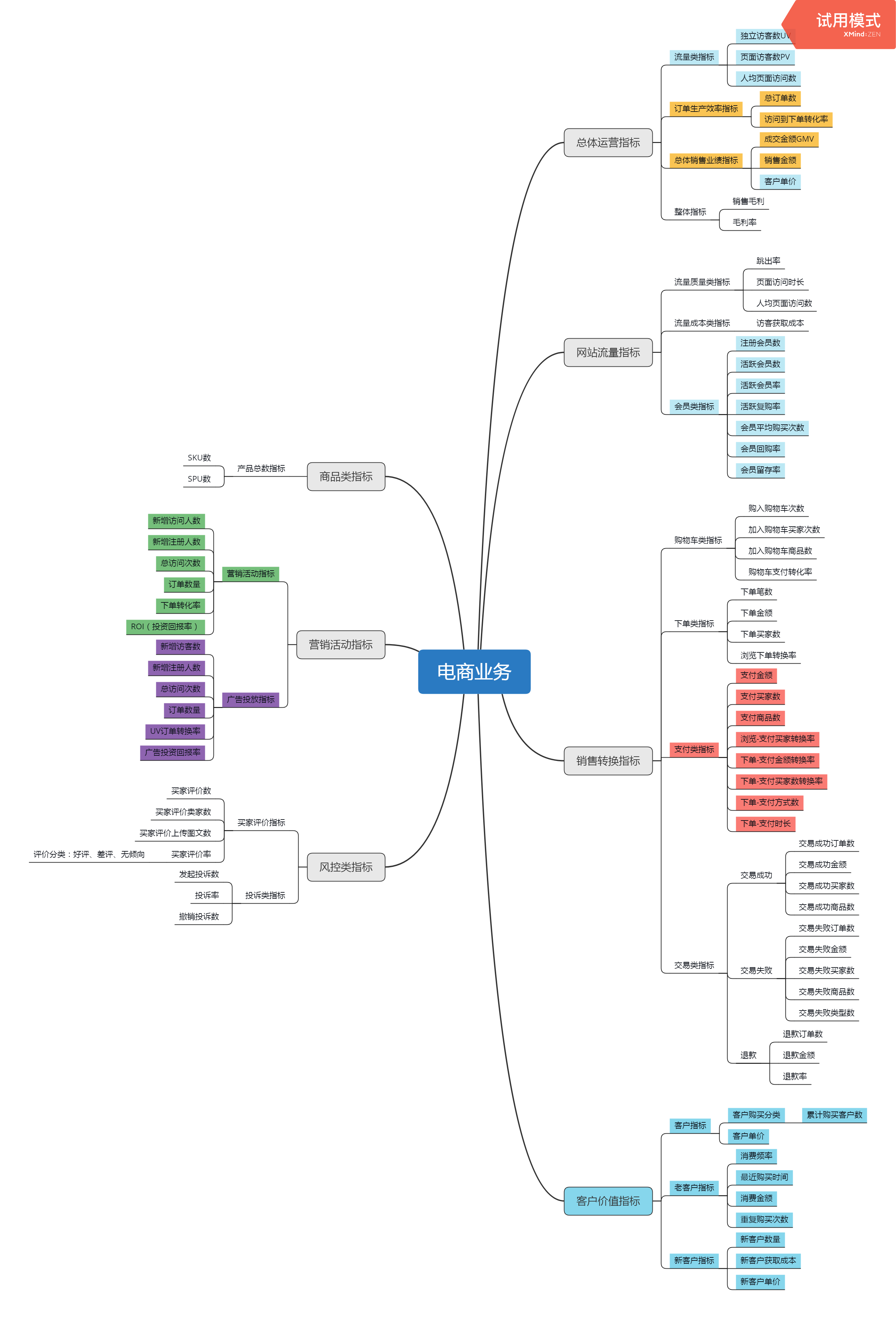
#### 4.4.5 订单情况时段统计图

将指定的日期的每个时间区间的订单统计情况进行图形展示,横坐标是时间点,纵坐标是下单数量

#### 4.4.6 当天区域下单量的分布图

以地图的方式显示当天下单金额的各省份的分布图。

#### 4.4.7 电商数据分析基础指标体系



# 五 项目建设实战

确认以上的内容：

查询频率：

业务部门的查询频率：每天的日报、每周的周报、每月的月报。。。。。。

企业高层的查询频率：每季度的季报、每年的年报等

数据存储：

数据集市：3个月、半年、一年等

数据仓库：保留企业所有的数据，随着运行时间，数据越来越大

查询方式：

自然年、财务年

响应时长：

实时响应、可延迟

## 5.1 技术选型

数据仓库项目架构确认

业务数据库:mysql

数据导入导出:sqoop(flume)

数据仓库:Hive

数据计算:Spark SQL(/MR)

数据导出到mysql

数据可视化 echarts 、tableU、kibana

任务调度:Azkaban(oozie)，每天凌晨2点执行

## 5.2 建立物理模型

### 5.2.1 物理模型(PDM)

物理数据模型用于描述数据在存储介质上的组织结构，与具体的DBMS相关。它是在逻辑数据模型的基础上，考虑各种具体的技术实现因素，进行数据库体系结构设计，真正实现数据在数据库中的表示。物理数据模型目标是为一个给定的逻辑模型选取一个最适合应用要求的物理结构。

物理模型：描述的是对象系统“如何做”、“如何实现”系统的物理过程。

物理模型设计所做的工作是根据信息系统的容量，复杂度，项目资源以及数据仓库项目自身（当然，也可以是非数据仓库项目）的 软件生命周期确定数据仓库系统的软硬件配置，数据仓库分层设计模式，数据的存储结构，确定索引策略，确定数据存放位置，确定存储分配等等。这部分应该是由项目经理和数据仓库架构师共同实施的。

### 5.2.2 模型设计

映射关系

ODS层

|  |  |  |
| --- | --- | --- |
| 源表(mysql) | 目标表(hive) | 映射关系 |
| user | qfbap*ods.ods*user |  |
| user\_extend | qfbap*ods.ods*user\_extend |  |
| user*pc*click\_log | qfbap*ods.ods*user*pc*click |  |
| user*app*click\_log | qfbap*ods.ods*user*app*click\_log |  |
| user\_addr | qfbap*ods.ods*user\_addr |  |
| user\_order | qfbap*ods.ods*user\_order |  |
| order\_item | qfbap*ods.ods*order\_item |  |
| order\_delivery | qfbap*ods.ods*order\_delivery |  |
| order\_cart | qfbap*ods.ods*order\_cart |  |
| biz\_trade | qfbap*ods.ods*biz\_trade |  |
| order\_cart | qfbap*ods.ods*order\_cart |  |

DWD层

|  |  |  |
| --- | --- | --- |
| 源表(hive) | 目标表(hive) | 映射关系 |
| qfbap*ods.ods*user | qfbap*dwd.dwd*qfbap\_user |  |
| qfbap*ods.ods*user\_extend | qfbap*dwd.dwd*user\_extend |  |
| qfbap*ods.ods*user*pc*click\_log | qfbap*dwd.dwd*user*pc*pv | 按session统计pv |
| qfbap*ods.ods*user*app*pv | qfbap*dwd.dwd*user*app*pv | 增加log*hour,对hour(log*time)à log\_hour |
| qfbap*ods.ods*user\_addr | qfbap*dwd.dwd*user\_addr |  |
| qfbap*ods.ods*user\_order | qfbap*dwd.dwd*user\_order |  |
| qfbap*ods.ods*order\_item | qfbap*dwd.dwd*order\_item |  |
| qfbap*ods.ods*order\_delivery | qfbap*dwd.dwd*order\_delivery |  |
| qfbap*ods.ods*order\_cart | qfbap*dwd.dwd*order\_cart |  |
| qfbap*ods.ods*biz\_trade | qfbap*dwd.dwd*biz\_trade |  |
| qfbap*ods.ods*order\_cart | qfbap*dwd.dwd*order\_cart |  |

DWS层

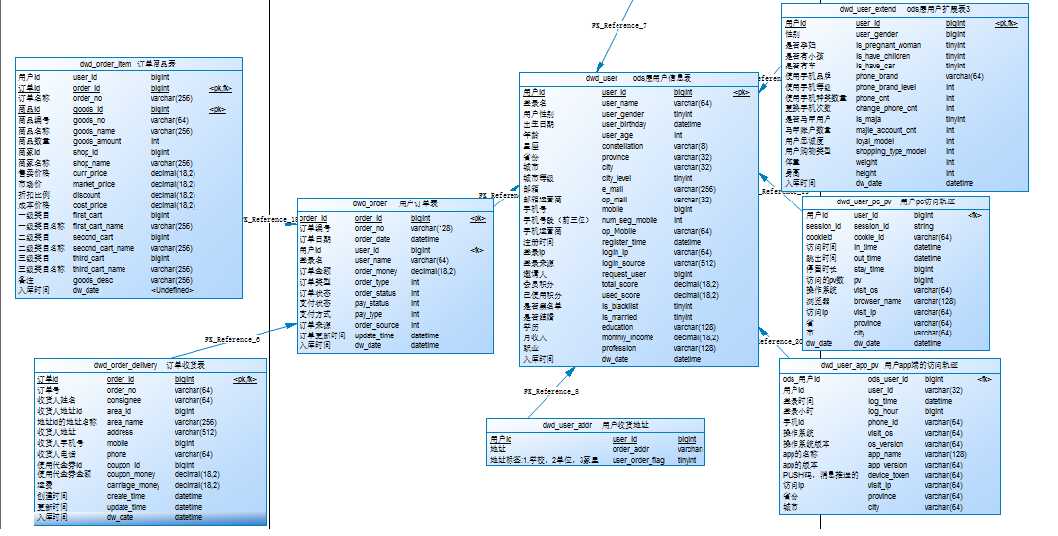
|  |  |  |
| --- | --- | --- |
| 源表(hive) | 目标表(hive) | 映射关系 |
| qfbap*dwd.dwd*user*pc*pv | qfbap*dws.dws*user*visit*month1 |  |
| qfbap*dwd.dwd*user\_order | qfbap*dws.dws*user*order*all |  |

DM层

|  |  |  |
| --- | --- | --- |
| 源表(hive) | 目标表(mysql) | 映射关系 |
| qfbap*dwd.dwd*user <br />qfbap*dwd.dwd*user\_extend | dm*user*basic | left join |
| dws*user*basic<br />qfbap*dwd.dwd*user*app*pv <br />qfbap*dwd.dwd*user*pc*pv qfbap*dws.dws*user*visit*month1 | qfbap*dm.dm*user\_visit | left join |

最终模型如下：

dwd层：



dws层：



dm层：



建表语句如下：

CREATE TABLE qfbap\_dwd.dwd\_biz\_trade (  
 trade\_id bigint ,  
 order\_id bigint ,  
 user\_id bigint ,  
 amount decimal(18,2) ,  
 trade\_type tinyint ,  
 trade\_time string ,  
 dw\_date string)   
partitioned by (dt string)  
location '/qfbap/dwd/dwd\_biz\_trade'  
;  
  
CREATE TABLE qfbap\_dwd.dwd\_cart (  
 cart\_id bigint ,  
 session\_id string ,  
 user\_id bigint ,  
 godwd\_id bigint ,  
 godwd\_num int ,  
 add\_time string ,  
 cancle\_time string ,  
 sumbit\_time string ,  
 create\_date string,  
 dw\_date string  
)   
partitioned by (dt string)  
location '/qfbap/dwd/dwd\_cart'  
;  
  
CREATE TABLE qfbap\_dwd.dwd\_code\_category (  
 first\_category\_id int ,  
 first\_category\_name string ,  
 second\_category\_id int ,  
 second\_catery\_name string ,  
 third\_category\_id int ,  
 third\_category\_name string ,  
 category\_id int ,  
 dw\_date string  
)   
location '/qfbap/dwd/dwd\_code\_category'  
;  
  
CREATE TABLE qfbap\_dwd.dwd\_order\_delivery (  
 order\_id bigint ,  
 order\_no string ,  
 consignee string ,  
 area\_id bigint ,  
 area\_name string ,  
 address string ,  
 mobile bigint ,  
 phone string ,  
 coupon\_id bigint ,  
 coupon\_money decimal(18,2) ,  
 carriage\_money decimal(18,2) ,  
 create\_time string ,  
 update\_time string ,  
 addr\_id bigint ,  
 dw\_date string  
)   
partitioned by (dt string)  
location '/qfbap/dwd/dwd\_order\_delivery'  
;  
  
CREATE TABLE qfbap\_dwd.dwd\_order\_item (  
 user\_id bigint ,  
 order\_id bigint ,  
 order\_no string ,  
 godwd\_id bigint ,  
 godwd\_no string ,  
 godwd\_name string ,  
 godwd\_amount int ,  
 shop\_id bigint ,  
 shop\_name string ,  
 curr\_price decimal(18,2) ,  
 market\_price decimal(18,2) ,  
 discount decimal(18,2) ,  
 cost\_price decimal(18,2) ,  
 first\_cart bigint ,  
 first\_cart\_name string ,  
 second\_cart bigint ,  
 second\_cart\_name string ,  
 third\_cart bigint ,  
 third\_cart\_name string ,  
 godwd\_desc string ,  
 dw\_date string   
)   
partitioned by (dt string)  
location '/qfbap/dwd/dwd\_order\_item';  
  
CREATE TABLE qfbap\_dwd.dwd\_us\_order (  
 order\_id bigint ,  
 order\_no string ,  
 order\_date string ,  
 user\_id bigint ,  
 user\_name string ,  
 order\_money decimal(18,2) ,  
 order\_type int ,  
 order\_status int ,  
 pay\_status int ,  
 pay\_type int ,  
 order\_source string ,  
 update\_time string ,  
 dw\_date string  
)  
partitioned by (dt string)  
location '/qfbap/dwd/dwd\_us\_order';  
  
CREATE TABLE qfbap\_dwd.dwd\_user (  
 user\_id bigint ,  
 user\_name string ,  
 user\_gender tinyint ,  
 user\_birthday string ,  
 user\_age int ,  
 constellation string ,  
 province string ,  
 city string ,  
 city\_level tinyint ,  
 e\_mail string ,  
 op\_mail string ,  
 mobile bigint ,  
 num\_seg\_mobile int ,  
 op\_Mobile string ,  
 register\_time string ,  
 login\_ip string ,  
 login\_source string ,  
 request\_user string ,  
 total\_score decimal(18,2) ,  
 used\_score decimal(18,2) ,  
 is\_blacklist tinyint ,  
 is\_married tinyint ,  
 education string ,  
 monthly\_income decimal(18,2) ,  
 profession string ,  
 dw\_date string   
)   
location '/qfbap/dwd/dwd\_user';  
  
CREATE TABLE qfbap\_dwd.dwd\_user\_addr (  
 user\_id bigint ,  
 order\_addr string ,  
 user\_order\_flag tinyint ,  
 addr\_id bigint ,  
 arear\_id int ,  
 dw\_date string  
)  
location '/qfbap/dwd/dwd\_user\_addr' ;  
  
  
CREATE TABLE qfbap\_dwd.dwd\_user\_extend (  
 user\_id bigint ,  
 user\_gender bigint ,  
 is\_pregnant\_woman tinyint ,  
 is\_have\_children tinyint ,  
 is\_have\_car tinyint ,  
 phone\_brand string ,  
 phone\_brand\_level string ,  
 phone\_cnt int ,  
 change\_phone\_cnt int ,  
 is\_maja tinyint ,  
 majia\_account\_cnt int ,  
 loyal\_model string ,  
 shopping\_type\_model string ,  
 weight int ,  
 height int ,  
 dw\_date string  
)   
location '/qfbap/dwd/dwd\_user\_extend';  
  
CREATE external TABLE qfbap\_dwd.dwd\_user\_app\_pv (  
 log\_id bigint ,  
 user\_id bigint ,  
 imei string ,  
 log\_time string ,  
 log\_hour string,  
 visit\_os string ,  
 os\_version string ,  
 app\_name string ,  
 app\_version string ,  
 device\_token string ,  
 visit\_ip string ,  
 province string ,  
 city string ,  
 dw\_date string  
)   
partitioned by (dt string)  
location '/qfbap/dwd/dwd\_user\_app\_pv'  
stored as orc;  
  
CREATE TABLE qfbap\_dwd.dwd\_user\_pc\_pv (  
 log\_id bigint ,  
 user\_id bigint ,  
 session\_id string ,  
 cookie\_id string ,  
 in\_time string,  
 out\_time string,  
 stay\_time bigint,  
 pv bigint,  
 visit\_os string ,  
 browser\_name string ,  
 visit\_ip string ,  
 province string ,  
 city string ,  
 dw\_date string  
)  
partitioned by (dt string)  
location '/qfbap/dwd/dwd\_user\_pc\_pv'  
;

DWS层设计：

-- 用户访问模型汇总  
create table if not exists qfbap\_dws.dws\_user\_visit\_month1  
(  
 user\_id bigint ,  
 type string,  
 content string,  
 cnt bigint,  
 rn bigint,  
 dw\_date string  
)  
partitioned by (dt string)  
stored as orc  
;  
  
-- 用户订单模型汇总  
create table if not exists qfbap\_dws.dws\_user\_order\_all  
(  
 user\_id bigint,  
 type string,  
 content string,  
 cnt bigint,  
 rn bigint,  
 dw\_date string  
)  
partitioned by (dt string)  
stored as orc  
;

## 5.3 ETL过程

根据ETL过程，确定项目的抽取方式为拉取方式。

抽取类型采用全量抽取、增量抽取。

对所有的维表采用全量抽取，对业务表采用增量抽取的方式

### 5.3.1 全量抽取案例

user表

先测试直接导入到hive表

sqoop import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password 123456 \  
--table user \  
--hive-import \  
--hive-overwrite \  
--hive-table qfbap\_ods.ods\_user \  
;

如果此时hive中还没有表，可以通过--create-hive-table参数自动创建表。但只能执行一次，所以一般创建表

CREATE DATABASE IF NOT EXISTS qfbap\_ods;  
  
CREATE TABLE IF NOT EXISTS qfbap\_ods.ods\_user (  
 user\_id bigint ,  
 user\_name string ,  
 user\_gender tinyint ,  
 user\_birthday string ,  
 user\_age int ,  
 constellation string ,  
 province string ,  
 city string ,  
 city\_level tinyint ,  
 e\_mail string ,  
 op\_mail string ,  
 mobile bigint ,  
 num\_seg\_mobile int ,  
 op\_Mobile string ,  
 register\_time string ,  
 login\_ip string ,  
 login\_source string ,  
 request\_user string ,  
 total\_score decimal(18,2) ,  
 used\_score decimal(18,2) ,  
 is\_blacklist tinyint ,  
 is\_married tinyint ,  
 education string ,  
 monthly\_income decimal(18,2) ,  
 profession string ,  
 create\_date string   
)   
;  
  
-- 查看表定义，主要下面需要知道表的HDFS存储路径  
show create table qfbap\_ods.ods\_user;

日期在hive中一般使用string类型

在hive中查询数据，测试是否导入成功。

通常做法，将数据导入到hdfs上

sqoop import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password 123456 \  
--table user \  
--delete-target-dir \  
--target-dir /hive/qfbap\_ods.db/ods\_user/ \  
--fields-terminated-by '\001' \  
;

--target-dir 指定导入的目标hdfs地址，这里是hive表的地址，可以在hive中通过show create table tableName;来查看表路径。

执行测试并查看结果后。

企业环境中，sqoop的命令一般采用sqoop job的方式存储

创建sqoop的job

sqoop job \  
--create qfbap\_ods\_user -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password 123456 \  
--table user \  
--delete-target-dir \  
--target-dir /hive/qfbap\_ods.db/ods\_user/ \  
--fields-terminated-by '\001' \  
;  
  
## sqoop常用job命令  
sqoop job --exec qfbap\_ods\_user  
sqoop job --list  
sqoop job --show qfbap\_ods\_user  
sqoop job --delete qfbap\_ods\_user

然后通过以下命令执行测试

sqoop job --exec qfbap\_ods\_user  
  
## 注意：  
## 观察执行过程。job方式执行时会提示输入执行密码  
  
## 现象  
## sqoop执行任务时会提示输入执行密码  
  
## 解决  
## 需要建立密码文件并指定--password-file，文件需要上传到hdfs之上，文件权限为400  
  
echo -n '123456' >> sqoopPWD.pwd ## mysql的登录密码  
hdfs dfs -mkdir -p /sqoop/pwd ## 在hdfs上创建密码目录  
hdfs dfs -put sqoopPWD.pwd /sqoop/pwd/ ## 上传密码文件  
hdfs dfs -chmod 400 /sqoop/pwd/sqoopPWD.pwd ## 修改文件权限

然后，修改job的配置，并重新执行

## 先删除job  
sqoop job --delete qfbap\_ods\_user  
  
## 修改配置并重新创建  
sqoop job \  
--create gp1907\_ods\_user -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--table user \  
--delete-target-dir \  
--target-dir /hive/gp1907\_ods.db/ods\_user/ \  
--fields-terminated-by '\001' \  
;  
  
## 执行job  
sqoop job --exec qfbap\_ods\_user  
  
## 观察执行过程，不再需要输入执行密码  
  
## 在hive客户端查询user表

### 5.3.2 增量抽取案例

us\_order表

业务表一般都设置值成分区表

回顾hive基础，思考数据如何导入到分区表

1、先将数据导入到临时目录，再通过hive命令load data 将数据加载到分区目录

load data inpath '/hive/qfbap\_ods.db/ods\_us\_order/dt=2020-01-01/\*' into table qfbap\_ods.ods\_us\_order partition(dt = '2020-01-01');

2、直接指定分区目录导入，然后添加分区元数据

alter table qfbap\_ods.ods\_us\_order add partition(dt='2020-01-01') location '/hive/qfbap\_ods.db/ods\_us\_order/dt=2020-01-01/'

先创建表

CREATE TABLE IF NOT EXISTS qfbap\_ods.ods\_us\_order (  
 order\_id bigint ,  
 order\_no string ,  
 order\_date string ,  
 user\_id bigint ,  
 user\_name string ,  
 order\_money decimal(18,2) ,  
 order\_type int ,  
 order\_status int ,  
 pay\_status int ,  
 pay\_type int ,  
 order\_source string ,  
 update\_time string   
)  
partitioned by (dt string)  
;

创建sqoop 的 job

方案1，导数据到临时目录

sqoop job \  
--create qfbap\_ods\_us\_order -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--table us\_order \  
--target-dir /hive/tempdb/ods\_us\_order/ \  
--fields-terminated-by '\001' \  
--check-column order\_id \  
--incremental append \  
--last-value 0 \  
;

执行job

sqoop job --exec qfbap\_ods\_us\_order

在hive客户端中执行以下语句，加载数据到分区目录

load data inpath '/hive/tempdb/ods\_us\_order/\*' into table qfbp\_ods.ods\_us\_order partition(dt='2020-01-01');

方案2，指定分区目录导入

sqoop job \  
--create qfbap\_ods\_us\_order -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--table us\_order \  
--target-dir /hive/qfbap\_ods.db/ods\_us\_order/dt=2020-01-01/ \  
--fields-terminated-by '\001' \  
--check-column order\_id \  
--incremental append \  
--last-value 0 \  
;

执行job

## 通过 -- 后面带参数名来动态指定导入分区目录   
sqoop job --exec qfbap\_ods\_us\_order -- --target-dir /hive/qfbap\_ods.db/ods\_us\_order/dt=2020-01-01/  
  
## 2020-01-01 动态指定

在hive客户端执行，增加表的分区，将分区目录加入到元数据管理

alter table qfbap\_ods.ods\_us\_order add partition(dt='2020-01-01') location '/hive/qfbap\_ods.db/ods\_us\_order/dt=2020-01-01/'

### 5.3.3 导入数据的问题

当数据量大时，导入数据可能发生错误。基于个人的全分布式集群，数据量大概在1.2G左右

错误代码

exception “GC Overhead limit exceeded

错误原因

Why Sqoop Import throws this exception?  
The answer is – During the process, RDBMS database (NOT SQOOP) fetches all the rows at one shot and tries to load everything into memory. This causes memory spill out and throws error. To overcome this you need to tell RDBMS database to return the data in batches. The following parameters “?dontTrackOpenResources=true&defaultFetchSize=10000&useCursorFetch=true” following the jdbc connection string tells database to fetch 10000 rows per batch.

解决方案

1、指定mappers的数量（数量最好不要超过节点的个数）  
sqoop job --exec gp1813\_user -- --num-mappers 8;  
  
2、调整jvm的内存，缺点:  
 -Dmapreduce.map.memory.mb=6000 \  
 -Dmapreduce.map.java.opts=-Xmx1600m \  
 -Dmapreduce.task.io.sort.mb=4800 \  
  
3、设置mysql的读取数据的方式，不要一次性将所有数据都fetch到内存  
## 在  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods  
## 之后加上  
?dontTrackOpenResources=true&defaultFetchSize=10000&useCursorFetch=true  
  
## 前两种方式，数据量大小不一，只能针对单个确定数据量的导入任务  
## 不能完全解决所有场景  
## 方式3，能很好解决此类内存不足的问题

解决内存不足完整sqoop设置

## 先删除job  
sqoop job --delete qfbap\_ods\_us\_order  
  
## 重新创建job  
sqoop job \  
--create qfbap\_ods\_us\_order -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods?dontTrackOpenResources=true\&defaultFetchSize=10000\&useCursorFetch=true \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--table us\_order \  
--target-dir /hive/tempdb/ods\_us\_order/ \  
--fields-terminated-by '\001' \  
--check-column order\_id \  
--incremental append \  
--last-value 0 \  
;

## 5.4 项目ETL完整代码

### 5.4.1 数据仓库中的表

#### 5.4.1.1 事物事实表

事实表是数据仓库结构中的中央表，它包含联系事实与维度表的数字度量值和键。事实数据表包含描述业务（例如产品销售）内特定事件的数据。

事务事实表记录的事务层面的事实，保存的是最原子的数据，也称“原子事实表”。事务事实表中的数据在事务事件发生后产生，数据的粒度通常是每个事务一条记录。一旦事务被提交，事实表数据被插入，数据就不再进行更改，其更新方式为增量更新。

事务事实表的日期维度记录的是事务发生的日期，它记录的事实是事务活动的内容。用户可以通过事务事实表对事务行为进行特别详细的分析。

通过事务事实表，还可以建立聚集事实表，为用户提供高性能的分析。

#### 5.4.1.2 维度表

维度表是维度属性的集合。是分析问题的一个窗口。是人们观察数据的特定角度，是考虑问题时的一类属性，属性的集合构成一个维。数据库结构中的星型结构，该结构在位于结构中心的单个事实数据表中维护数据，其它维度数据存储在维度表中。每个维度表与事实数据表直接相关，且通常通过一个键联接到事实数据表中。星型架构是数据仓库比较流向的一种架构。

维度表的冗余很大，主要是因为维度一般不大(相对于事实表来说的)，而维度表的冗余可以使事实表节省很多空间。

1)、事实表就是你要关注的内容；  
2)、维度表就是你观察该事务的角度，是从哪个角度去观察这个内容的。  
例如，某地区商品的销量，是从地区这个角度观察商品销量的。事实表就是销量表，维度表就是地区表

#### 5.4.1.3 全量表

每天的所有的最新状态的数据。

（1）全量表，有无变化，都要报  
（2）每次上报的数据都是所有的数据（变化的 + 没有变化的）

#### 5.4.1.4 增量表

新增数据，增量数据是上次导出之后的新数据。  
（1）记录每次增加的量，而不是总量；  
（2）流量是指在一定时间内的增量；  
（3）流量一般设计成增量表（日报-常用、月报）；  
（4）流量和存量的区别：流量是增量；存量是总量；  
（5）增量表，只报变化量，无变化不用报

#### 5.4.1.5 拉链表

拉链表是针对数据仓库设计中表存储数据的方式而定义的，顾名思义，所谓拉链，就是记录历史。记录一个事物从开始，一直到当前状态的所有变化的信息。

拉链表的使用场景

在数据仓库的数据模型设计过程中，经常会遇到下面这种表的设计：

有一些表的数据量很大，比如一张用户表，大约10亿条记录，50个字段，这种表，即使使用ORC压缩，单张表的存储也会超过100G，在HDFS使用双备份或者三备份的话就更大一些。   
表中的部分字段会被update更新操作，如用户联系方式，产品的描述信息，订单的状态等等。   
需要查看某一个时间点或者时间段的历史快照信息，比如，查看某一个订单在历史某一个时间点的状态。   
表中的记录变化的比例和频率不是很大，比如，总共有10亿的用户，每天新增和发生变化的有200万左右，变化的比例占的很小。   
那么对于这种表我该如何设计呢？下面有几种方案可选：

1. 方案一：每天只留最新的一份，比如我们每天用Sqoop抽取最新的一份全量数据到Hive中。
2. 方案二：每天保留一份全量的切片数据。
3. 方案三：使用拉链表。

为什么使用拉链表

现在我们对前面提到的三种进行逐个的分析。

* 方案一

这种方案就不用多说了，实现起来很简单，每天drop掉前一天的数据，重新抽一份最新的。

优点很明显，节省空间，一些普通的使用也很方便，不用在选择表的时候加一个时间分区什么的。

缺点同样明显，没有历史数据，先翻翻旧账只能通过其它方式，比如从流水表里面抽。

* 方案二

每天一份全量的切片是一种比较稳妥的方案，而且历史数据也在。缺点就是存储空间占用量太大太大了，如果对这边表每天都保留一份全量，那么每次全量中会保存很多不变的信息，对存储是极大的浪费，这点我感触还是很深的……当然我们也可以做一些取舍，比如只保留近一个月的数据？但是，需求是无耻的，数据的生命周期不是我们能完全左右的。

* 拉链表

拉链表在使用上基本兼顾了我们的需求。首先它在空间上做了一个取舍，虽说不像方案一那样占用量那么小，但是它每日的增量可能只有方案二的千分之一甚至是万分之一。其实它能满足方案二所能满足的需求，既能获取最新的数据，也能添加筛选条件也获取历史的数据。

#### 5.4.1.6 流水表

对于表的每一个修改都会记录，可以用于反映实际记录的变更

区别于拉链表：

1. 拉链表通常是对账户信息的历史变动进行处理保留的结果，流水表是每天的交易形成的历史；
2. 流水表用于统计业务相关情况，拉链表用于统计账户及客户的情况

#### 5.4.1.7 快照表

1.周期快照表

周期快照事实表记录的是重复的可预测到的时间间隔的事实

周期快照是在一个给定的时间对事实表进行一段时期的总计。例如，一个月销售订单周期快照是每个月底时总的销售订单金额。

周期快照事实表的粒度是每个时间段一条记录，通常比事务事实表的粒度要粗，是在事务事实表之上建立的聚集表。周期快照事实表的维度个数比事务事实表要少，但是记录的事实要比事务事实表多。

周期快照事实表的日期维度通常是记录时间段的终止日，记录的事实是这个时间段内一些聚集事实值。事实表的数据一旦插入即不能更改，其更新方式为增量更新。

2.累计快照表

累计快照适用于较短周期，有着明确的开始和结束状态的过程，如一个订单执行的过程，并记录过程中每个步骤的执行时间，使分析人员对执行的过程有整体的把握。周期快照事实表记录上每个步骤的执行时间是逐步建立的，随着执行的过程逐步更新的事实表中

例如，该累加快照跟踪五个销售订单的里程碑：下订单、分配库房、出库、配送和收货。这五个里程碑的日期及其各自的数量来自源数据库的销售订单表。一个订单完整的生命周期由五行描述：下订单的时间一行，订单商品被分配到库房的时间一行，产品出库的时间一行，订单配送的时间一行，订单客户收货的时间一行。每个里程碑各有一个状态：N为新订单，A为已分配库房，P为已出库，S为已配送，R为已收货。

+--------------+---------------+------------------+-----------------+--------------+-----------------+  
| order*number | order*date*sk | allocate*date*sk | packing*date*sk | ship*date*sk | receive*date\_sk |  
+--------------+---------------+------------------+-----------------+--------------+-----------------+  
| 50 | 5543 | 5544 | 5544 | 5545 | 5545 |  
| 51 | 5543 | 5544 | 5545 | NULL | NULL |  
+--------------+---------------+------------------+-----------------+--------------+-----------------+  
 注意 第一个订单号为50的订单，具有了全部日期代理键，这意味着订单已完成（客户已经收货）。第二个订单已经出库，但是还没有配送。

### 5.4.2 缓慢变化数据

如果数据量小并且容易处理，一般采用全量抽取即可。这种方式适合基础编码类的数据源，最典型的是我们通常所说的数据字典类的码表，在数据仓库中就是维表。如果数据量很大，就只能抽取变化的源数据，即最后一次抽取以来发生了变化的数据。这种数据抽取模式称为变化数据捕获，简称CDC（change data caputer）。

CDC大体分为两种侵入式和非侵入式。侵入式是指CDC操作会给源系统带来性能影响。只要CDC对源库进行了SQL语句操作，就可以认为是侵入式的。常用的CDC方法也有以下四种：

#### 5.4.2.1 基于时间戳的CDC

时间戳：要求源数据里有插入时间和更新时间列，取新增数据根据插入时间，取修改数据根据更新时间

自增列：根据自增列取新增数据

这个方法最常用也最简单，简单的sql语句就可以实现

但是缺点也很明显,不能记录删除操作,触发有逻辑删除的功能,无法识别过多次更新,只能拿到最后一次的,一般为批量操作,不具有实时性,而且具有侵入性.

#### 5.4.2.2 基于触发器的CDC

当执行insert,update,delete这些SQL语句是,可以使用数据库的触发器来执行一些动作,比如触发器将变更的数据保存到临时表中,然后从临时表中抽取数据到贴源层,大多数情况下,触发器会降低业务系统的性能,因此这种方式使用的不多.作为替代方案,可以使用源数据库的复制功能,复制一份数据到备库上,在备库上建立触发器,是个很有效的方法,而且没有侵入性.这个方式需要额外的存储空间,看起来是冗余数据,实际上对业务库来讲,实现主库上进行写操作,备库上进行读操作,实现读写分离,是提高性能和高可用一个手段.

#### 5.4.2.3 基于快照的CDC

如果没有时间戳也不允许使用触发器,就要使用快照表了.可以通过比较原表和快照表来获得变化数据。所谓快照就是一次性抽取源数据中的全部数据，把这些数据加载到ODS层，下次需要同步的时候，再从源数据抽取全部的数据，并把这些数据放到ODS层，比较这两个版本的数据，找出变化数据。一般通过SQL语句可以实现，比如有个表有两列：id和name，在临时过渡区有两个快照 snapshot*1，snapshot*2在mysql中可以这么写：

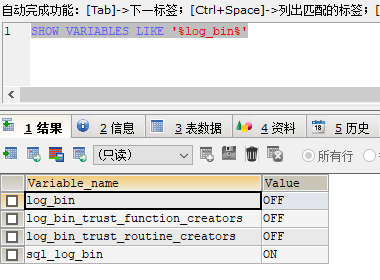
SELECT 'U' AS OprType, t2.id AS id, t2. NAME AS `name`  
FROM snapshot\_1 t1 INNER JOIN snapshot\_2 t2 ON t1. id = t2.1  
WHERE t1.NAME <> a t2.NAME  
  
SELECT 'D' AS OprType, t1,id AS id, t1.NAME AS `name`  
FROM SNAPSHOT\_1 t1 LEFT JOIN snapshot\_2 t22 ON t1.id=t2.id  
WHERE t2. id IS NULL  
UNION ALL  
SELECT 'I' AS OprType, t2.Id AS id, L2. NAME AS `name`  
FROM snapshot\_2 t2 LEFT JOIN snapshot\_1 t1 ON t2.id =t  
WHERE t1 id IS NULL;

就可以获取到数据，这种方式需要临时过渡区来保存快照，浪费存储空间，另外数据量很大时，这种查询会有严重的性能问题。

#### 5.4.2.4 基于日志的CDC

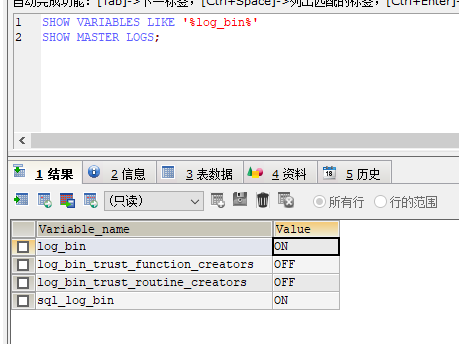
最复杂和最没有入侵的CDC方式就是基于日志的方式。数据库会把每个插入、更新和删除操作记录在日志里。只要在数据库中开启事务日志，再将日志读取出来，就可以还原这部分变化数据了。例如在mysql中，启用log\_bin日志，并使用mysqlbinlog工具读取出来

首先查看一下是否开启了二进制日志：SHOW VARIABLES LIKE '%log\_bin%'



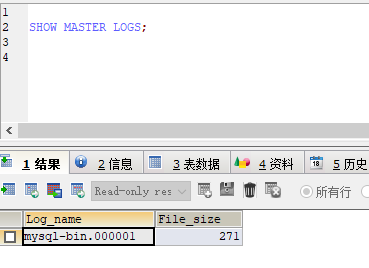
off没有开启，开启步骤如下

1 找到mysql的配置文件，my.cnf一般在/etc，在[mysqld]下面添加log-bin= /var/log/mysql-bin，重启mysql服务



on变为开启

插入一条数据  
  
INSERT INTO dim\_date VALUES (NULL,'2050-01-02',1,'January',1,2050)  
  
查看当前日志  
  
show master status; 或者 SHOW MASTER LOGS;



查看日志，在shell下执行

mysqlbinlog --start-position=0 mysql-bin.000001 | mysql -uroot -p123456  
  
把日志文件输出到文本文件中  
  
mysqlbinlog –start-date=’2018-11-25’ –end-date=’2018-11-26’ mysql-bin.000001 > /tmp/000001.txt

执行文本文件的sql语句就可以了。基于日志的CDC也有缺陷，即只能用来处理一种特定的数据库。

### 5.4.3 用户拉链表实现

拉链表是针对数据仓库设计中表存储数据的方式而定义的，顾名思义，所谓拉链，就是记录历史。记录一个事物从开始，一直到当前状态的所有变化的信息。

我们先看一个示例，这就是一张拉链表，存储的是用户的最基本信息以及每条记录的生命周期。我们可以使用这张表拿到最新的当天的最新数据以及之前的历史数据。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 注册日期 | 用户编号 | 手机号码 | t*start*date | t*end*date |
| 2020-01-01 | 001 | 111111 | 2020-01-01 | 9999-12-31 |
| 2020-01-01 | 002 | 222222 | 2020-01-01 | 2020-01-01 |
| 2020-01-01 | 002 | 233333 | 2020-01-02 | 9999-12-31 |
| 2020-01-01 | 003 | 333333 | 2020-01-01 | 9999-12-31 |
| 2020-01-01 | 004 | 444444 | 2020-01-01 | 2020-01-01 |
| 2020-01-01 | 004 | 432432 | 2020-01-02 | 2020-01-02 |
| 2020-01-01 | 004 | 432432 | 2020-01-03 | 9999-12-31 |
| 2020-01-02 | 005 | 555555 | 2020-01-02 | 2020-01-02 |
| 2020-01-02 | 005 | 115115 | 2020-01-03 | 9999-12-31 |
| 2020-01-03 | 006 | 666666 | 2020-01-03 | 9999-12-31 |

**说明**

* t*start*date表示该条记录的生命周期开始时间，t*end*date表示该条记录的生命周期结束时间。
* t*end*date = ‘9999-12-31’表示该条记录目前处于有效状态。
* 如果查询当前所有有效的记录，则select \* from user where t*end*date = ‘9999-12-31’。
* 如果查询2020-01-02的历史快照，则select from user where t*start*date <= ‘2020-01-02’ and t*end*date >= ‘2020-01-02’。（**此处要好好理解，是拉链表比较重要的一块。**）

好，总共有10亿的用户，每天新增和发生变化的有200万左右。单表数据量超过100G，每天增量算2G。

hive的查询时全表扫描，那么问题来了，这个查询性能如何？如何优化？是否可以分区？如何分区？

分区设计：

分析查询场景，主要条件在于2点：

1、数据状态：有效状态、（过期状态）历史数据

2、历史数据查询，基于时间，查询某天的快照，或者某个时间段的快照

所以，分区可以考虑：

状态

时间，按数据过期时间分区

故用户拉链表结构如下：

CREATE TABLE gp1907\_dws.dws\_user\_his (  
 user\_id bigint,  
 user\_name string,  
 start\_date string,  
 end\_date string,  
 dw\_date string  
) partitioned by (state string,dt string)  
;

用于记录用户姓名的历史数据。

下面我们来举个栗子详细看一下用户拉链表。

我们先看一下在Mysql关系型数据库里的user表中信息变化。

在2020-01-01这一天表中的数据是：

|  |  |  |
| --- | --- | --- |
| 注册日期 | 用户编号 | 手机号码 |
| 2020-01-01 | 001 | 111111 |
| 2020-01-01 | 002 | 222222 |
| 2020-01-01 | 003 | 333333 |
| 2020-01-01 | 004 | 444444 |

在2020-01-02这一天表中的数据是， 用户002和004资料进行了修改，005是新增用户：

|  |  |  |  |
| --- | --- | --- | --- |
| 注册日期 | 用户编号 | 手机号码 | 备注 |
| 2020-01-01 | 001 | 111111 |  |
| 2020-01-01 | 002 | 233333 | （由222222变成233333） |
| 2020-01-01 | 003 | 333333 |  |
| 2020-01-01 | 004 | 432432 | （由444444变成432432） |
| 2020-01-02 | 005 | 555555 | （2020-01-02新增） |

在2020-01-03这一天表中的数据是， 用户004和005资料进行了修改，006是新增用户：

|  |  |  |  |
| --- | --- | --- | --- |
| 注册日期 | 用户编号 | 手机号码 | 备注 |
| 2020-01-01 | 001 | 111111 |  |
| 2020-01-01 | 002 | 233333 |  |
| 2020-01-01 | 003 | 333333 |  |
| 2020-01-01 | 004 | 654321 | （由432432变成654321） |
| 2020-01-02 | 005 | 115115 | （由555555变成115115） |
| 2020-01-03 | 006 | 666666 | （2020-01-03新增） |

如果在数据仓库中设计成历史拉链表保存该表，则会有下面这样一张表，这是最新一天（即2020-01-03）的数据：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 注册日期 | 用户编号 | 手机号码 | t*start*date | t*end*date |
| 2020-01-01 | 001 | 111111 | 2020-01-01 | 9999-12-31 |
| 2020-01-01 | 002 | 222222 | 2020-01-01 | 2020-01-01 |
| 2020-01-01 | 002 | 233333 | 2020-01-02 | 9999-12-31 |
| 2020-01-01 | 003 | 333333 | 2020-01-01 | 9999-12-31 |
| 2020-01-01 | 004 | 444444 | 2020-01-01 | 2020-01-01 |
| 2020-01-01 | 004 | 432432 | 2020-01-02 | 2020-01-02 |
| 2020-01-01 | 004 | 654321 | 2020-01-03 | 9999-12-31 |
| 2020-01-02 | 005 | 555555 | 2020-01-02 | 2020-01-02 |
| 2020-01-02 | 005 | 115115 | 2020-01-03 | 9999-12-31 |
| 2020-01-03 | 006 | 666666 | 2020-01-03 | 9999-12-31 |

我们要实现用户的拉链表。在实现它之前，我们需要先确定一下我们有哪些数据源可以用。

1. 我们需要一张ODS层的用户全量表。至少需要用它来初始化。
2. 每日的用户更新表。

**而且我们要确定拉链表的时间粒度，比如说拉链表每天只取一个状态，也就是说如果一天有3个状态变更，我们只取最后一个状态，这种天粒度的表其实已经能解决大部分的问题了。**

如何获取用户更新的数据，回顾5.7.1 缓慢变化数据的4种获取方式

本次采用快照方式处理。每天全量导入用户当前的全量数据

**ods层的user表**

现在我们来看一下我们ods层的用户信息表的结构：

CREATE DATABASE IF NOT EXISTS qfbap\_ods;  
  
CREATE TABLE if not exists qfbap\_ods.ods\_user (  
 user\_id bigint ,  
 user\_name string ,  
 user\_gender tinyint ,  
 user\_birthday string ,  
 user\_age int ,  
 constellation string ,  
 province string ,  
 city string ,  
 city\_level tinyint ,  
 e\_mail string ,  
 op\_mail string ,  
 mobile bigint ,  
 num\_seg\_mobile int ,  
 op\_Mobile string ,  
 register\_time string ,  
 login\_ip string ,  
 login\_source string ,  
 request\_user string ,  
 total\_score decimal(18,2) ,  
 used\_score decimal(18,2) ,  
 is\_blacklist tinyint ,  
 is\_married tinyint ,  
 education string ,  
 monthly\_income decimal(18,2) ,  
 profession string ,  
 create\_date string   
)   
;

dwd层的user表：

数据需求经过ETL加载到数仓中

CREATE DATABASE IF NOT EXISTS qfbap\_dwd;  
  
CREATE TABLE if not exists qfbap\_dwd.dwd\_user (  
 user\_id bigint ,  
 user\_name string ,  
 user\_gender tinyint ,  
 user\_birthday string ,  
 user\_age int ,  
 constellation string ,  
 province string ,  
 city string ,  
 city\_level tinyint ,  
 e\_mail string ,  
 op\_mail string ,  
 mobile bigint ,  
 num\_seg\_mobile int ,  
 op\_Mobile string ,  
 register\_time string ,  
 login\_ip string ,  
 login\_source string ,  
 request\_user string ,  
 total\_score decimal(18,2) ,  
 used\_score decimal(18,2) ,  
 is\_blacklist tinyint ,  
 is\_married tinyint ,  
 education string ,  
 monthly\_income decimal(18,2) ,  
 profession string ,  
 dw\_date string   
)   
;

将数据导入到dwd表：

insert overwrite table qfbap\_dwd.dwd\_user  
select  
user\_id,  
regexp\_replace(user\_name,' ','') user\_name,  
user\_gender,  
user\_birthday,  
year(current\_date()) - year(user\_birthday) - case when month(current\_date()) < month(user\_birthday) or (month(current\_date()) = month(user\_birthday) and day(current\_date()) < day(user\_birthday)) then 1 else 0 end user\_age,  
constellation,  
province,  
city,  
city\_level,  
e\_mail,  
op\_mail,  
mobile,  
num\_seg\_mobile,  
op\_Mobile,  
register\_time,  
login\_ip,  
login\_source,  
request\_user,  
total\_score,  
used\_score,  
is\_blacklist,  
is\_married,  
education,  
monthly\_income,  
profession,  
current\_timestamp()  
from qfap\_ods.ods\_user  
;

**拉链表**

现在我们创建一张拉链表：

CREATE DATABASE IF NOT EXISTS qfbap\_dws;  
  
CREATE TABLE qfbap\_dws.dws\_user\_his (  
 user\_id bigint,  
 user\_name string,  
 start\_date string,  
 end\_date string,  
 dw\_date string  
) partitioned by (state string,dt string)  
;

现在先初始化数据：

将数据初始化到active状态，到期时间为：9999-12-31。

开始时间为用户注册时间。

insert overwrite table qfbap\_dws.dws\_user\_his partition(state='active',dt='9999-12-31')  
select  
user\_id,  
user\_name,  
from\_unixtime(unix\_timestamp(register\_time,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') start\_date,  
'9999-12-31' end\_date,  
current\_timestamp() dw\_date  
from qfbap\_dwd.dwd\_user  
;

当然，初始化是一次性操作，此过程也可以不使用dwd，直接从ods加载数据：

insert overwrite table qfbap\_dws.dws\_user\_his partition(state='active',dt='9999-12-31')  
select  
user\_id,  
regexp\_replace(user\_name,' ','') user\_name,  
from\_unixtime(unix\_timestamp(register\_time,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') start\_date,  
'9999-12-31' end\_date,  
current\_timestamp() dw\_date  
from qfbap\_ods.ods\_user  
;

然后实现每天修改数据的更新：

修改数据的更新分为两部分

1、改变前的数据需要插入到历史数据（状态设置为过期expired，过期时间为更新时间的前一天）

2、改变后的数据需要更新到当前数据（状态设置为有效active）

先在业务数据库中修改user表信息，在执行导入user表数据的sqoop的job，执行导入数据到dwd\_user的语句

后执行以下语句

1、插入历史数据

insert overwrite table qfbap\_dws.dws\_user\_his partition(state = 'expired',dt='2020-01-01')  
-- 本次被删除的数据  
select   
his.user\_id,  
his.user\_name,  
his.start\_date,  
'2020-01-01' end\_date,  
current\_timestamp() dw\_date  
from qfbap\_dws.dws\_user\_his his  
left join qfbap\_dwd.dwd\_user us on his.state = 'active' and his.dt = '9999-12-31' and us.user\_id = his.user\_id  
where us.user\_id is null  
union all  
-- 本次修改的数据，将修改前的数据更新到历史分区  
select   
his.user\_id,  
his.user\_name,  
his.start\_date,  
'2020-01-05' end\_date,  
current\_timestamp() dw\_date  
from qfbap\_dws.dws\_user\_his his  
join qfbap\_dwd.dwd\_user us on his.state = 'active' and his.dt = '9999-12-31' and us.user\_id = his.user\_id  
where us.user\_name <> his.user\_name  
;  
  
-- 以上分步骤语句可合并：  
insert overwrite table qfbap\_dws.dws\_user\_his partition(state = 'expired',dt='2020-01-01')  
select   
his.user\_id,  
his.user\_name,  
his.start\_date,  
'2020-01-01' end\_date,  
current\_timestamp() dw\_date  
from qfbap\_dws.dws\_user\_his his  
left join qfbap\_dwd.dwd\_user us on his.state = 'active' and his.dt = '9999-12-31' and us.user\_id = his.user\_id  
where us.user\_name <> his.user\_name or us.user\_id is null  
;

2、更新当前数据：

insert overwrite table qfbap\_dws.dws\_user\_his partition(state = 'active',dt='9999-12-31')  
-- 本次被新增或者修改的数据  
select   
us.user\_id,  
us\_user\_name,  
'2020-01-02' start\_date,  
'9999-12-31' end\_date,  
current\_timestamp() dw\_date  
from qfbap\_dwd.dwd\_user us  
left join qfbap\_dws.dws\_user\_his his  
on his.state = 'active' and his.dt = '9999-12-31' and his.user\_id = us.user\_id  
where us.user\_name <> his.user\_name or his.user\_id is null  
union all  
-- 未修改的数据，需要原数据写回his表  
select   
his.user\_id,  
his.user\_name,  
his.start\_date,  
his.end\_date,  
his.dw\_date  
from qfbap\_dwd.dwd\_user us  
join qfbap\_dws.dws\_user\_his his  
on his.state = 'active' and his.dt = '9999-12-31' and his.user\_id = us.user\_id  
where us.user\_name = his.user\_name  
;  
  
-- 以上语句合并：  
insert overwrite table qfbap\_dws.dws\_user\_his partition(state = 'active',dt='9999-12-31')  
select   
us.user\_id,  
us.user\_name,  
case when us.user\_name <> his.user\_name or his.user\_id is null then '2020-01-02' else his.start\_date end start\_date,  
'9999-12-31' end\_date,  
case when us.user\_name <> his.user\_name or his.user\_id is null then cast(current\_timestamp() as string) else dw\_date end dw\_date  
from qfbap\_dwd.dwd\_user us  
left join qfbap\_dws.dws\_user\_his his  
on his.state = 'active' and his.dt = '9999-12-31' and his.user\_id = us.user\_id  
;

### 5.4.4 项目ETL代码

建表语句

CREATE DATABASE IF NOT EXISTS qfbap\_ods;  
  
-- 用户相关表  
CREATE TABLE IF NOT EXISTS qfbap\_ods.ods\_user (  
 user\_id bigint ,  
 user\_name string ,  
 user\_gender tinyint ,  
 user\_birthday string ,  
 user\_age int ,  
 constellation string ,  
 province string ,  
 city string ,  
 city\_level tinyint ,  
 e\_mail string ,  
 op\_mail string ,  
 mobile bigint ,  
 num\_seg\_mobile int ,  
 op\_Mobile string ,  
 register\_time string ,  
 login\_ip string ,  
 login\_source string ,  
 request\_user string ,  
 total\_score decimal(18,2) ,  
 used\_score decimal(18,2) ,  
 is\_blacklist tinyint ,  
 is\_married tinyint ,  
 education string ,  
 monthly\_income decimal(18,2) ,  
 profession string ,  
 create\_date string   
)   
;  
  
CREATE TABLE IF NOT EXISTS qfbap\_ods.ods\_user\_addr (  
 user\_id bigint ,  
 order\_addr string ,  
 user\_order\_flag tinyint ,  
 addr\_id bigint ,  
 arear\_id int   
)  
;  
  
CREATE TABLE IF NOT EXISTS qfbap\_ods.ods\_user\_extend (  
 user\_id bigint ,  
 user\_gender bigint ,  
 is\_pregnant\_woman tinyint ,  
 is\_have\_children tinyint ,  
 is\_have\_car tinyint ,  
 phone\_brand string ,  
 phone\_brand\_level string ,  
 phone\_cnt int ,  
 change\_phone\_cnt int ,  
 is\_maja tinyint ,  
 majia\_account\_cnt int ,  
 loyal\_model string ,  
 shopping\_type\_model string ,  
 weight int ,  
 height int   
)   
;  
  
-- 分区表，业务数据  
CREATE TABLE IF NOT EXISTS qfbap\_ods.ods\_biz\_trade (  
 trade\_id bigint ,  
 order\_id bigint ,  
 user\_id bigint ,  
 amount decimal(18,2) ,  
 trade\_type tinyint ,  
 trade\_time string   
)   
partitioned by (dt string)  
;  
  
CREATE TABLE IF NOT EXISTS qfbap\_ods.ods\_order\_delivery (  
 order\_id bigint ,  
 order\_no string ,  
 consignee string ,  
 area\_id bigint ,  
 area\_name string ,  
 address string ,  
 mobile bigint ,  
 phone string ,  
 coupon\_id bigint ,  
 coupon\_money decimal(18,2) ,  
 carriage\_money decimal(18,2) ,  
 create\_time string ,  
 update\_time string ,  
 addr\_id bigint   
)   
partitioned by (dt string)  
;  
  
CREATE TABLE IF NOT EXISTS qfbap\_ods.ods\_order\_item (  
 user\_id bigint ,  
 order\_id bigint ,  
 order\_no string ,  
 goods\_id bigint ,  
 goods\_no string ,  
 goods\_name string ,  
 goods\_amount int ,  
 shop\_id bigint ,  
 shop\_name string ,  
 curr\_price decimal(18,2) ,  
 market\_price decimal(18,2) ,  
 discount decimal(18,2) ,  
 cost\_price decimal(18,2) ,  
 first\_cart bigint ,  
 first\_cart\_name string ,  
 second\_cart bigint ,  
 second\_cart\_name string ,  
 third\_cart bigint ,  
 third\_cart\_name string ,  
 goods\_desc string   
)   
partitioned by (dt string)  
;  
  
CREATE TABLE IF NOT EXISTS qfbap\_ods.ods\_us\_order (  
 order\_id bigint ,  
 order\_no string ,  
 order\_date string ,  
 user\_id bigint ,  
 user\_name string ,  
 order\_money decimal(18,2) ,  
 order\_type int ,  
 order\_status int ,  
 pay\_status int ,  
 pay\_type int ,  
 order\_source string ,  
 update\_time string   
)  
partitioned by (dt string)  
;  
  
CREATE TABLE IF NOT EXISTS qfbap\_ods.ods\_user\_app\_click\_log (  
 log\_id bigint ,  
 user\_id bigint ,  
 imei string ,  
 log\_time string ,  
 visit\_os string ,  
 os\_version string ,  
 app\_name string ,  
 app\_version string ,  
 device\_token string ,  
 visit\_ip string ,  
 province string ,  
 city string   
)   
partitioned by (dt string)  
;  
  
CREATE TABLE IF NOT EXISTS qfbap\_ods.ods\_user\_pc\_click\_log (  
 log\_id bigint ,  
 user\_id bigint ,  
 session\_id string ,  
 cookie\_id string ,  
 visit\_time string ,  
 visit\_url string ,  
 visit\_os string ,  
 browser\_name string ,  
 visit\_ip string ,  
 province string ,  
 city string ,  
 page\_id int ,  
 goods\_id bigint ,  
 shop\_id bigint   
)  
partitioned by (dt string)  
;

全量导入

#!/bin/bash  
###################################  
##1、该脚本是创建sqoop的job脚本，原则上只需要执行一次即可，以后不需要执行该脚本，只需要执行sqoop job -exec jobname。  
##2、该文件中的全部为非增量表的sqoop的job语句。  
##3、如果需要重新运行，则运行方式:load\_data\_full\_sqoop.sh  
################################  
#1. full load data of bap\_ods\_user, run every day.  
# 用户相关信息导入  
sqoop job \  
--create qfbap\_ods\_user -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods?dontTrackOpenResources=true\&defaultFetchSize=10000\&useCursorFetch=true \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--table user \  
--delete-target-dir \  
--target-dir /hive/qfbap\_ods.db/ods\_user/ \  
--fields-terminated-by '\001' \  
;  
  
sqoop job \  
--create qfbap\_ods\_user\_extend -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods?dontTrackOpenResources=true\&defaultFetchSize=10000\&useCursorFetch=true \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--table user\_extend \  
--delete-target-dir \  
--target-dir /hive/qfbap\_ods.db/ods\_user\_extend/ \  
--fields-terminated-by '\001' \  
;  
  
sqoop job \  
--create qfbap\_ods\_user\_addr -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods?dontTrackOpenResources=true\&defaultFetchSize=10000\&useCursorFetch=true \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--table user\_addr \  
--delete-target-dir \  
--target-dir /hive/qfbap\_ods.db/ods\_user\_addr/ \  
--fields-terminated-by '\001' \  
;

增量导入

#!/bin/bash  
###################################  
##1、该脚本是创建sqoop的job脚本，原则上只需要执行一次即可，以后不需要执行该脚本，只需要执行sqoop job -exec jobname。  
##2、该文件中的全部为增量表的sqoop的job语句。  
##3、如果需要重新运行，则运行方式:load\_data\_incr\_sqoop.sh  
################################  
#增量将数据导入到hive中(注：业务系统中表有自增的id的就用id来增量导出；否则都用create\_date字段来导出)  
#1、增量导出ods\_order：  
sqoop job \  
--create qfbap\_ods\_us\_order -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods?dontTrackOpenResources=true\&defaultFetchSize=10000\&useCursorFetch=true \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--table us\_order \  
--target-dir /hive/tempdb/ods\_us\_order/ \  
--fields-terminated-by '\001' \  
--check-column order\_id \  
--incremental append \  
--last-value 0 \  
;  
  
#2、增量导入order\_item  
sqoop job \  
--create qfbap\_ods\_order\_item -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods?dontTrackOpenResources=true\&defaultFetchSize=10000\&useCursorFetch=true \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--table order\_item \  
--target-dir /hive/tempdb/ods\_order\_item/ \  
--fields-terminated-by '\001' \  
--check-column order\_id \  
--incremental append \  
--last-value 0 \  
;  
  
#3、增量导入biz\_trade  
sqoop job \  
--create qfbap\_ods\_biz\_trade -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods?dontTrackOpenResources=true\&defaultFetchSize=10000\&useCursorFetch=true \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--table biz\_trade \  
--target-dir /hive/tempdb/ods\_biz\_trade/ \  
--fields-terminated-by '\001' \  
--check-column order\_id \  
--incremental append \  
--last-value 0 \  
;  
  
#4、增量导入order\_delivery  
sqoop job \  
--create qfbap\_ods\_order\_delivery -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods?dontTrackOpenResources=true\&defaultFetchSize=10000\&useCursorFetch=true \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--table order\_delivery \  
--target-dir /hive/tempdb/ods\_order\_delivery/ \  
--fields-terminated-by '\001' \  
--check-column order\_id \  
--incremental append \  
--last-value 0 \  
;  
  
#5、增量导入app\_click  
sqoop job \  
--create qfbap\_ods\_user\_app\_click\_log -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods?dontTrackOpenResources=true\&defaultFetchSize=10000\&useCursorFetch=true \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--table user\_app\_click\_log \  
--target-dir /hive/tempdb/ods\_user\_app\_click\_log/ \  
--fields-terminated-by '\001' \  
--check-column log\_id \  
--incremental append \  
--last-value 0 \  
;  
  
#6、增量导入pc\_click  
sqoop job \  
--create qfbap\_ods\_user\_pc\_click\_log -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods?dontTrackOpenResources=true\&defaultFetchSize=10000\&useCursorFetch=true \  
--driver com.mysql.jdbc.Driver \  
--username root \  
--password-file /sqoop/pwd/sqoopPWD.pwd \  
--table user\_pc\_click\_log \  
--target-dir /hive/tempdb/ods\_user\_pc\_click\_log/ \  
--fields-terminated-by '\001' \  
--check-column log\_id \  
--incremental append \  
--last-value 0 \  
;

执行job脚本

全量导入job执行

#!/bin/bash  
############################  
##1、执行该脚本，需要保证sqoop的job已经被创建好(语句在load\_data\_full\_sqoop.sh中,如果没有job，需创建).  
sqoop job --list  
#sqoop job --delete jobname  
#sqoop job -exec jobname  
##2、每天凌晨2点定时执行如下的sqoop的job，全为全量表的数据加载。  
##3、如果有问题，需要重新跑job。  
##4、运行方式。./load\_data\_full.sh  
################################  
#run sqoop job  
sqoop job --exec qfbap\_ods\_user;  
sqoop job --exec qfbap\_ods\_user\_extend;  
sqoop job --exec qfbap\_ods\_user\_addr;  
sqoop job --exec qfbap\_code\_city;  
sqoop job --exec qfbap\_os\_name;  
sqoop job --exec qfbap\_browser\_name;

#!/bin/bash  
############################  
##1、执行该脚本，需要保证sqoop的job已经被创建好(语句在load\_data\_incr\_sqoop.sh中,如果没有job，需创建).  
sqoop job --list  
#sqoop job --delete jobname  
#sqoop job -exec jobname  
##2、每天凌晨2点定时执行如下的sqoop的job，全为增量表的数据加载。  
##3、如果有问题，需要重新跑job。  
##4、运行方式。./load\_data\_incr.sh  
################################  
#run sqoop job  
#run with args.eg: ./load\_data\_incr.sh 2018-12-25  
args=$1  
dt=  
if [ ${#args} == 0 ]  
then  
dt=`date -d "1 days ago" +"%Y-%m-%d"`  
else  
dt=$1  
fi  
  
sqoop job --exec qfbap\_ods\_us\_order;  
sqoop job --exec qfbap\_ods\_order\_item;  
sqoop job --exec qfbap\_ods\_biz\_trade;  
sqoop job --exec qfbap\_ods\_order\_delivery;  
sqoop job --exec qfbap\_ods\_user\_app\_click\_log;  
sqoop job --exec qfbap\_ods\_user\_pc\_click\_log;  
  
hive -e "load data inpath '/hive/tempdb/ods\_us\_order/\*' into table qfbap\_ods.ods\_us\_order partition(dt='${dt}')"  
hive -e "load data inpath '/hive/tempdb/ods\_order\_item/\*' into table qfbap\_ods.ods\_order\_item partition(dt='${dt}')"  
hive -e "load data inpath '/hive/tempdb/ods\_biz\_trade/\*' into table qfbap\_ods.ods\_biz\_trade partition(dt='${dt}')"  
hive -e "load data inpath '/hive/tempdb/ods\_order\_delivery/\*' into table qfbap\_ods.ods\_order\_delivery partition(dt='${dt}')"  
hive -e "load data inpath '/hive/tempdb/ods\_user\_app\_click\_log/\*' into table qfbap\_ods.ods\_user\_app\_click\_log partition(dt='${dt}')"  
hive -e "load data inpath '/hive/tempdb/ods\_user\_pc\_click\_log/\*' into table qfbap\_ods.ods\_user\_pc\_click\_log partition(dt='${dt}')"

## 5.5 开发数据仓库的分析应用

### 5.5.1 用户基本宽表

前面ETL过程时，已经将数据导入到了ods层，所以，这里直接接着写分析语句

1、创建dwd层

CREATE TABLE IF NOT EXISTS qfbap\_dwd.dwd\_user (  
 user\_id bigint ,  
 user\_name string ,  
 user\_gender tinyint ,  
 user\_birthday string ,  
 user\_age int ,  
 constellation string ,  
 province string ,  
 city string ,  
 city\_level tinyint ,  
 e\_mail string ,  
 op\_mail string ,  
 mobile bigint ,  
 num\_seg\_mobile int ,  
 op\_Mobile string ,  
 register\_time string ,  
 login\_ip string ,  
 login\_source string ,  
 request\_user string ,  
 total\_score decimal(18,2) ,  
 used\_score decimal(18,2) ,  
 is\_blacklist tinyint ,  
 is\_married tinyint ,  
 education string ,  
 monthly\_income decimal(18,2) ,  
 profession string ,  
 dw\_date string   
)   
;  
  
CREATE TABLE IF NOT EXISTS qfbap\_dwd.dwd\_user\_extend (  
 user\_id bigint ,  
 user\_gender bigint ,  
 is\_pregnant\_woman tinyint ,  
 is\_have\_children tinyint ,  
 is\_have\_car tinyint ,  
 phone\_brand string ,  
 phone\_brand\_level string ,  
 phone\_cnt int ,  
 change\_phone\_cnt int ,  
 is\_maja tinyint ,  
 majia\_account\_cnt int ,  
 loyal\_model string ,  
 shopping\_type\_model string ,  
 weight int ,  
 height int ,  
 dw\_date string  
)  
;

2、加载数据

-- 加载数据到dwd\_user表  
insert overwrite table qfbp\_dwd.dwd\_user  
select  
user\_id,  
regexp\_replace(user\_name,' ','') user\_name,  
user\_gender,  
user\_birthday,  
year(current\_date()) - year(user\_birthday) - case when month(current\_date()) < month(user\_birthday) or (month(current\_date()) = month(user\_birthday) and day(current\_date()) < day(user\_birthday)) then 1 else 0 end user\_age,  
constellation,  
province,  
city,  
city\_level,  
e\_mail,  
op\_mail,  
mobile,  
num\_seg\_mobile,  
op\_Mobile,  
register\_time,  
login\_ip,  
login\_source,  
request\_user,  
total\_score,  
used\_score,  
is\_blacklist,  
is\_married,  
education,  
monthly\_income,  
profession,  
current\_timestamp()  
from qfbap\_ods.ods\_user  
;  
  
-- 加载user\_extend的数据到dwd：  
insert overwrite table qfbap\_dwd.dwd\_user\_extend  
select  
user\_id,  
user\_gender,  
is\_pregnant\_woman,  
is\_have\_children,  
is\_have\_car,  
phone\_brand,  
phone\_brand\_level,  
phone\_cnt,  
change\_phone\_cnt,  
is\_maja,  
majia\_account\_cnt,  
loyal\_model,  
shopping\_type\_model,  
weight,  
height,  
current\_timestamp() dw\_date  
from qfbap\_ods.ods\_user\_extend  
;

后执行分析语句将数据导入到dm表

-- 将数据加载到dm层  
-- 先创建dm表  
CREATE TABLE if not exists qfbap\_dm.dm\_user\_basic(  
user\_id bigint,   
user\_name string,   
user\_gender tinyint,   
user\_birthday string,   
user\_age int,   
constellation string,   
province string,   
city string,   
city\_level tinyint,   
e\_mail string,   
op\_mail string,   
mobile bigint,   
num\_seg\_mobile int,   
op\_mobile string,   
register\_time string,   
login\_ip string,   
login\_source string,   
request\_user string,   
total\_score decimal(18,2),   
used\_score decimal(18,2),   
is\_blacklist tinyint,   
is\_married tinyint,   
education string,   
monthly\_income decimal(18,2),   
profession string,   
is\_pregnant\_woman tinyint,   
is\_have\_children tinyint,   
is\_have\_car tinyint,   
phone\_brand string,   
phone\_brand\_level string,   
phone\_cnt int,   
change\_phone\_cnt int,   
is\_maja tinyint,   
majia\_account\_cnt int,   
loyal\_model string,   
shopping\_type\_model string,   
weight int,   
height int,  
dw\_date string  
)  
;  
  
-- 加载数据到dm\_user表  
insert overwrite table qfbap\_dm.dm\_user\_basic  
select   
u.user\_id,  
u.user\_name,  
u.user\_gender,  
u.user\_birthday,  
u.user\_age,  
u.constellation,  
u.province,  
u.city,  
u.city\_level,  
u.e\_mail,  
u.op\_mail,  
u.mobile,  
u.num\_seg\_mobile,  
u.op\_Mobile,  
u.register\_time,  
u.login\_ip,  
u.login\_source,  
u.request\_user,  
u.total\_score,  
u.used\_score,  
u.is\_blacklist,  
u.is\_married,  
u.education,  
u.monthly\_income,  
u.profession,  
ue.is\_pregnant\_woman,  
ue.is\_have\_children,  
ue.is\_have\_car,  
ue.phone\_brand,  
ue.phone\_brand\_level,  
ue.phone\_cnt,  
ue.change\_phone\_cnt,  
ue.is\_maja,  
ue.majia\_account\_cnt,  
ue.loyal\_model,  
ue.shopping\_type\_model,  
ue.weight,  
ue.height,  
current\_timestamp() dw\_date  
from qfbap\_dwd.dwd\_user u  
left join qfbap\_dwd.dwd\_user\_extend ue on ue.user\_id = u.user\_id   
;

### 5.5.2 用户访问模型表

1、创建dwd表

CREATE TABLE qfbap\_dwd.dwd\_user\_pc\_pv (  
log\_id bigint ,  
user\_id bigint ,  
session\_id string ,  
cookie\_id string ,  
in\_time string,  
out\_time string,  
stay\_time bigint,  
pv bigint,  
visit\_os string ,  
browser\_name string ,  
visit\_ip string ,  
province string ,  
city string ,  
dw\_date string  
)  
partitioned by (dt string)  
;  
  
CREATE external TABLE if not exists qfbap\_dwd.dwd\_user\_app\_pv (  
 log\_id bigint ,  
 user\_id bigint ,  
 imei string ,  
 log\_time string ,  
 log\_hour string,  
 visit\_os string ,  
 os\_version string ,  
 app\_name string ,  
 app\_version string ,  
 device\_token string ,  
 visit\_ip string ,  
 province string ,  
 city string ,  
 dw\_date string  
)   
partitioned by (dt string)  
;

2、加载数据到dwd

insert overwrite table qfbap\_dwd.dwd\_user\_pc\_pv partition(dt='2020-01-05')  
select  
max(log\_id) log\_id,  
user\_id,  
session\_id,  
cookie\_id,  
min(visit\_time) in\_time,  
max(visit\_time) out\_time,  
unix\_timestamp(max(visit\_time)) - unix\_timestamp(min(visit\_time)) stay\_time,  
count(1) pv,  
visit\_os,  
browser\_name,  
visit\_ip,  
province,  
city,  
current\_timestamp() dw\_date  
from qfbap\_ods.ods\_user\_pc\_click\_log   
where dt = '2020-01-05'  
group by   
user\_id,  
session\_id,  
cookie\_id,  
visit\_os,  
browser\_name,  
visit\_ip,  
province,  
city,  
dt  
;  
  
insert overwrite table qfbap\_dwd.dwd\_user\_app\_pv partition (dt = '2020-01-05')  
select  
log\_id,  
user\_id,  
imei,  
log\_time,  
hour(log\_time) log\_hour,  
visit\_os,  
os\_version,  
app\_name,  
app\_version,  
device\_token,  
visit\_ip,  
province,  
city,  
current\_timestamp() dw\_date  
from qfbap\_ods.ods\_user\_app\_click\_log  
where dt = '2020-01-05'  
;

3、创建dws表

create database if not exists qfbap\_dws;  
create table if not exists qfbap\_dws.dws\_user\_visit\_month1  
(  
 user\_id bigint ,  
 type string,  
 content string,  
 cnt bigint,  
 rn bigint,  
 dw\_date string  
)  
partitioned by (dt string)  
stored as orc  
;

4、加载数据到dws

insert overwrite table qfbap\_dws.dws\_user\_visit\_month1 partition(dt = '2020-01-05')  
select   
user\_id,  
type,  
content,  
cnt,  
row\_number() over(distribute by user\_id,type sort by cnt desc) rn,  
current\_timestamp() dw\_date  
from   
(  
select  
user\_id,  
'visit\_ip' as type,  
visit\_ip as content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv   
where dt >= date\_sub('2020-01-05',29) and dt <= '2020-01-05'  
group by   
user\_id,  
visit\_ip  
union all  
select  
user\_id,  
'cookie\_id' as type,  
cookie\_id as content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv   
where dt >= date\_sub('2020-01-05',29) and dt <= '2020-01-05'  
group by   
user\_id,  
cookie\_id  
union all  
select  
user\_id,  
'browser\_name' as type,  
browser\_name as content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv   
where dt >= date\_sub('2020-01-05',29) and dt <= '2020-01-05'  
group by   
user\_id,  
browser\_name  
union all  
select  
user\_id,  
'visit\_os' as type,  
visit\_os as content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv   
where dt >= date\_sub('2020-01-05',29) and dt <= '2020-01-05'  
group by   
user\_id,  
visit\_os  
) pc  
;  
  
-- 另一种写法：  
with m as  
(  
select   
pc.user\_id,  
'visit\_ip' type,  
pc.visit\_ip content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv pc  
where dt >= date\_sub('2020-01-05',29) and dt <= '2020-01-05'  
group by   
pc.user\_id,  
pc.visit\_ip  
union all  
select   
pc.user\_id,  
'cookie\_id' type,  
pc.cookie\_id content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv pc  
where dt >= date\_sub('2020-01-05',29) and dt <= '2020-01-05'  
group by   
pc.user\_id,  
pc.cookie\_id  
union all  
select   
pc.user\_id,  
'browser\_id' type,  
pc.browser\_id content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv pc  
where dt >= date\_sub('2020-01-05',29) and dt <= '2020-01-05'  
group by   
pc.user\_id,  
pc.browser\_id  
union all  
select   
pc.user\_id,  
'visit\_os' type,  
pc.os\_id content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv pc  
where dt >= date\_sub('2020-01-05',29) and dt <= '2020-01-05'  
group by   
pc.user\_id,  
pc.os\_id  
)  
  
insert overwrite table qfbap\_dws.dws\_user\_visit\_month1 partition(dt='2020-01-05')  
select  
m.user\_id,  
m.type,  
m.content,  
m.cnt,  
row\_number() over(distribute by m.user\_id,m.type sort by m.cnt desc) rn,  
current\_timestamp() dw\_date  
from m  
;  
  
  
-- 另一种写法：  
from  
(  
select   
pc.user\_id,  
'visit\_ip' type,  
pc.visit\_ip content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv pc  
where dt >= date\_sub('2020-01-05',29) and dt <= '2020-01-05'  
group by   
pc.user\_id,  
pc.visit\_ip  
union all  
select   
pc.user\_id,  
'cookie\_id' type,  
pc.cookie\_id content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv pc  
where dt >= date\_sub('2020-01-05',29) and dt <= '2020-01-05'  
group by   
pc.user\_id,  
pc.cookie\_id  
union all  
select   
pc.user\_id,  
'browser\_id' type,  
pc.browser\_id content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv pc  
where dt >= date\_sub('2020-01-05',29) and dt <= '2020-01-05'  
group by   
pc.user\_id,  
pc.browser\_id  
union all  
select   
pc.user\_id,  
'visit\_os' type,  
pc.os\_id content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv pc  
where dt >= date\_sub('2020-01-05',29) and dt <= '2020-01-05'  
group by   
pc.user\_id,  
pc.os\_id  
) m  
  
insert overwrite table qfbap\_dws.dws\_user\_visit\_month1 partition(dt='2020-01-05')  
select  
m.user\_id,  
m.type,  
m.content,  
m.cnt,  
row\_number() over(distribute by m.user\_id,m.type sort by m.cnt desc) rn,  
current\_timestamp() dw\_date  
;

5、指标分析与计算

-- 先进行pc端的统计：  
-- 将指标分类  
--最近一次访问时间  
--最近一次访问使用的session  
--最近一次使用的cookie  
--最近一次的pc端的pv量  
--最近一次访问使用的浏览器  
--最近一次访问使用的操作系统  
--第一次pc端访问的日期  
--第一次pc端访问的session  
--第一次pc端访问的cookie  
--第一次访问的pv  
--第一次访问使用的浏览器  
--第一次访问的os   
  
-- first\_value  
-- last\_value  
-- row\_number  
-- max  
  
select   
pc.user\_id,  
max(case when pc.rn\_desc = 1 then pc.in\_time end) as last\_time,  
max(case when pc.rn\_desc = 1 then pc.session\_id end) as last\_session,  
max(case when pc.rn\_desc = 1 then pc.cookie\_id end) as last\_cookie,  
max(case when pc.rn\_desc = 1 then pc.pv end) as last\_pv,  
max(case when pc.rn\_desc = 1 then pc.browser\_name end) as last\_browser,  
max(case when pc.rn\_desc = 1 then pc.visit\_os end) as last\_os,  
max(case when pc.rn\_asc = 1 then pc.in\_time end) as first\_time,  
max(case when pc.rn\_asc = 1 then pc.session\_id end) as first\_session,  
max(case when pc.rn\_asc = 1 then pc.cookie\_id end) as first\_cookie,  
max(case when pc.rn\_asc = 1 then pc.pv end) as first\_pv,  
max(case when pc.rn\_asc = 1 then pc.browser\_name end) as first\_browser,  
max(case when pc.rn\_asc = 1 then pc.visit\_os end) as first\_os  
from  
(  
select   
user\_id,  
session\_id,  
cookie\_id,  
visit\_os,  
browser\_name,  
visit\_ip,  
province,  
city,  
pv,  
in\_time,  
out\_time,  
stay\_time,  
row\_number() over(distribute by user\_id sort by in\_time asc) as rn\_asc,  
row\_number() over(distribute by user\_id sort by in\_time desc) as rn\_desc  
from qfbap\_dwd.dwd\_user\_pc\_pv  
) pc  
group by   
user\_id  
;  
  
--PC连续7天访问次数(跑任务的日期的前7天,以下相同不再赘述)  
--连续15天访问次数  
--连续30天访问次数  
--连续60天访问的次数  
--连续90天访问的次数  
select   
pc.user\_id,  
max(case when pc.rn\_desc = 1 then pc.in\_time end) as last\_time,  
max(case when pc.rn\_desc = 1 then pc.session\_id end) as last\_session,  
max(case when pc.rn\_desc = 1 then pc.cookie\_id end) as last\_cookie,  
max(case when pc.rn\_desc = 1 then pc.pv end) as last\_pv,  
max(case when pc.rn\_desc = 1 then pc.browser\_name end) as last\_browser,  
max(case when pc.rn\_desc = 1 then pc.visit\_os end) as last\_os,  
max(case when pc.rn\_asc = 1 then pc.in\_time end) as first\_time,  
max(case when pc.rn\_asc = 1 then pc.session\_id end) as first\_session,  
max(case when pc.rn\_asc = 1 then pc.cookie\_id end) as first\_cookie,  
max(case when pc.rn\_asc = 1 then pc.pv end) as first\_pv,  
max(case when pc.rn\_asc = 1 then pc.browser\_name end) as first\_browser,  
max(case when pc.rn\_asc = 1 then pc.visit\_os end) as first\_os,  
count(case when pc.dt7 = 1 then pc.session\_id end) as cnt7,  
count(case when pc.dt15 = 1 then pc.session\_id end) as cnt15,  
count(case when pc.dt30 = 1 then pc.session\_id end) as cnt30,  
count(case when pc.dt60 = 1 then pc.session\_id end) as cnt60,  
count(case when pc.dt90 = 1 then pc.session\_id end) as cnt90,  
count(distinct substring(case when pc.dt30 = 1 then pc.in\_time end,1,10)) day30,  
sum(case when pc.dt30 = 1 then pc.pv end) pv30,  
sum(case when pc.dt30 = 1 then pc.pv end) / count(distinct substring(case when pc.dt30 = 1 then pc.in\_time end,1,10)) avgpv30  
from  
(  
select   
user\_id,  
session\_id,  
cookie\_id,  
visit\_os,  
browser\_name,  
visit\_ip,  
province,  
city,  
pv,  
in\_time,  
out\_time,  
stay\_time,  
row\_number() over(distribute by user\_id sort by in\_time asc) as rn\_asc,  
row\_number() over(distribute by user\_id sort by in\_time desc) as rn\_desc,  
case when dt >= date\_sub('2019-11-14',6) then 1 end as dt7,  
case when dt >= date\_sub('2019-11-14',14) then 1 end as dt15,  
case when dt >= date\_sub('2019-11-14',29) then 1 end as dt30,  
case when dt >= date\_sub('2019-11-14',59) then 1 end as dt60,  
case when dt >= date\_sub('2019-11-14',89) then 1 end as dt90  
from qfbap\_dwd.dwd\_user\_pc\_pv  
where dt >= date\_sub('2019-11-14',89) and dt <= '2019-11-14'  
) pc  
group by   
user\_id  
;  
  
--按小时统计pv指标  
select   
pc.user\_id,  
max(case when pc.rn\_desc = 1 then pc.in\_time end) as last\_time,  
max(case when pc.rn\_desc = 1 then pc.session\_id end) as last\_session,  
max(case when pc.rn\_desc = 1 then pc.cookie\_id end) as last\_cookie,  
max(case when pc.rn\_desc = 1 then pc.pv end) as last\_pv,  
max(case when pc.rn\_desc = 1 then pc.browser\_name end) as last\_browser,  
max(case when pc.rn\_desc = 1 then pc.visit\_os end) as last\_os,  
max(case when pc.rn\_asc = 1 then pc.in\_time end) as first\_time,  
max(case when pc.rn\_asc = 1 then pc.session\_id end) as first\_session,  
max(case when pc.rn\_asc = 1 then pc.cookie\_id end) as first\_cookie,  
max(case when pc.rn\_asc = 1 then pc.pv end) as first\_pv,  
max(case when pc.rn\_asc = 1 then pc.browser\_name end) as first\_browser,  
max(case when pc.rn\_asc = 1 then pc.visit\_os end) as first\_os,  
count(case when pc.dt7 = 1 then pc.session\_id end) as cnt7,  
count(case when pc.dt15 = 1 then pc.session\_id end) as cnt15,  
count(case when pc.dt30 = 1 then pc.session\_id end) as cnt30,  
count(case when pc.dt60 = 1 then pc.session\_id end) as cnt60,  
count(case when pc.dt90 = 1 then pc.session\_id end) as cnt90,  
count(distinct substring(case when pc.dt30 = 1 then pc.in\_time end,1,10)) day30,  
sum(case when pc.dt30 = 1 then pc.pv end) pv30,  
sum(case when pc.dt30 = 1 then pc.pv end) / count(distinct substring(case when pc.dt30 = 1 then pc.in\_time end,1,10)) avgpv30,  
sum(case when pc.dt30 = 1 and pc.hr025 = 1 then pc.pv end) pv025,  
sum(case when pc.dt30 = 1 and pc.hr627 = 1 then pc.pv end) pv627,  
sum(case when pc.dt30 = 1 and pc.hr829 = 1 then pc.pv end) pv829,  
sum(case when pc.dt30 = 1 and pc.hr10211 = 1 then pc.pv end) pv10211,  
sum(case when pc.dt30 = 1 and pc.hr12213 = 1 then pc.pv end) pv12213,  
sum(case when pc.dt30 = 1 and pc.hr14215 = 1 then pc.pv end) pv14215,  
sum(case when pc.dt30 = 1 and pc.hr16217 = 1 then pc.pv end) pv16217,  
sum(case when pc.dt30 = 1 and pc.hr18219 = 1 then pc.pv end) pv18219,  
sum(case when pc.dt30 = 1 and pc.hr20221 = 1 then pc.pv end) pv20221,  
sum(case when pc.dt30 = 1 and pc.hr22223 = 1 then pc.pv end) pv22223  
from  
(  
select   
user\_id,  
session\_id,  
cookie\_id,  
visit\_os,  
browser\_name,  
visit\_ip,  
province,  
city,  
pv,  
in\_time,  
out\_time,  
stay\_time,  
row\_number() over(distribute by user\_id sort by in\_time asc) as rn\_asc,  
row\_number() over(distribute by user\_id sort by in\_time desc) as rn\_desc,  
case when dt >= date\_sub('2019-11-14',6) then 1 end as dt7,  
case when dt >= date\_sub('2019-11-14',14) then 1 end as dt15,  
case when dt >= date\_sub('2019-11-14',29) then 1 end as dt30,  
case when dt >= date\_sub('2019-11-14',59) then 1 end as dt60,  
case when dt >= date\_sub('2019-11-14',89) then 1 end as dt90,  
case when hour(in\_time) between 0 and 5 then 1 end as hr025,  
case when hour(in\_time) between 6 and 7 then 1 end as hr627,  
case when hour(in\_time) between 8 and 9 then 1 end as hr829,  
case when hour(in\_time) between 10 and 11 then 1 end as hr10211,  
case when hour(in\_time) between 12 and 13 then 1 end as hr12213,  
case when hour(in\_time) between 14 and 15 then 1 end as hr14215,  
case when hour(in\_time) between 16 and 17 then 1 end as hr16217,  
case when hour(in\_time) between 18 and 19 then 1 end as hr18219,  
case when hour(in\_time) between 20 and 21 then 1 end as hr20221,  
case when hour(in\_time) between 22 and 23 then 1 end as hr22223  
from qfbap\_dwd.dwd\_user\_pc\_pv  
where dt >= date\_sub('2019-11-14',89) and dt <= '2019-11-14'  
) pc  
group by   
user\_id  
;  
  
-- pc端汇总指标计算：  
--近30天访问使用的不同ip数量  
--近30天最常用的ip  
--近30天使用的cookie的数量  
--近30使用最常用的cookie\_id  
--近30pc最常用浏览器  
--近30天使用最常用系统  
select  
mon.user\_id,  
count(case when mon.type = 'visit\_ip' then mon.content end) ipcnt,  
max(case when mon.type = 'visit\_ip' and mon.rn = 1 then mon.content end) topip,  
count(case when mon.type = 'cookie\_id' then mon.content end) ckcnt,  
max(case when mon.type = 'cookie\_id' and mon.rn = 1 then mon.content end) topck,  
max(case when mon.type = 'browser\_name' and mon.rn = 1 then mon.content end) topbr,  
max(case when mon.type = 'visit\_os' and mon.rn = 1 then mon.content end) topos  
from qfbap\_dws.dws\_user\_visit\_month1 mon  
where dt = '2019-11-14'  
group by   
mon.user\_id   
-- limit 10  
;  
  
-- app端指标：  
select  
app.user\_id,  
max(case when app.rn\_desc = 1 then app.log\_time end) last\_apptime,  
max(case when app.rn\_desc = 1 then app.app\_name end) last\_appname,  
max(case when app.rn\_desc = 1 then app.visit\_os end) last\_appos,  
max(case when app.rn\_desc = 1 then app.visit\_ip end) last\_appip,  
max(case when app.rn\_asc = 1 then app.log\_time end) first\_apptime,  
max(case when app.rn\_asc = 1 then app.app\_name end) first\_appname,  
max(case when app.rn\_asc = 1 then app.visit\_os end) first\_appos,  
max(case when app.rn\_desc = 1 then app.visit\_ip end) first\_appip,  
count(case when app.dt7 = 1 then app.user\_id end) as appcnt7,  
count(case when app.dt15 = 1 then app.user\_id end) as appcnt15,  
count(case when app.dt30 = 1 then app.user\_id end) as appcnt30,  
count(case when app.dt60 = 1 then app.user\_id end) as appcnt60,  
count(case when app.dt90 = 1 then app.user\_id end) as appcnt90,  
sum(case when app.dt30 = 1 and app.hr025 = 1 then 1 end) apppv025,  
sum(case when app.dt30 = 1 and app.hr627 = 1 then 1 end) apppv627,  
sum(case when app.dt30 = 1 and app.hr829 = 1 then 1 end) apppv829,  
sum(case when app.dt30 = 1 and app.hr10211 = 1 then 1 end) apppv10211,  
sum(case when app.dt30 = 1 and app.hr12213 = 1 then 1 end) apppv12213,  
sum(case when app.dt30 = 1 and app.hr14215 = 1 then 1 end) apppv14215,  
sum(case when app.dt30 = 1 and app.hr16217 = 1 then 1 end) apppv16217,  
sum(case when app.dt30 = 1 and app.hr18219 = 1 then 1 end) apppv18219,  
sum(case when app.dt30 = 1 and app.hr20221 = 1 then 1 end) apppv20221,  
sum(case when app.dt30 = 1 and app.hr22223 = 1 then 1 end) apppv22223  
from   
(  
select   
user\_id,  
imei,  
log\_time,  
log\_hour,  
visit\_os,  
os\_version,  
app\_name,  
app\_version,  
device\_token,  
visit\_ip,  
province,  
city,  
row\_number() over(distribute by user\_id sort by log\_time asc) rn\_asc,  
row\_number() over(distribute by user\_id sort by log\_time desc) rn\_desc,  
case when dt >= date\_sub('2019-11-14',6) then 1 end dt7,  
case when dt >= date\_sub('2019-11-14',14) then 1 end dt15,  
case when dt >= date\_sub('2019-11-14',29) then 1 end dt30,  
case when dt >= date\_sub('2019-11-14',59) then 1 end dt60,  
case when dt >= date\_sub('2019-11-14',89) then 1 end dt90,  
case when log\_hour between 0 and 5 then 1 end as hr025,  
case when log\_hour between 6 and 7 then 1 end as hr627,  
case when log\_hour between 8 and 9 then 1 end as hr829,  
case when log\_hour between 10 and 11 then 1 end as hr10211,  
case when log\_hour between 12 and 13 then 1 end as hr12213,  
case when log\_hour between 14 and 15 then 1 end as hr14215,  
case when log\_hour between 16 and 17 then 1 end as hr16217,  
case when log\_hour between 18 and 19 then 1 end as hr18219,  
case when log\_hour between 20 and 21 then 1 end as hr20221,  
case when log\_hour between 22 and 23 then 1 end as hr22223  
from qfbap\_dwd.dwd\_user\_app\_pv   
where dt >= date\_sub('2019-11-14',89) and dt <= '2019-11-14'  
) app  
group by   
app.user\_id  
-- limit 10  
;

6、创建dm表

create database if not exists qfbap\_dm;  
CREATE TABLE if not exists qfbap\_dm.dm\_user\_visit(  
user\_id bigint,   
last\_time string,   
last\_session string,   
last\_cookie string,   
last\_pv bigint,   
last\_browser string,   
last\_os string,   
first\_time string,   
first\_session string,   
first\_cookie string,   
first\_pv bigint,   
first\_browser string,   
first\_os string,   
cnt7 bigint,   
cnt15 bigint,   
cnt30 bigint,   
cnt60 bigint,   
cnt90 bigint,   
day30 bigint,   
pv30 bigint,   
avgpv30 double,   
pv025 bigint,   
pv627 bigint,   
pv829 bigint,   
pv10211 bigint,   
pv12213 bigint,   
pv14215 bigint,   
pv16217 bigint,   
pv18219 bigint,   
pv20221 bigint,   
pv22223 bigint,   
ipcnt bigint,   
topip string,   
ckcnt bigint,   
topck string,   
topbr string,   
topos string,   
last\_apptime string,   
last\_appname string,   
last\_appos string,   
last\_appip string,   
first\_apptime string,   
first\_appname string,   
first\_appos string,   
first\_appip string,   
appcnt7 bigint,   
appcnt15 bigint,   
appcnt30 bigint,   
appcnt60 bigint,   
appcnt90 bigint,   
apppv025 bigint,   
apppv627 bigint,   
apppv829 bigint,   
apppv10211 bigint,   
apppv12213 bigint,   
apppv14215 bigint,   
apppv16217 bigint,   
apppv18219 bigint,   
apppv20221 bigint,   
apppv22223 bigint,   
maxip string,   
maxprovince string,   
maxcity string,   
minip string,   
minprovince string,   
mincity string)  
partitioned by (dt string)  
;

7、加载数据到dm表

-- 代码整合  
--1、所有用户  
-- 以用户宽表为基准，left join访问行为表  
--2、对今天的活跃用户  
--抽取活跃用户  
--或者union all 聚合数据  
insert overwrite table qfbap\_dm.dm\_user\_visit partition(dt = '2019-11-14')  
select   
user\_id,   
last\_time,   
last\_session,   
last\_cookie,   
last\_pv,   
last\_browser,   
last\_os,   
first\_time,   
first\_session,   
first\_cookie,   
first\_pv,   
first\_browser,   
first\_os,   
cnt7,   
cnt15,   
cnt30,   
cnt60,   
cnt90,   
day30,   
pv30,   
avgpv30,   
pv025,   
pv627,   
pv829,   
pv10211,   
pv12213,   
pv14215,   
pv16217,   
pv18219,   
pv20221,   
pv22223,   
ipcnt,   
topip,   
ckcnt,   
topck,   
topbr,   
topos,   
last\_apptime,   
last\_appname,   
last\_appos,   
last\_appip,   
first\_apptime,   
first\_appname,   
first\_appos,   
first\_appip,   
appcnt7,   
appcnt15,   
appcnt30,   
appcnt60,   
appcnt90,   
apppv025,   
apppv627,   
apppv829,   
apppv10211,   
apppv12213,   
apppv14215,   
apppv16217,   
apppv18219,   
apppv20221,   
apppv22223,  
case when last\_time > last\_apptime then last\_ip else last\_appip end maxip,  
case when last\_time > last\_apptime then last\_province else last\_appprovince end maxprovince,  
case when last\_time > last\_apptime then last\_city else last\_appcity end maxcity,  
case when first\_time < first\_apptime then first\_ip else first\_appip end minip,  
case when first\_time < first\_apptime then first\_province else first\_appprovince end minprovince,  
case when first\_time < first\_apptime then first\_city else first\_appcity end mincity  
from   
(  
select  
us.user\_id,  
max(case when pc.rn\_desc = 1 then pc.in\_time end) as last\_time,  
max(case when pc.rn\_desc = 1 then pc.session\_id end) as last\_session,  
max(case when pc.rn\_desc = 1 then pc.cookie\_id end) as last\_cookie,  
max(case when pc.rn\_desc = 1 then pc.pv end) as last\_pv,  
max(case when pc.rn\_desc = 1 then pc.browser\_name end) as last\_browser,  
max(case when pc.rn\_desc = 1 then pc.visit\_os end) as last\_os,  
max(case when pc.rn\_desc = 1 then pc.visit\_ip end) as last\_ip,  
max(case when pc.rn\_desc = 1 then pc.province end) as last\_province,  
max(case when pc.rn\_desc = 1 then pc.city end) as last\_city,  
max(case when pc.rn\_asc = 1 then pc.in\_time end) as first\_time,  
max(case when pc.rn\_asc = 1 then pc.session\_id end) as first\_session,  
max(case when pc.rn\_asc = 1 then pc.cookie\_id end) as first\_cookie,  
max(case when pc.rn\_asc = 1 then pc.pv end) as first\_pv,  
max(case when pc.rn\_asc = 1 then pc.browser\_name end) as first\_browser,  
max(case when pc.rn\_asc = 1 then pc.visit\_os end) as first\_os,  
max(case when pc.rn\_asc = 1 then pc.visit\_ip end) as first\_ip,  
max(case when pc.rn\_asc = 1 then pc.province end) as first\_province,  
max(case when pc.rn\_asc = 1 then pc.city end) as first\_city,  
count(case when pc.dt7 = 1 then pc.session\_id end) as cnt7,  
count(case when pc.dt15 = 1 then pc.session\_id end) as cnt15,  
count(case when pc.dt30 = 1 then pc.session\_id end) as cnt30,  
count(case when pc.dt60 = 1 then pc.session\_id end) as cnt60,  
count(case when pc.dt90 = 1 then pc.session\_id end) as cnt90,  
count(distinct substring(case when pc.dt30 = 1 then pc.in\_time end,1,10)) day30,  
sum(case when pc.dt30 = 1 then pc.pv end) pv30,  
sum(case when pc.dt30 = 1 then pc.pv end) / count(distinct substring(case when pc.dt30 = 1 then pc.in\_time end,1,10)) avgpv30,  
sum(case when pc.dt30 = 1 and pc.hr025 = 1 then pc.pv end) pv025,  
sum(case when pc.dt30 = 1 and pc.hr627 = 1 then pc.pv end) pv627,  
sum(case when pc.dt30 = 1 and pc.hr829 = 1 then pc.pv end) pv829,  
sum(case when pc.dt30 = 1 and pc.hr10211 = 1 then pc.pv end) pv10211,  
sum(case when pc.dt30 = 1 and pc.hr12213 = 1 then pc.pv end) pv12213,  
sum(case when pc.dt30 = 1 and pc.hr14215 = 1 then pc.pv end) pv14215,  
sum(case when pc.dt30 = 1 and pc.hr16217 = 1 then pc.pv end) pv16217,  
sum(case when pc.dt30 = 1 and pc.hr18219 = 1 then pc.pv end) pv18219,  
sum(case when pc.dt30 = 1 and pc.hr20221 = 1 then pc.pv end) pv20221,  
sum(case when pc.dt30 = 1 and pc.hr22223 = 1 then pc.pv end) pv22223,  
count(case when mon.type = 'visit\_ip' then mon.content end) ipcnt,  
max(case when mon.type = 'visit\_ip' and mon.rn = 1 then mon.content end) topip,  
count(case when mon.type = 'cookie\_id' then mon.content end) ckcnt,  
max(case when mon.type = 'cookie\_id' and mon.rn = 1 then mon.content end) topck,  
max(case when mon.type = 'browser\_name' and mon.rn = 1 then mon.content end) topbr,  
max(case when mon.type = 'visit\_os' and mon.rn = 1 then mon.content end) topos,  
max(case when app.rn\_desc = 1 then app.log\_time end) last\_apptime,  
max(case when app.rn\_desc = 1 then app.app\_name end) last\_appname,  
max(case when app.rn\_desc = 1 then app.visit\_os end) last\_appos,  
max(case when app.rn\_desc = 1 then app.visit\_ip end) last\_appip,  
max(case when app.rn\_desc = 1 then app.province end) last\_appprovince,  
max(case when app.rn\_desc = 1 then app.city end) last\_appcity,  
max(case when app.rn\_asc = 1 then app.log\_time end) first\_apptime,  
max(case when app.rn\_asc = 1 then app.app\_name end) first\_appname,  
max(case when app.rn\_asc = 1 then app.visit\_os end) first\_appos,  
max(case when app.rn\_asc = 1 then app.visit\_ip end) first\_appip,  
max(case when app.rn\_asc = 1 then app.province end) first\_appprovince,  
max(case when app.rn\_asc = 1 then app.city end) first\_appcity,  
count(case when app.dt7 = 1 then app.user\_id end) as appcnt7,  
count(case when app.dt15 = 1 then app.user\_id end) as appcnt15,  
count(case when app.dt30 = 1 then app.user\_id end) as appcnt30,  
count(case when app.dt60 = 1 then app.user\_id end) as appcnt60,  
count(case when app.dt90 = 1 then app.user\_id end) as appcnt90,  
sum(case when app.dt30 = 1 and app.hr025 = 1 then 1 end) apppv025,  
sum(case when app.dt30 = 1 and app.hr627 = 1 then 1 end) apppv627,  
sum(case when app.dt30 = 1 and app.hr829 = 1 then 1 end) apppv829,  
sum(case when app.dt30 = 1 and app.hr10211 = 1 then 1 end) apppv10211,  
sum(case when app.dt30 = 1 and app.hr12213 = 1 then 1 end) apppv12213,  
sum(case when app.dt30 = 1 and app.hr14215 = 1 then 1 end) apppv14215,  
sum(case when app.dt30 = 1 and app.hr16217 = 1 then 1 end) apppv16217,  
sum(case when app.dt30 = 1 and app.hr18219 = 1 then 1 end) apppv18219,  
sum(case when app.dt30 = 1 and app.hr20221 = 1 then 1 end) apppv20221,  
sum(case when app.dt30 = 1 and app.hr22223 = 1 then 1 end) apppv22223  
from qfbap\_dm.dm\_user us  
left join   
(select   
user\_id,  
session\_id,  
cookie\_id,  
visit\_os,  
browser\_name,  
visit\_ip,  
province,  
city,  
pv,  
in\_time,  
out\_time,  
stay\_time,  
row\_number() over(distribute by user\_id sort by in\_time asc) as rn\_asc,  
row\_number() over(distribute by user\_id sort by in\_time desc) as rn\_desc,  
case when dt >= date\_sub('2019-11-14',6) then 1 end as dt7,  
case when dt >= date\_sub('2019-11-14',14) then 1 end as dt15,  
case when dt >= date\_sub('2019-11-14',29) then 1 end as dt30,  
case when dt >= date\_sub('2019-11-14',59) then 1 end as dt60,  
case when dt >= date\_sub('2019-11-14',89) then 1 end as dt90,  
--case when from\_unixtime(unix\_timestamp(log\_time,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') >= date\_sub('2020-01-05',6) and from\_unixtime(unix\_timestamp(log\_time,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') <= '2020-01-05' then 1 else 0 end day7,  
--case when from\_unixtime(unix\_timestamp(log\_time,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') >= date\_sub('2020-01-05',14) and from\_unixtime(unix\_timestamp(log\_time,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') <= '2020-01-05' then 1 end day15,  
--case when from\_unixtime(unix\_timestamp(log\_time,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') >= date\_sub('2020-01-05',29) and from\_unixtime(unix\_timestamp(log\_time,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') <= '2020-01-05' then 1 end day30,  
--case when from\_unixtime(unix\_timestamp(log\_time,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') >= date\_sub('2020-01-05',59) and from\_unixtime(unix\_timestamp(log\_time,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') <= '2020-01-05' then 1 end day60,  
--case when from\_unixtime(unix\_timestamp(log\_time,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') >= date\_sub('2020-01-05',89) and from\_unixtime(unix\_timestamp(log\_time,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') <= '2020-01-05' then 1 end day90,  
case when hour(in\_time) between 0 and 5 then 1 end as hr025,  
case when hour(in\_time) between 6 and 7 then 1 end as hr627,  
case when hour(in\_time) between 8 and 9 then 1 end as hr829,  
case when hour(in\_time) between 10 and 11 then 1 end as hr10211,  
case when hour(in\_time) between 12 and 13 then 1 end as hr12213,  
case when hour(in\_time) between 14 and 15 then 1 end as hr14215,  
case when hour(in\_time) between 16 and 17 then 1 end as hr16217,  
case when hour(in\_time) between 18 and 19 then 1 end as hr18219,  
case when hour(in\_time) between 20 and 21 then 1 end as hr20221,  
case when hour(in\_time) between 22 and 23 then 1 end as hr22223  
from qfbap\_dwd.dwd\_user\_pc\_pv  
where dt >= date\_sub('2019-11-14',89) and dt <= '2019-11-14'  
) pc on us.user\_id = pc.user\_id  
left join qfbap\_dws.dws\_user\_visit\_month1 mon on mon.dt = '2019-11-14' and mon.user\_id = us.user\_id   
left join   
(  
select   
user\_id,  
imei,  
log\_time,  
log\_hour,  
visit\_os,  
os\_version,  
app\_name,  
app\_version,  
device\_token,  
visit\_ip,  
province,  
city,  
row\_number() over(distribute by user\_id sort by log\_time asc) rn\_asc,  
row\_number() over(distribute by user\_id sort by log\_time desc) rn\_desc,  
case when dt >= date\_sub('2019-11-14',6) then 1 end dt7,  
case when dt >= date\_sub('2019-11-14',14) then 1 end dt15,  
case when dt >= date\_sub('2019-11-14',29) then 1 end dt30,  
case when dt >= date\_sub('2019-11-14',59) then 1 end dt60,  
case when dt >= date\_sub('2019-11-14',89) then 1 end dt90,  
case when log\_hour between 0 and 5 then 1 end as hr025,  
case when log\_hour between 6 and 7 then 1 end as hr627,  
case when log\_hour between 8 and 9 then 1 end as hr829,  
case when log\_hour between 10 and 11 then 1 end as hr10211,  
case when log\_hour between 12 and 13 then 1 end as hr12213,  
case when log\_hour between 14 and 15 then 1 end as hr14215,  
case when log\_hour between 16 and 17 then 1 end as hr16217,  
case when log\_hour between 18 and 19 then 1 end as hr18219,  
case when log\_hour between 20 and 21 then 1 end as hr20221,  
case when log\_hour between 22 and 23 then 1 end as hr22223  
from qfbap\_dwd.dwd\_user\_app\_pv   
where dt >= date\_sub('2019-11-14',89) and dt <= '2019-11-14'  
) app on us.user\_id = app.user\_id   
group by   
us.user\_id  
) all  
;  
  
-- 2、对今天的活跃用户  
-- 抽取活跃用户表  
create table qfbap\_dws.dws\_active\_user  
as  
select   
user\_id  
from  
(  
select  
pc.user\_id   
from qfbap\_dwd.dwd\_user\_pc\_pv pc  
group by   
pc.user\_id  
union all  
select   
app.user\_id  
from qfbap\_dwd.dwd\_user\_app\_pv app  
group by   
app.user\_id  
) all  
group by   
user\_id  
;  
  
--将上面语句的from qfbap\_dm.dm\_user us  
--替换成   
--from qfbap\_dws.dws\_active\_user us  
--测试  
  
  
  
-- union all  
with tmp as  
(  
select   
user\_id,   
max(last\_time) last\_time,   
max(last\_session) last\_session,   
max(last\_cookie) last\_cookie,   
max(last\_pv) last\_pv,   
max(last\_browser) last\_browser,   
max(last\_os) last\_os,   
max(first\_time) first\_time,   
max(first\_session) first\_session,   
max(first\_cookie) first\_cookie,   
max(first\_pv) first\_pv,   
max(first\_browser) first\_browser,   
max(first\_os) first\_os,   
max(cnt7) cnt7,   
max(cnt15) cnt15,   
max(cnt30) cnt30,   
max(cnt60) cnt60,   
max(cnt90) cnt90,   
max(day30) day30,   
max(pv30) pv30,   
max(avgpv30) avgpv30,   
max(pv025) pv025,   
max(pv627) pv627,   
max(pv829) pv829,   
max(pv10211) pv10211,   
max(pv12213) pv12213,   
max(pv14215) pv14215,   
max(pv16217) pv16217,   
max(pv18219) pv18219,   
max(pv20221) pv20221,   
max(pv22223) pv22223,   
max(ipcnt) ipcnt,   
max(topip) topip,   
max(ckcnt) ckcnt,   
max(topck) topck,   
max(topbr) topbr,   
max(topos) topos,   
max(last\_apptime) last\_apptime,   
max(last\_appname) last\_appname,   
max(last\_appos) last\_appos,   
max(last\_appip) last\_appip,   
max(first\_apptime) first\_apptime,   
max(first\_appname) first\_appname,   
max(first\_appos) first\_appos,   
max(first\_appip) first\_appip,   
max(appcnt7) appcnt7,   
max(appcnt15) appcnt15,   
max(appcnt30) appcnt30,   
max(appcnt60) appcnt60,   
max(appcnt90) appcnt90,   
max(apppv025) apppv025,   
max(apppv627) apppv627,   
max(apppv829) apppv829,   
max(apppv10211) apppv10211,   
max(apppv12213) apppv12213,   
max(apppv14215) apppv14215,   
max(apppv16217) apppv16217,   
max(apppv18219) apppv18219,   
max(apppv20221) apppv20221,   
max(apppv22223) apppv22223  
from   
(  
select  
pc.user\_id,  
max(case when pc.rn\_desc = 1 then pc.in\_time end) as last\_time,  
max(case when pc.rn\_desc = 1 then pc.session\_id end) as last\_session,  
max(case when pc.rn\_desc = 1 then pc.cookie\_id end) as last\_cookie,  
max(case when pc.rn\_desc = 1 then pc.pv end) as last\_pv,  
max(case when pc.rn\_desc = 1 then pc.browser\_name end) as last\_browser,  
max(case when pc.rn\_desc = 1 then pc.visit\_os end) as last\_os,  
max(case when pc.rn\_desc = 1 then pc.visit\_ip end) as last\_ip,  
max(case when pc.rn\_desc = 1 then pc.province end) as last\_province,  
max(case when pc.rn\_desc = 1 then pc.city end) as last\_city,  
max(case when pc.rn\_asc = 1 then pc.in\_time end) as first\_time,  
max(case when pc.rn\_asc = 1 then pc.session\_id end) as first\_session,  
max(case when pc.rn\_asc = 1 then pc.cookie\_id end) as first\_cookie,  
max(case when pc.rn\_asc = 1 then pc.pv end) as first\_pv,  
max(case when pc.rn\_asc = 1 then pc.browser\_name end) as first\_browser,  
max(case when pc.rn\_asc = 1 then pc.visit\_os end) as first\_os,  
max(case when pc.rn\_asc = 1 then pc.visit\_ip end) as first\_ip,  
max(case when pc.rn\_asc = 1 then pc.province end) as first\_province,  
max(case when pc.rn\_asc = 1 then pc.city end) as first\_city,  
count(case when pc.dt7 = 1 then pc.session\_id end) as cnt7,  
count(case when pc.dt15 = 1 then pc.session\_id end) as cnt15,  
count(case when pc.dt30 = 1 then pc.session\_id end) as cnt30,  
count(case when pc.dt60 = 1 then pc.session\_id end) as cnt60,  
count(case when pc.dt90 = 1 then pc.session\_id end) as cnt90,  
count(distinct substring(case when pc.dt30 = 1 then pc.in\_time end,1,10)) day30,  
sum(case when pc.dt30 = 1 then pc.pv end) pv30,  
sum(case when pc.dt30 = 1 then pc.pv end) / count(distinct substring(case when pc.dt30 = 1 then pc.in\_time end,1,10)) avgpv30,  
sum(case when pc.dt30 = 1 and pc.hr025 = 1 then pc.pv end) pv025,  
sum(case when pc.dt30 = 1 and pc.hr627 = 1 then pc.pv end) pv627,  
sum(case when pc.dt30 = 1 and pc.hr829 = 1 then pc.pv end) pv829,  
sum(case when pc.dt30 = 1 and pc.hr10211 = 1 then pc.pv end) pv10211,  
sum(case when pc.dt30 = 1 and pc.hr12213 = 1 then pc.pv end) pv12213,  
sum(case when pc.dt30 = 1 and pc.hr14215 = 1 then pc.pv end) pv14215,  
sum(case when pc.dt30 = 1 and pc.hr16217 = 1 then pc.pv end) pv16217,  
sum(case when pc.dt30 = 1 and pc.hr18219 = 1 then pc.pv end) pv18219,  
sum(case when pc.dt30 = 1 and pc.hr20221 = 1 then pc.pv end) pv20221,  
sum(case when pc.dt30 = 1 and pc.hr22223 = 1 then pc.pv end) pv22223,  
null ipcnt,  
null topip,  
null ckcnt,  
null topck,  
null topbr,  
null topos,  
null last\_apptime,  
null last\_appname,  
null last\_appos,  
null last\_appip,  
null last\_appprovince,  
null last\_appcity,  
null first\_apptime,  
null first\_appname,  
null first\_appos,  
null first\_appip,  
null first\_appprovince,  
null first\_appcity,  
null as appcnt7,  
null as appcnt15,  
null as appcnt30,  
null as appcnt60,  
null as appcnt90,  
null apppv025,  
null apppv627,  
null apppv829,  
null apppv10211,  
null apppv12213,  
null apppv14215,  
null apppv16217,  
null apppv18219,  
null apppv20221,  
null apppv22223  
from   
(  
select   
user\_id,  
session\_id,  
cookie\_id,  
visit\_os,  
browser\_name,  
visit\_ip,  
province,  
city,  
pv,  
in\_time,  
out\_time,  
stay\_time,  
row\_number() over(distribute by user\_id sort by in\_time asc) as rn\_asc,  
row\_number() over(distribute by user\_id sort by in\_time desc) as rn\_desc,  
case when dt >= date\_sub('2019-11-14',6) then 1 end as dt7,  
case when dt >= date\_sub('2019-11-14',14) then 1 end as dt15,  
case when dt >= date\_sub('2019-11-14',29) then 1 end as dt30,  
case when dt >= date\_sub('2019-11-14',59) then 1 end as dt60,  
case when dt >= date\_sub('2019-11-14',89) then 1 end as dt90,  
case when hour(in\_time) between 0 and 5 then 1 end as hr025,  
case when hour(in\_time) between 6 and 7 then 1 end as hr627,  
case when hour(in\_time) between 8 and 9 then 1 end as hr829,  
case when hour(in\_time) between 10 and 11 then 1 end as hr10211,  
case when hour(in\_time) between 12 and 13 then 1 end as hr12213,  
case when hour(in\_time) between 14 and 15 then 1 end as hr14215,  
case when hour(in\_time) between 16 and 17 then 1 end as hr16217,  
case when hour(in\_time) between 18 and 19 then 1 end as hr18219,  
case when hour(in\_time) between 20 and 21 then 1 end as hr20221,  
case when hour(in\_time) between 22 and 23 then 1 end as hr22223  
from qfbap\_dwd.dwd\_user\_pc\_pv  
where dt >= date\_sub('2019-11-14',89) and dt <= '2019-11-14'  
) pc  
group by pc.user\_id  
union all  
select  
mon.user\_id,  
null last\_time,  
null last\_session,  
null last\_cookie,  
null last\_pv,  
null as last\_browser,  
null last\_os,  
null last\_ip,  
null last\_province,  
null last\_city,  
null first\_time,  
null first\_session,  
null first\_cookie,  
null first\_pv,  
null first\_browser,  
null first\_os,  
null first\_ip,  
null first\_province,  
null first\_city,  
null cnt7,  
null cnt15,  
null cnt30,  
null cnt60,  
null cnt90,  
null day30,  
null pv30,  
null avgpv30,  
null pv025,  
null pv627,  
null pv829,  
null pv10211,  
null pv12213,  
null pv14215,  
null pv16217,  
null pv18219,  
null pv20221,  
null pv22223,  
count(case when mon.type = 'visit\_ip' then mon.content end) ipcnt,  
max(case when mon.type = 'visit\_ip' and mon.rn = 1 then mon.content end) topip,  
count(case when mon.type = 'cookie\_id' then mon.content end) ckcnt,  
max(case when mon.type = 'cookie\_id' and mon.rn = 1 then mon.content end) topck,  
max(case when mon.type = 'browser\_name' and mon.rn = 1 then mon.content end) topbr,  
max(case when mon.type = 'visit\_os' and mon.rn = 1 then mon.content end) topos,  
null last\_apptime,  
null last\_appname,  
null last\_appos,  
null last\_appip,  
null last\_appprovince,  
null last\_appcity,  
null first\_apptime,  
null first\_appname,  
null first\_appos,  
null first\_appip,  
null first\_appprovince,  
null first\_appcity,  
null as appcnt7,  
null as appcnt15,  
null as appcnt30,  
null as appcnt60,  
null as appcnt90,  
null apppv025,  
null apppv627,  
null apppv829,  
null apppv10211,  
null apppv12213,  
null apppv14215,  
null apppv16217,  
null apppv18219,  
null apppv20221,  
null apppv22223  
from qfbap\_dws.dws\_user\_visit\_month1 mon where mon.dt = '2019-11-14'  
group by   
mon.user\_id  
union all  
select  
app.user\_id,  
null last\_time,  
null last\_session,  
null last\_cookie,  
null last\_pv,  
null as last\_browser,  
null last\_os,  
null last\_ip,  
null last\_province,  
null last\_city,  
null first\_time,  
null first\_session,  
null first\_cookie,  
null first\_pv,  
null first\_browser,  
null first\_os,  
null first\_ip,  
null first\_province,  
null first\_city,  
null cnt7,  
null cnt15,  
null cnt30,  
null cnt60,  
null cnt90,  
null day30,  
null pv30,  
null avgpv30,  
null pv025,  
null pv627,  
null pv829,  
null pv10211,  
null pv12213,  
null pv14215,  
null pv16217,  
null pv18219,  
null pv20221,  
null pv22223,  
null ipcnt,  
null topip,  
null ckcnt,  
null topck,  
null topbr,  
null topos,  
max(case when app.rn\_desc = 1 then app.log\_time end) last\_apptime,  
max(case when app.rn\_desc = 1 then app.app\_name end) last\_appname,  
max(case when app.rn\_desc = 1 then app.visit\_os end) last\_appos,  
max(case when app.rn\_desc = 1 then app.visit\_ip end) last\_appip,  
max(case when app.rn\_desc = 1 then app.province end) last\_appprovince,  
max(case when app.rn\_desc = 1 then app.city end) last\_appcity,  
max(case when app.rn\_asc = 1 then app.log\_time end) first\_apptime,  
max(case when app.rn\_asc = 1 then app.app\_name end) first\_appname,  
max(case when app.rn\_asc = 1 then app.visit\_os end) first\_appos,  
max(case when app.rn\_asc = 1 then app.visit\_ip end) first\_appip,  
max(case when app.rn\_asc = 1 then app.province end) first\_appprovince,  
max(case when app.rn\_asc = 1 then app.city end) first\_appcity,  
count(case when app.dt7 = 1 then app.user\_id end) as appcnt7,  
count(case when app.dt15 = 1 then app.user\_id end) as appcnt15,  
count(case when app.dt30 = 1 then app.user\_id end) as appcnt30,  
count(case when app.dt60 = 1 then app.user\_id end) as appcnt60,  
count(case when app.dt90 = 1 then app.user\_id end) as appcnt90,  
sum(case when app.dt30 = 1 and app.hr025 = 1 then 1 end) apppv025,  
sum(case when app.dt30 = 1 and app.hr627 = 1 then 1 end) apppv627,  
sum(case when app.dt30 = 1 and app.hr829 = 1 then 1 end) apppv829,  
sum(case when app.dt30 = 1 and app.hr10211 = 1 then 1 end) apppv10211,  
sum(case when app.dt30 = 1 and app.hr12213 = 1 then 1 end) apppv12213,  
sum(case when app.dt30 = 1 and app.hr14215 = 1 then 1 end) apppv14215,  
sum(case when app.dt30 = 1 and app.hr16217 = 1 then 1 end) apppv16217,  
sum(case when app.dt30 = 1 and app.hr18219 = 1 then 1 end) apppv18219,  
sum(case when app.dt30 = 1 and app.hr20221 = 1 then 1 end) apppv20221,  
sum(case when app.dt30 = 1 and app.hr22223 = 1 then 1 end) apppv22223  
from  
(  
select   
user\_id,  
imei,  
log\_time,  
log\_hour,  
visit\_os,  
os\_version,  
app\_name,  
app\_version,  
device\_token,  
visit\_ip,  
province,  
city,  
row\_number() over(distribute by user\_id sort by log\_time asc) rn\_asc,  
row\_number() over(distribute by user\_id sort by log\_time desc) rn\_desc,  
case when dt >= date\_sub('2019-11-14',6) then 1 end dt7,  
case when dt >= date\_sub('2019-11-14',14) then 1 end dt15,  
case when dt >= date\_sub('2019-11-14',29) then 1 end dt30,  
case when dt >= date\_sub('2019-11-14',59) then 1 end dt60,  
case when dt >= date\_sub('2019-11-14',89) then 1 end dt90,  
case when log\_hour between 0 and 5 then 1 end as hr025,  
case when log\_hour between 6 and 7 then 1 end as hr627,  
case when log\_hour between 8 and 9 then 1 end as hr829,  
case when log\_hour between 10 and 11 then 1 end as hr10211,  
case when log\_hour between 12 and 13 then 1 end as hr12213,  
case when log\_hour between 14 and 15 then 1 end as hr14215,  
case when log\_hour between 16 and 17 then 1 end as hr16217,  
case when log\_hour between 18 and 19 then 1 end as hr18219,  
case when log\_hour between 20 and 21 then 1 end as hr20221,  
case when log\_hour between 22 and 23 then 1 end as hr22223  
from qfbap\_dwd.dwd\_user\_app\_pv   
where dt >= date\_sub('2019-11-14',89) and dt <= '2019-11-14'  
) app  
group by   
app.user\_id  
) all  
group by user\_id  
) tmp  
  
select  
user\_id,   
last\_time,   
last\_session,   
last\_cookie,   
last\_pv,   
last\_browser,   
last\_os,   
first\_time,   
first\_session,   
first\_cookie,   
first\_pv,   
first\_browser,   
first\_os,   
cnt7,   
cnt15,   
cnt30,   
cnt60,   
cnt90,   
day30,   
pv30,   
avgpv30,   
pv025,   
pv627,   
pv829,   
pv10211,   
pv12213,   
pv14215,   
pv16217,   
pv18219,   
pv20221,   
pv22223,   
ipcnt,   
topip,   
ckcnt,   
topck,   
topbr,   
topos,   
last\_apptime,   
last\_appname,   
last\_appos,   
last\_appip,   
first\_apptime,   
first\_appname,   
first\_appos,   
first\_appip,   
appcnt7,   
appcnt15,   
appcnt30,   
appcnt60,   
appcnt90,   
apppv025,   
apppv627,   
apppv829,   
apppv10211,   
apppv12213,   
apppv14215,   
apppv16217,   
apppv18219,   
apppv20221,   
apppv22223,  
case when last\_time > last\_apptime then last\_ip else last\_appip end maxip,  
case when last\_time > last\_apptime then last\_province else last\_appprovince end maxprovince,  
case when last\_time > last\_apptime then last\_city else last\_appcity end maxcity,  
case when first\_time < first\_apptime then first\_ip else first\_appip end minip,  
case when first\_time < first\_apptime then first\_province else first\_appprovince end minprovince,  
case when first\_time < first\_apptime then first\_city else first\_appcity end mincity  
from tmp  
;

### 5.5.3 用户订单模型

-- 1、创建dwd表  
CREATE TABLE IF NOT EXISTS qfbap\_dwd.dwd\_order\_item (  
 user\_id bigint ,  
 order\_id bigint ,  
 order\_no string ,  
 goods\_id bigint ,  
 goods\_no string ,  
 goods\_name string ,  
 goods\_amount int ,  
 shop\_id bigint ,  
 shop\_name string ,  
 curr\_price decimal(18,2) ,  
 market\_price decimal(18,2) ,  
 discount decimal(18,2) ,  
 cost\_price decimal(18,2) ,  
 first\_cart bigint ,  
 first\_cart\_name string ,  
 second\_cart bigint ,  
 second\_cart\_name string ,  
 third\_cart bigint ,  
 third\_cart\_name string ,  
 goods\_desc string ,  
 dw\_date string   
)   
partitioned by (dt string)  
;  
  
CREATE TABLE IF NOT EXISTS qfbap\_dwd.dwd\_us\_order (  
 order\_id bigint ,  
 order\_no string ,  
 order\_date string ,  
 user\_id bigint ,  
 user\_name string ,  
 order\_money decimal(18,2) ,  
 order\_type int ,  
 order\_status int ,  
 pay\_status int ,  
 pay\_type int ,  
 order\_source string ,  
 update\_time string ,  
 order\_addr string,  
 user\_order\_flag tinyint,  
 trade\_type tinyint,  
 trade\_time string,  
 dw\_date string  
)  
partitioned by (dt string)  
;  
  
  
-- 加载数据  
-- 加载订单表  
insert overwrite table qfbap\_dwd.dwd\_us\_order partition(dt='2020-01-05')  
select  
uo.order\_id,  
uo.order\_no,  
uo.order\_date,  
uo.user\_id,  
uo.user\_name,  
uo.order\_money,  
uo.order\_type,  
uo.order\_status,  
uo.pay\_status,  
uo.pay\_type,  
uo.order\_source,  
uo.update\_time,  
ua.order\_addr,  
ua.user\_order\_flag,  
bt.trade\_type,  
bt.trade\_time,  
current\_timestamp() dw\_date  
from qfbap\_ods.ods\_us\_order uo  
left join qfbap\_ods.ods\_order\_delivery od on od.dt = '2020-01-05' and od.order\_id = uo.order\_id   
left join qfbap\_ods.ods\_user\_addr ua on ua.addr\_id = od.addr\_id  
left join qfbap\_ods.ods\_biz\_trade bt on bt.dt = '2020-01-05' and bt.order\_id = uo.order\_id  
where uo.dt = '2020-01-05'  
;  
  
-- 加载订单明细数据  
insert overwrite table qfbap\_dwd.dwd\_order\_item partition(dt = '2020-01-05')  
select  
user\_id,  
order\_id,  
order\_no,  
goods\_id,  
goods\_no,  
goods\_name,  
goods\_amount,  
shop\_id,  
shop\_name,  
curr\_price,  
market\_price,  
discount,  
cost\_price,  
first\_cart,  
first\_cart\_name,  
second\_cart,  
second\_cart\_name,  
third\_cart,  
third\_cart\_name,  
goods\_desc,  
current\_timestamp() dw\_date   
from qfbap\_ods.ods\_order\_item  
where dt = '2020-01-05'  
;  
  
  
-- 创建dws表  
create table if not exists qfbap\_dws.dws\_user\_order\_all  
(  
 user\_id bigint ,  
 type string,  
 content string,  
 cnt bigint,  
 rn bigint,  
 dw\_date string  
)  
partitioned by (dt string)  
stored as orc  
;  
  
--最常用收货地址,  
--最常用支付方式,  
  
-- 历史数据初始化  
insert overwrite table qfbap\_dws.dws\_user\_order\_all partition(dt = '2020-01-04')  
select  
user\_id,  
type,  
content,  
cnt,  
row\_number() over(distribute by user\_id,type sort by cnt desc) rn  
from   
(  
select  
uo.user\_id,  
'order\_addr' as type,  
uo.order\_addr as content,  
count(distinct uo.order\_addr) cnt  
from qfbap\_dwd.dwd\_us\_order uo  
where dt <= '2020-01-04'  
group by   
uo.user\_id,  
uo.order\_addr  
union all  
select  
uo.user\_id,  
'pay\_type' as type,  
uo.pay\_type as content,  
count(distinct uo.pay\_type) cnt  
from qfbap\_dwd.dwd\_us\_order uo  
where dt <= '2020-01-04'  
group by   
uo.user\_id,  
uo.pay\_type  
) all  
;  
  
-- 每天的定期加载  
from   
(  
select  
user\_id,  
type,  
content,  
sum(cnt) cnt  
from   
(  
select  
uo.user\_id,  
'order\_addr' as type,  
uo.order\_addr as content,  
count(distinct uo.order\_addr) cnt  
from qfbap\_dwd.dwd\_us\_order uo  
where dt = '2020-01-05'  
group by   
uo.user\_id,  
uo.order\_addr  
union all  
select  
uo.user\_id,  
'pay\_type' as type,  
uo.pay\_type as content,  
count(distinct uo.pay\_type) cnt  
from qfbap\_dwd.dwd\_us\_order uo  
where dt = '2020-01-05'  
group by   
uo.user\_id,  
uo.pay\_type  
union all  
select  
user\_id,  
type,  
content,  
cnt   
from qfbap\_dws.dws\_user\_order\_all  
where dt = '2020-01-04'  
) all  
group by   
user\_id,  
type,  
content  
) a  
insert overwrite table qfbap\_dws.dws\_user\_order\_all partition(dt = '2020-01-05')  
select  
user\_id,  
type,  
content,  
cnt,  
row\_number() over(distribute by user\_id,type sort by cnt desc) rn,  
current\_timestamp() dw\_date  
;  
  
  
-- 创建dm表  
CREATE TABLE IF NOT EXISTS `qfbap\_dm.dm\_user\_order`(  
 `user\_id` bigint,   
 `first\_order\_date` string,   
 `last\_order\_date` string,   
 `first\_days` int,   
 `last\_days` int,   
 `nums30` bigint,   
 `moneys30` decimal(28,2),   
 `nums60` bigint,   
 `moneys60` decimal(28,2),   
 `nums90` bigint,   
 `moneys90` decimal(28,2),   
 `rejnums30` bigint,   
 `rejmoneys30` decimal(28,2),   
 `rejnums60` bigint,   
 `rejmoneys60` decimal(28,2),   
 `rejnums90` bigint,   
 `rejmoneys90` decimal(28,2),   
 `rejavg90` decimal(38,16),   
 `maxmoney` decimal(18,2),   
 `minmoney` decimal(18,2),   
 `totalnums` bigint,   
 `totalmoneys` decimal(28,2),   
 `avg1` decimal(38,16),   
 `topaddr` string,   
 `toppaytype` string,   
 `backitemnums` bigint,   
 `backmoneys` decimal(28,2),   
 `rejitemnums` bigint,   
 `rejmoneys` decimal(28,2),   
 `rejtimes` string,   
 `schoolnums` bigint,   
 `companynums` bigint,   
 `homenums` bigint,   
 `mornums` bigint,   
 `amnums` bigint,   
 `noonnums` bigint,   
 `pmnums` bigint,   
 `ntnums` bigint)  
partitioned by (dt string)  
;  
  
-- 初始化历史数据  
INSERT OVERWRITE TABLE qfbap\_dm.dm\_user\_order partition(dt = '2020-01-04')  
select  
od.user\_id,  
max(case when od.rn = 1 then od.order\_date end) first\_order\_date,  
max(case when od.drn = 1 then od.order\_date end) last\_order\_date,  
datediff(current\_date(),max(case when od.rn = 1 then od.order\_date end)) first\_days,  
datediff(current\_date(),max(case when od.drn = 1 then od.order\_date end)) last\_days,  
count(case when day30 = 1 and od.trade\_type = 1 then od.order\_id end) nums30,  
sum(case when day30 = 1 and od.trade\_type = 1 then od.order\_money else 0 end) moneys30,  
count(case when day60 = 1 and od.trade\_type = 1 then od.order\_id end) nums60,  
sum(case when day60 = 1 and od.trade\_type = 1 then od.order\_money else 0 end) moneys60,  
count(case when day90 = 1 and od.trade\_type = 1 then od.order\_id end) nums90,  
sum(case when day90 = 1 and od.trade\_type = 1 then od.order\_money else 0 end) moneys90,  
count(case when day30 = 1 then od.order\_id end) rejnums30,  
sum(case when day30 = 1 then od.order\_money else 0 end) rejmoneys30,  
count(case when day60 = 1 then od.order\_id end) rejnums60,  
sum(case when day60 = 1 then od.order\_money else 0 end) rejmoneys60,  
count(case when day90 = 1 then od.order\_id end) rejnums90,  
sum(case when day90 = 1 then od.order\_money else 0 end) rejmoneys90,  
sum(case when day90 = 1 then od.order\_money else 0 end) / count(case when day90 = 1 then od.order\_id end) rejavg90,  
max(od.order\_money) maxmoney,  
min(od.order\_money) minmoney,  
count(od.order\_id) totalnums,  
sum(case when od.trade\_type = 1 then od.order\_money else 0 end) totalmoneys,  
sum(case when od.trade\_type = 1 then od.order\_money else 0 end) / count(case when od.trade\_type = 1 then od.order\_id end) avg1,  
max(case when ua.type = 'order\_addr' and ua.rn = 1 then ua.content end) topaddr,  
max(case when ua.type = 'pay\_type' and ua.rn = 1 then ua.content end) toppaytype,  
sum(case when od.trade\_type = -1 then oi.itemnum end) backitemnums,  
sum(case when od.trade\_type = -1 then od.order\_money else 0 end) backmoneys,  
sum(case when od.trade\_type = -2 then oi.itemnum end) rejitemnums,  
sum(case when od.trade\_type = -2 then od.order\_money else 0 end) rejmoneys,  
max(case when od.trade\_type = -1 then od.order\_date end) rejtimes,  
sum(case when od.user\_order\_flag = 1 then 1 else 0 end) schoolnums,  
sum(case when od.user\_order\_flag = 2 then 1 else 0 end) companynums,  
sum(case when od.user\_order\_flag = 3 then 1 else 0 end) homenums,  
sum(od.mor) mornums,  
sum(od.am) amnums,  
sum(od.noon) noonnums,  
sum(od.pm) pmnums,  
sum(od.nt) ntnums  
from  
(  
select  
\*,  
row\_number() over(distribute by uo.user\_id sort by uo.order\_date asc) rn,  
row\_number() over(distribute by uo.user\_id sort by uo.order\_date desc) drn,  
case when from\_unixtime(unix\_timestamp(uo.order\_date,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') >= date\_sub('2020-01-04',29) and from\_unixtime(unix\_timestamp(uo.order\_date,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') <= '2020-01-04' then 1 else 0 end day30,  
case when from\_unixtime(unix\_timestamp(uo.order\_date,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') >= date\_sub('2020-01-04',59) and from\_unixtime(unix\_timestamp(uo.order\_date,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') <= '2020-01-04' then 1 else 0 end day60,  
case when from\_unixtime(unix\_timestamp(uo.order\_date,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') >= date\_sub('2020-01-04',89) and from\_unixtime(unix\_timestamp(uo.order\_date,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') <= '2020-01-04' then 1 else 0 end day90,  
case when hour(uo.order\_date) between 0 and 5 then 1 end mor,  
case when hour(uo.order\_date) between 6 and 12 then 1 end am,  
case when hour(uo.order\_date) between 13 and 15 then 1 end noon,  
case when hour(uo.order\_date) between 16 and 20 then 1 end pm,  
case when hour(uo.order\_date) between 21 and 23 then 1 end nt  
from qfbap\_dwd.dwd\_us\_order uo  
where uo.dt <= '2020-01-04'  
) od  
left join (select \* from qfbap\_dws.dws\_user\_order\_all where dt = '2020-01-04' and rn = 1) ua on ua.user\_id = od.user\_id  
left join (select order\_id,sum(goods\_amount) as itemnum from qfbap\_dwd.dwd\_order\_item where dt <= '2020-01-04' group by order\_id) oi on oi.order\_id = od.order\_id  
group by   
od.user\_id  
;  
  
-- 分析语句  
INSERT OVERWRITE TABLE qfbap\_dm.dm\_user\_order partition(dt = '2020-01-05')  
select  
user\_id,  
max(first\_order\_date) first\_order\_date,  
max(last\_order\_date) last\_order\_date,  
max(first\_days) first\_days,  
max(last\_days) last\_days,  
max(nums30) nums30,  
max(moneys30) moneys30,  
max(nums60) nums60,  
max(moneys60) moneys60,  
max(nums90) nums90,  
max(moneys90) moneys90,  
max(rejnums30) rejnums30,  
max(rejmoneys30) rejmoneys30,  
max(rejnums60) rejnums60,  
max(rejmoneys60) rejmoneys60,  
max(rejnums90) rejnums90,  
max(rejmoneys90) rejmoneys90,  
max(rejavg90) rejavg90,  
max(maxmoney) maxmoney,  
min(minmoney) minmoney,  
sum(totalnums) totalnums,  
sum(totalmoneys) totalmoneys,  
sum(totalmoneys) / sum(totalnums) avg1,  
max(topaddr) topaddr,  
max(toppaytype) toppaytype,  
sum(backitemnums) backitemnums,  
sum(backmoneys) backmoneys,  
sum(rejitemnums) rejitemnums,  
sum(rejmoneys) rejmoneys,  
sum(rejtimes) rejtimes,  
sum(schoolnums) schoolnums,  
sum(companynums) companynums,  
sum(homenums) homenums,  
sum(mornums) mornums,  
sum(amnums) amnums,  
sum(noonnums) noonnums,  
sum(pmnums) pmnums,  
sum(ntnums) ntnums  
from  
(  
select  
od.user\_id,  
max(case when od.rn = 1 then od.order\_date end) first\_order\_date,  
max(case when od.drn = 1 then od.order\_date end) last\_order\_date,  
datediff(current\_date(),max(case when od.rn = 1 then od.order\_date end)) first\_days,  
datediff(current\_date(),max(case when od.drn = 1 then od.order\_date end)) last\_days,  
count(case when day30 = 1 and od.trade\_type = 1 then od.order\_id end) nums30,  
nvl(sum(case when day30 = 1 and od.trade\_type = 1 then od.order\_money else 0 end),0) moneys30,  
count(case when day60 = 1 and od.trade\_type = 1 then od.order\_id end) nums60,  
nvl(sum(case when day60 = 1 and od.trade\_type = 1 then od.order\_money else 0 end),0) moneys60,  
count(case when day90 = 1 and od.trade\_type = 1 then od.order\_id end) nums90,  
nvl(sum(case when day90 = 1 and od.trade\_type = 1 then od.order\_money else 0 end),0) moneys90,  
count(case when day30 = 1 then od.order\_id end) rejnums30,  
nvl(sum(case when day30 = 1 then od.order\_money else 0 end),0) rejmoneys30,  
count(case when day60 = 1 then od.order\_id end) rejnums60,  
nvl(sum(case when day60 = 1 then od.order\_money else 0 end),0) rejmoneys60,  
count(case when day90 = 1 then od.order\_id end) rejnums90,  
nvl(sum(case when day90 = 1 then od.order\_money else 0 end),0) rejmoneys90,  
nvl(sum(case when day90 = 1 then od.order\_money else 0 end) / count(case when day90 = 1 then od.order\_id end),0) rejavg90,  
max(case when day1 = 1 then od.order\_money end) maxmoney,  
min(case when day1 = 1 then od.order\_money end) minmoney,  
count(case when day1 = 1 and od.trade\_type = 1 then od.order\_id end) totalnums,  
nvl(sum(case when day1 = 1 and od.trade\_type = 1 then od.order\_money else 0 end),0) totalmoneys,  
nvl(sum(case when day1 = 1 and od.trade\_type = 1 then od.order\_money else 0 end) / count(case when day1 = 1 and od.trade\_type = 1 then od.order\_id end),0) avg1,  
max(case when ua.type = 'order\_addr' and ua.rn = 1 then ua.content end) topaddr,  
max(case when ua.type = 'pay\_type' and ua.rn = 1 then ua.content end) toppaytype,  
nvl(sum(case when day1 = 1 and od.trade\_type = -1 then oi.itemnum end),0) backitemnums,  
nvl(sum(case when day1 = 1 and od.trade\_type = -1 then od.order\_money else 0 end),0) backmoneys,  
nvl(sum(case when day1 = 1 and od.trade\_type = -2 then oi.itemnum end),0) rejitemnums,  
nvl(sum(case when day1 = 1 and od.trade\_type = -2 then od.order\_money else 0 end),0) rejmoneys,  
max(case when day1 = 1 and od.trade\_type = -1 then od.order\_date end) rejtimes,  
nvl(sum(case when day1 = 1 and od.user\_order\_flag = 1 then 1 else 0 end),0) schoolnums,  
nvl(sum(case when day1 = 1 and od.user\_order\_flag = 2 then 1 else 0 end),0) companynums,  
nvl(sum(case when day1 = 1 and od.user\_order\_flag = 3 then 1 else 0 end),0) homenums,  
nvl(sum(case when day1 = 1 then od.mor else 0 end),0) mornums,  
nvl(sum(case when day1 = 1 then od.am else 0 end),0) amnums,  
nvl(sum(case when day1 = 1 then od.noon else 0 end),0) noonnums,  
nvl(sum(case when day1 = 1 then od.pm else 0 end),0) pmnums,  
nvl(sum(case when day1 = 1 then od.nt else 0 end),0) ntnums  
from  
(  
select  
\*,  
row\_number() over(distribute by uo.user\_id sort by uo.order\_date asc) rn,  
row\_number() over(distribute by uo.user\_id sort by uo.order\_date desc) drn,  
case when from\_unixtime(unix\_timestamp(uo.order\_date,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') >= date\_sub('2020-01-05',29) and from\_unixtime(unix\_timestamp(uo.order\_date,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') <= '2020-01-05' then 1 else 0 end day30,  
case when from\_unixtime(unix\_timestamp(uo.order\_date,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') >= date\_sub('2020-01-05',59) and from\_unixtime(unix\_timestamp(uo.order\_date,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') <= '2020-01-05' then 1 else 0 end day60,  
case when from\_unixtime(unix\_timestamp(uo.order\_date,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') >= date\_sub('2020-01-05',89) and from\_unixtime(unix\_timestamp(uo.order\_date,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') <= '2020-01-05' then 1 else 0 end day90,  
case when from\_unixtime(unix\_timestamp(uo.order\_date,'yyyy-MM-dd HH:mm:ss'),'yyyy-MM-dd') = '2020-01-05' then 1 else 0 end day1,  
case when hour(uo.order\_date) between 0 and 5 then 1 end mor,  
case when hour(uo.order\_date) between 6 and 12 then 1 end am,  
case when hour(uo.order\_date) between 13 and 15 then 1 end noon,  
case when hour(uo.order\_date) between 16 and 20 then 1 end pm,  
case when hour(uo.order\_date) between 21 and 23 then 1 end nt  
from qfbap\_dwd.dwd\_us\_order uo  
where uo.dt >= date\_sub('2020-01-05',89) and uo.dt <= '2020-01-05'  
) od  
left join (select \* from qfbap\_dws.dws\_user\_order\_all where dt = '2020-01-05' and rn = 1) ua on ua.user\_id = od.user\_id  
left join (select order\_id,sum(goods\_amount) as itemnum from qfbap\_dwd.dwd\_order\_item where dt = '2020-01-05' group by order\_id) oi on oi.order\_id = od.order\_id  
group by   
od.user\_id  
union all  
select  
user\_id,  
null first\_order\_date,  
null last\_order\_date,  
null first\_days,  
null last\_days,  
null nums30,  
null moneys30,  
null nums60,  
null moneys60,  
null nums90,  
null moneys90,  
null rejnums30,  
null rejmoneys30,  
null rejnums60,  
null rejmoneys60,  
null rejnums90,  
null rejmoneys90,  
null rejavg90,  
maxmoney,  
minmoney,  
totalnums,  
totalmoneys,  
null avg1,  
null topaddr,  
null toppaytype,  
backitemnums,  
backmoneys,  
rejitemnums,  
rejmoneys,  
rejtimes,  
schoolnums,  
companynums,  
homenums,  
mornums,  
amnums,  
noonnums,  
pmnums,  
ntnums  
from qfbap\_dm.dm\_user\_order  
where dt = '2020-01-04'  
) all  
group by   
user\_id  
;

### 5.5.4 项目调度案例

azkaban安装使用文档详见：千锋大数据任务调度技术之azkaban。

#### 5.5.4.1 建立工作流前的准备

我们的定期ETL需要使用Azkaban中的FS、Sqoop和SSH三种动作，其中增量数据抽取要用到Sqoop job。由于Azkaban在执行这些动作时存在一些特殊要求，因此在定义工作流前先要进行适当的配置。

1. 启动Azkaban服务
2. 配置Sqoop的metastore  
    缺省时，Sqoop metastore自动连接存储在~/.sqoop/.目录下的本地嵌入式数据库。然而要在Azkaban中执行Sqoop job需要Sqoop使用共享的元数据存储，这里选择将sqoop的元数据存储到mysql中。

* 否则会报类似如下的错误：ERROR org.apache.sqoop.metastore.hsqldb.HsqldbJobStorage - Cannot restore job。

（1）在MySQL中创建Sqoop的元数据存储数据库

create database sqoop;   
create user 'sqoop'@'%' identified by 'sqoop';   
grant all privileges on sqoop.\* to 'sqoop'@'%';   
flush privileges;

（2）配置Sqoop的元数据存储参数  
 在$SQOOP\_HOME/conf/sqoop-site.xml中添加以下的参数。

sqoop.metastore.server.location：指定元数据服务器位置，初始化建表时需要。  
sqoop.metastore.client.autoconnect.url：客户端自动连接的数据库的URL。  
sqoop.metastore.client.autoconnect.username：连接数据库的用户名。  
sqoop.metastore.client.enable.autoconnect：启用客户端自动连接数据库。  
sqoop.metastore.client.record.password：在数据库中保存密码，不需要密码即可执行sqoop job脚本。  
sqoop.metastore.client.autoconnect.password：连接数据库的密码。  
  
 <property>  
 <name>sqoop.metastore.client.enable.autoconnect</name>  
 <value>true</value>  
 <description>If true, Sqoop will connect to a local metastore  
 for job management when no other metastore arguments are  
 provided.  
 </description>  
 </property>  
  
 <property>  
 <name>sqoop.metastore.client.autoconnect.url</name>  
 <value>jdbc:mysql://192.168.137.77:3306/sqoop</value>  
 <description>The connect string to use when connecting to a  
 job-management metastore. If unspecified, uses ~/.sqoop/.  
 You can specify a different path here.  
 </description>  
 </property>  
 <property>  
 <name>sqoop.metastore.client.autoconnect.username</name>  
 <value>root</value>  
 <description>The username to bind to the metastore.  
 </description>  
 </property>  
 <property>  
 <name>sqoop.metastore.client.autoconnect.password</name>  
 <value>123456</value>  
 <description>The password to bind to the metastore.  
 </description>  
 </property>  
  
 <property>  
 <name>sqoop.metastore.client.record.password</name>  
 <value>true</value>  
 <description>If true, allow saved passwords in the metastore.  
 </description>  
 </property>  
  
 <property>  
 <name>sqoop.metastore.server.location</name>  
 <value>/usr/local/sqoop-1.4.6-cdh5.13.2/sqoop-metastore/shared.db</value>  
 <description>Path to the shared metastore database files.  
 If this is not set, it will be placed in ~/.sqoop/.  
 </description>  
 </property>  
  
 <property>  
 <name>sqoop.metastore.server.port</name>  
 <value>16000</value>  
 <description>Port that this metastore should listen on.  
 </description>  
 </property>

（3）重启Sqoop服务  
 保存配置并重启完成后，MySQL的sqoop库中有了一个名为SQOOP\_ROOT的空表。

启动：sqoop metastore &  
  
关闭：sqoop metastore --shutdown

（4）预装载SQOOP表

insert into SQOOP\_ROOT values (NULL, 'sqoop.hsqldb.job.storage.version', '0');

（5）创建初始表

sqoop job --list

此时并不会返回先前已经创建的myjob*incremental*import作业，因为此时MySQL中没有元数据信息。该命令执行完成后，MySQL的sqoop库中有了一个名为SQOOP\_SESSIONS的空表，该表存储sqoop job相关信息。

（6）将表的存储引擎修改为MYISAM

alter table SQOOP\_ROOT engine=myisam;   
alter table SQOOP\_SESSIONS engine=myisam;

因为每次执行增量抽取后都会更新last\_value值，如果使用Innodb可能引起事务锁超时错误。

#### 5.5.4.2 项目调度脚本

本项目任务包含

1、创建sqoop job  
2、创建表  
3、执行sqoop job  
4、初始化数据  
5、分析语句

本项目任务分类

1、一次性任务  
 创建job  
 创建hive表  
 执行job  
 初始化数据  
 在mysql上创建结果数据表  
2、定期任务  
 执行job  
 分析语句  
 将分析结果导出到mysql

##### 5.5.4.2.1 一次性任务脚本

以下内容对应一个个脚本，脚本文件名同注释。

以下文件需要放在同一个目录下

创建全量导入job

#1sqoop\_desc\_job\_full.sh  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job \  
--create qfbap\_user \  
 -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--username root \  
--password-file /sqoop/pwd/qfbapsqoopPWD.pwd \  
--table user \  
--delete-target-dir \  
--target-dir /hive/qfbap\_ods.db/ods\_user \  
--fields-terminated-by '\001'  
  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job \  
--create qfbap\_user\_extend \  
 -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--username root \  
--password-file /sqoop/pwd/qfbapsqoopPWD.pwd \  
--table user\_extend \  
--delete-target-dir \  
--target-dir /hive/qfbap\_ods.db/ods\_user\_extend \  
--fields-terminated-by '\001'  
  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job \  
--create qfbap\_user\_addr \  
 -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--username root \  
--password-file /sqoop/pwd/qfbapsqoopPWD.pwd \  
--table user\_addr \  
--delete-target-dir \  
--target-dir /hive/qfbap\_ods.db/ods\_user\_addr \  
--fields-terminated-by '\001'

azkaban任务配置

#1sqoop\_desc\_job\_full.job  
type=command  
command=/bin/bash 1sqoop\_desc\_job\_full.sh

创建增量导入job

#2sqoop\_desc\_job\_incr.sh  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job \  
--create qfbap\_user\_pc\_click\_log \  
 -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--username root \  
--password-file /sqoop/pwd/qfbapsqoopPWD.pwd \  
--table user\_pc\_click\_log \  
--target-dir /hive/tmp/ods\_user\_pc\_click\_log \  
--fields-terminated-by '\001' \  
--incremental append \  
--check-column log\_id \  
--last-value 0  
  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job \  
--create qfbap\_user\_app\_click\_log \  
 -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--username root \  
--password-file /sqoop/pwd/qfbapsqoopPWD.pwd \  
--table user\_app\_click\_log \  
--target-dir /hive/tmp/ods\_user\_app\_click\_log \  
--fields-terminated-by '\001' \  
--incremental append \  
--check-column log\_id \  
--last-value 0  
  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job \  
--create qfbap\_user\_order \  
 -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--username root \  
--password-file /sqoop/pwd/qfbapsqoopPWD.pwd \  
--table us\_order \  
--target-dir /hive/tmp/ods\_us\_order \  
--fields-terminated-by '\001' \  
--incremental append \  
--check-column order\_id \  
--last-value 0  
  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job \  
--create qfbap\_order\_item \  
 -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--username root \  
--password-file /sqoop/pwd/qfbapsqoopPWD.pwd \  
--table order\_item \  
--target-dir /hive/tmp/ods\_order\_item \  
--fields-terminated-by '\001' \  
--incremental append \  
--check-column order\_id \  
--last-value 0  
  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job \  
--create qfbap\_biz\_trade \  
 -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--username root \  
--password-file /sqoop/pwd/qfbapsqoopPWD.pwd \  
--table biz\_trade \  
--target-dir /hive/tmp/ods\_biz\_trade \  
--fields-terminated-by '\001' \  
--incremental append \  
--check-column order\_id \  
--last-value 0  
  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job \  
--create qfbap\_order\_delivery \  
 -- import \  
--connect jdbc:mysql://hadoop01:3306/qfbap\_ods \  
--username root \  
--password-file /sqoop/pwd/qfbapsqoopPWD.pwd \  
--table order\_delivery \  
--target-dir /hive/tmp/ods\_order\_delivery \  
--fields-terminated-by '\001' \  
--incremental append \  
--check-column order\_id \  
--last-value 0

azkaban任务配置

#2sqoop\_desc\_job\_incr.job  
type=command  
command=/bin/bash 2sqoop\_desc\_job\_incr.sh  
dependencies=1sqoop\_desc\_job\_full

创建hive的ods表的hql语句

-- 3createodstable.hql  
create database if not exists qfbap\_ods;  
create database if not exists qfbap\_tmp;  
  
CREATE TABLE if not exists qfbap\_tmp.ods\_user\_pc\_click\_log\_tmp (  
 log\_id bigint ,  
 user\_id bigint ,  
 session\_id string ,  
 cookie\_id string ,  
 visit\_time string ,  
 visit\_url string ,  
 visit\_os string ,  
 browser\_name string ,  
 visit\_ip string ,  
 province string ,  
 city string ,  
 page\_id int ,  
 goods\_id bigint ,  
 shop\_id bigint   
)  
;  
  
CREATE TABLE if not exists qfbap\_ods.ods\_user\_pc\_click\_log (  
 log\_id bigint ,  
 user\_id bigint ,  
 session\_id string ,  
 cookie\_id string ,  
 visit\_time string ,  
 visit\_url string ,  
 visit\_os string ,  
 browser\_name string ,  
 visit\_ip string ,  
 province string ,  
 city string ,  
 page\_id int ,  
 goods\_id bigint ,  
 shop\_id bigint   
)  
partitioned by (dt string)  
;  
  
CREATE TABLE if not exists qfbap\_ods.ods\_user\_extend (  
 user\_id bigint ,  
 user\_gender bigint ,  
 is\_pregnant\_woman tinyint ,  
 is\_have\_children tinyint ,  
 is\_have\_car tinyint ,  
 phone\_brand string ,  
 phone\_brand\_level string ,  
 phone\_cnt int ,  
 change\_phone\_cnt int ,  
 is\_maja tinyint ,  
 majia\_account\_cnt int ,  
 loyal\_model string ,  
 shopping\_type\_model string ,  
 weight int ,  
 height int   
);  
  
CREATE TABLE if not exists qfbap\_ods.ods\_user (  
 user\_id bigint ,  
 user\_name string ,  
 user\_gender tinyint ,  
 user\_birthday string ,  
 user\_age int ,  
 constellation string ,  
 province string ,  
 city string ,  
 city\_level tinyint ,  
 e\_mail string ,  
 op\_mail string ,  
 mobile bigint ,  
 num\_seg\_mobile int ,  
 op\_Mobile string ,  
 register\_time string ,  
 login\_ip string ,  
 login\_source string ,  
 request\_user string ,  
 total\_score decimal(18,2) ,  
 used\_score decimal(18,2) ,  
 is\_blacklist tinyint ,  
 is\_married tinyint ,  
 education string ,  
 monthly\_income decimal(18,2) ,  
 profession string ,  
 create\_date string   
) ;  
  
  
CREATE TABLE if not exists qfbap\_ods.ods\_user\_app\_click\_log (  
 log\_id bigint ,  
 user\_id bigint ,  
 imei string ,  
 log\_time string ,  
 visit\_os string ,  
 os\_version string ,  
 app\_name string ,  
 app\_version string ,  
 device\_token string ,  
 visit\_ip string ,  
 province string ,  
 city string   
)   
partitioned by (dt string)  
;  
  
CREATE TABLE if not exists qfbap\_tmp.ods\_user\_app\_click\_log\_tmp (  
 log\_id bigint ,  
 user\_id bigint ,  
 imei string ,  
 log\_time string ,  
 visit\_os string ,  
 os\_version string ,  
 app\_name string ,  
 app\_version string ,  
 device\_token string ,  
 visit\_ip string ,  
 province string ,  
 city string  
)  
;  
  
CREATE TABLE if not exists qfbap\_ods.ods\_biz\_trade (  
 trade\_id bigint ,  
 order\_id bigint ,  
 user\_id bigint ,  
 amount decimal(18,2) ,  
 trade\_type tinyint ,  
 trade\_time string   
)   
partitioned by (dt string)  
;  
  
CREATE TABLE if not exists qfbap\_ods.ods\_order\_delivery (  
 order\_id bigint ,  
 order\_no string ,  
 consignee string ,  
 area\_id bigint ,  
 area\_name string ,  
 address string ,  
 mobile bigint ,  
 phone string ,  
 coupon\_id bigint ,  
 coupon\_money decimal(18,2) ,  
 carriage\_money decimal(18,2) ,  
 create\_time string ,  
 update\_time string ,  
 addr\_id bigint   
)   
partitioned by (dt string)  
;  
  
CREATE TABLE if not exists qfbap\_ods.ods\_order\_item (  
 user\_id bigint ,  
 order\_id bigint ,  
 order\_no string ,  
 goods\_id bigint ,  
 goods\_no string ,  
 goods\_name string ,  
 goods\_amount int ,  
 shop\_id bigint ,  
 shop\_name string ,  
 curr\_price decimal(18,2) ,  
 market\_price decimal(18,2) ,  
 discount decimal(18,2) ,  
 cost\_price decimal(18,2) ,  
 first\_cart bigint ,  
 first\_cart\_name string ,  
 second\_cart bigint ,  
 second\_cart\_name string ,  
 third\_cart bigint ,  
 third\_cart\_name string ,  
 goods\_desc string   
)   
partitioned by (dt string)  
;  
  
CREATE TABLE if not exists qfbap\_ods.ods\_us\_order (  
 order\_id bigint ,  
 order\_no string ,  
 order\_date string ,  
 user\_id bigint ,  
 user\_name string ,  
 order\_money decimal(18,2) ,  
 order\_type int ,  
 order\_status int ,  
 pay\_status int ,  
 pay\_type int ,  
 order\_source string ,  
 update\_time string   
)  
partitioned by (dt string)  
;  
  
CREATE TABLE if not exists qfbap\_ods.ods\_user\_addr (  
 user\_id bigint ,  
 order\_addr string ,  
 user\_order\_flag tinyint ,  
 addr\_id bigint ,  
 arear\_id int   
)  
;  
  
CREATE TABLE if not exists qfbap\_tmp.ods\_biz\_trade\_tmp (  
 trade\_id bigint ,  
 order\_id bigint ,  
 user\_id bigint ,  
 amount decimal(18,2) ,  
 trade\_type tinyint ,  
 trade\_time string   
)   
;  
  
CREATE TABLE if not exists qfbap\_tmp.ods\_order\_delivery\_tmp (  
 order\_id bigint ,  
 order\_no string ,  
 consignee string ,  
 area\_id bigint ,  
 area\_name string ,  
 address string ,  
 mobile bigint ,  
 phone string ,  
 coupon\_id bigint ,  
 coupon\_money decimal(18,2) ,  
 carriage\_money decimal(18,2) ,  
 create\_time string ,  
 update\_time string ,  
 addr\_id bigint   
)   
;  
  
CREATE TABLE if not exists qfbap\_tmp.ods\_order\_item\_tmp (  
 user\_id bigint ,  
 order\_id bigint ,  
 order\_no string ,  
 goods\_id bigint ,  
 goods\_no string ,  
 goods\_name string ,  
 goods\_amount int ,  
 shop\_id bigint ,  
 shop\_name string ,  
 curr\_price decimal(18,2) ,  
 market\_price decimal(18,2) ,  
 discount decimal(18,2) ,  
 cost\_price decimal(18,2) ,  
 first\_cart bigint ,  
 first\_cart\_name string ,  
 second\_cart bigint ,  
 second\_cart\_name string ,  
 third\_cart bigint ,  
 third\_cart\_name string ,  
 goods\_desc string   
)   
;  
  
CREATE TABLE if not exists qfbap\_tmp.ods\_us\_order\_tmp (  
 order\_id bigint ,  
 order\_no string ,  
 order\_date string ,  
 user\_id bigint ,  
 user\_name string ,  
 order\_money decimal(18,2) ,  
 order\_type int ,  
 order\_status int ,  
 pay\_status int ,  
 pay\_type int ,  
 order\_source string ,  
 update\_time string   
)  
;

脚本

#3createodstable.sh  
/usr/local/hive-1.1.0-cdh5.13.2/bin/hive -f 3createodstable.hql

azkaban任务配置

#3createodstable.job  
type=command  
command=/bin/bash 3createodstable.sh  
dependencies=2sqoop\_desc\_job\_incr

导入全量数据

# 4sqoop\_exec\_job\_full.sh  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_user;  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_user\_addr;

azkaban任务调度

#4sqoop\_exec\_job\_full.job  
type=command  
command=/bin/bash 4sqoop\_exec\_job\_full.sh  
dependencies=3createodstable

导入增量数据（业务数据--订单、日志）

# 5sqoop\_exec\_job\_incr.sh  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_user\_order;   
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_order\_item;  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_biz\_trade;  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_order\_delivery;  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_user\_pc\_click\_log;  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_user\_app\_click\_log;

azkaban任务调度

#5sqoop\_exec\_job\_incr.job  
type=command  
command=/bin/bash 5sqoop\_exec\_job\_incr.sh  
dependencies=4sqoop\_exec\_job\_full

初始化用户拉链表

-- 6init\_user.hql  
create database if not exists qfbap\_dwd;  
  
CREATE TABLE qfbap\_dwd.dwd\_user (  
 user\_id bigint ,  
 user\_name string ,  
 user\_gender tinyint ,  
 user\_birthday string ,  
 user\_age int ,  
 constellation string ,  
 province string ,  
 city string ,  
 city\_level tinyint ,  
 e\_mail string ,  
 op\_mail string ,  
 mobile bigint ,  
 num\_seg\_mobile int ,  
 op\_Mobile string ,  
 register\_time string ,  
 login\_ip string ,  
 login\_source string ,  
 request\_user string ,  
 total\_score decimal(18,2) ,  
 used\_score decimal(18,2) ,  
 is\_blacklist tinyint ,  
 is\_married tinyint ,  
 education string ,  
 monthly\_income decimal(18,2) ,  
 profession string ,  
 create\_date string ,  
 start\_date string,  
 end\_date string  
)   
partitioned by (state string,dt string)  
;  
  
insert overwrite table qfbap\_dwd.dwd\_user partition(state = 'active',dt='9999-12-31')  
select   
nu.user\_id,  
nu.user\_name,  
nu.user\_gender,  
nu.user\_birthday,  
nu.user\_age,  
nu.constellation,  
nu.province,  
nu.city,  
nu.city\_level,  
nu.e\_mail,  
nu.op\_mail,  
nu.mobile,  
nu.num\_seg\_mobile,  
nu.op\_Mobile,  
nu.register\_time,  
nu.login\_ip,  
nu.login\_source,  
nu.request\_user,  
nu.total\_score,  
nu.used\_score,  
nu.is\_blacklist,  
nu.is\_married,  
nu.education,  
nu.monthly\_income,  
nu.profession,  
nu.create\_date,  
from\_unixtime(to\_unix\_timestamp(nu.register\_time),'yyyy-MM-dd') start\_date,  
'9999-12-31' end\_date  
from qfbap\_ods.ods\_user nu  
;

调度脚本

/usr/local/hive-1.1.0-cdh5.13.2/bin/hive -f 6init\_user.hql

azkaban任务配置

#6init\_user.job  
type=command  
command=/bin/bash 6init\_user.sh  
dependencies=5sqoop\_exec\_job\_incr

初始化用户访问历史数据

-- 7inituserlog.hql  
load data inpath '/hive/tmp/ods\_user\_pc\_click\_log/\*' into table qfbap\_tmp.ods\_user\_pc\_click\_log\_tmp;  
load data inpath '/hive/tmp/ods\_user\_app\_click\_log/\*' into table qfbap\_tmp.ods\_user\_app\_click\_log\_tmp;  
  
create database if not exists qfbap\_dwd;  
  
CREATE TABLE if not exists qfbap\_dwd.dwd\_user\_pc\_pv (  
log\_id bigint ,  
user\_id bigint ,  
session\_id string ,  
cookie\_id string ,  
in\_time string,  
out\_time string,  
stay\_time bigint,  
pv bigint,  
visit\_os string ,  
browser\_name string ,  
visit\_ip string ,  
province string ,  
city string ,  
dw\_date string  
)  
partitioned by (dt string)  
;  
  
CREATE external TABLE if not exists qfbap\_dwd.dwd\_user\_app\_pv (  
 log\_id bigint ,  
 user\_id bigint ,  
 imei string ,  
 log\_time string ,  
 log\_hour string,  
 visit\_os string ,  
 os\_version string ,  
 app\_name string ,  
 app\_version string ,  
 device\_token string ,  
 visit\_ip string ,  
 province string ,  
 city string ,  
 dw\_date string  
)   
partitioned by (dt string)  
;  
  
set hive.exec.dynamic.partition=true;  
set hive.exec.dynamic.partition.mode = nostrict;  
set hive.exec.max.dynamic.partitions = 1000;  
set hive.exec.max.dynamic.partitions.pernode = 1000;  
  
insert overwrite table qfbap\_dwd.dwd\_user\_pc\_pv partition(dt)  
select  
max(log\_id) log\_id,  
user\_id,  
session\_id,  
cookie\_id,  
min(visit\_time) in\_time,  
max(visit\_time) out\_time,  
unix\_timestamp(max(visit\_time)) - unix\_timestamp(min(visit\_time)) stay\_time,  
count(1) pv,  
visit\_os,  
browser\_name,  
visit\_ip,  
province,  
city,  
current\_timestamp() dw\_date,  
from\_unixtime(to\_unix\_timestamp(min(visit\_time)),'yyyy-MM-dd') dt  
from qfbap\_tmp.ods\_user\_pc\_click\_log\_tmp  
group by   
user\_id,  
session\_id,  
cookie\_id,  
visit\_os,  
browser\_name,  
visit\_ip,  
province,  
city  
;  
  
insert overwrite table qfbap\_dwd.dwd\_user\_app\_pv partition (dt)  
select  
log\_id,  
user\_id,  
imei,  
log\_time,  
hour(log\_time) log\_hour,  
visit\_os,  
os\_version,  
app\_name,  
app\_version,  
device\_token,  
visit\_ip,  
province,  
city,  
current\_timestamp() dw\_date,  
from\_unixtime(to\_unix\_timestamp(log\_time),'yyyy-MM-dd') dt  
from qfbap\_tmp.ods\_user\_app\_click\_log\_tmp  
;

调度脚本

# 7inituserlog.sh  
/usr/local/hive-1.1.0-cdh5.13.2/bin/hive -f 7inituserlog.hql

azkaban任务配置

#7inituserlog.job  
type=command  
command=/bin/bash 7inituserlog.sh  
dependencies=6init\_user

初始化用户订单历史数据

-- 8inituserorder.hql  
load data inpath '/hive/tmp/ods\_us\_order/\*' into table qfbap\_tmp.ods\_us\_order\_tmp;  
load data inpath '/hive/tmp/ods\_order\_item/\*' into table qfbap\_tmp.ods\_order\_item\_tmp;  
load data inpath '/hive/tmp/ods\_biz\_trade/\*' into table qfbap\_tmp.ods\_biz\_trade\_tmp;  
load data inpath '/hive/tmp/ods\_order\_delivery/\*' into table qfbap\_tmp.ods\_order\_delivery\_tmp;  
  
CREATE TABLE if not exists qfbap\_dwd.dwd\_us\_order (  
 order\_id bigint ,  
 order\_no string ,  
 order\_date string ,  
 order\_day string,  
 user\_id bigint ,  
 user\_name string ,  
 order\_money decimal(18,2) ,  
 order\_type int ,  
 order\_status int ,  
 pay\_status int ,  
 pay\_type int ,  
 order\_source string ,  
 update\_time string ,  
 goods\_amounts int,  
 goods\_moneys decimal(18,2),  
 trade\_type tinyint,  
 trade\_time string,  
 area\_name string,  
 user\_order\_flag tinyint,  
 dw\_date string  
)  
partitioned by (dt string)  
;  
  
set hive.exec.dynamic.partition=true;  
set hive.exec.dynamic.partition.mode = nostrict;  
set hive.exec.max.dynamic.partitions = 1000;  
set hive.exec.max.dynamic.partitions.pernode = 1000;  
  
insert overwrite table qfbap\_dwd.dwd\_us\_order partition(dt)  
select  
od.order\_id,  
od.order\_no,  
od.order\_date,  
from\_unixtime(to\_unix\_timestamp(od.order\_date),'yyyy-MM-dd') order\_day,  
od.user\_id,  
od.user\_name,  
od.order\_money,  
od.order\_type,  
od.order\_status,  
od.pay\_status,  
od.pay\_type,  
od.order\_source,  
od.update\_time,  
oi.goods\_amounts,  
oi.goods\_moneys,  
ot.trade\_type,  
ot.trade\_time,  
ood.area\_name,  
oua.user\_order\_flag,  
current\_timestamp() dw\_date,  
from\_unixtime(to\_unix\_timestamp(od.order\_date),'yyyy-MM-dd') dt  
from qfbap\_tmp.ods\_us\_order\_tmp od  
left join   
(  
select   
order\_id,  
sum(goods\_amount) goods\_amounts,  
sum(goods\_amount \* curr\_price) goods\_moneys   
from qfbap\_tmp.ods\_order\_item\_tmp  
group by order\_id  
) oi   
on oi.order\_id = od.order\_id  
left join qfbap\_tmp.ods\_biz\_trade\_tmp ot   
on ot.order\_id = od.order\_id  
left join qfbap\_tmp.ods\_order\_delivery\_tmp ood  
on ood.order\_id = od.order\_id  
left join qfbap\_ods.ods\_user\_addr oua  
on oua.addr\_id = ood.addr\_id  
;  
  
create database if not exists qfbap\_dws;  
create table if not exists qfbap\_dws.dws\_user\_order\_all  
(  
 user\_id bigint,  
 type string,  
 content string,  
 cnt bigint,  
 rn bigint,  
 dw\_date string  
)  
partitioned by (dt string)  
stored as orc  
;  
  
insert overwrite table qfbap\_dws.dws\_user\_order\_all partition(dt = '${hiveconf:params}')  
select  
user\_id,  
type,  
content,  
cnt,  
row\_number() over(distribute by user\_id,type sort by cnt desc) rn,  
current\_timestamp() dw\_date  
from  
(  
select  
user\_id,  
1 type,   
area\_name content,  
count(1) cnt  
from qfbap\_dwd.dwd\_us\_order  
group by   
user\_id,  
area\_name  
union all  
select  
user\_id,  
2 type,   
pay\_type content,  
count(1) cnt  
from qfbap\_dwd.dwd\_us\_order  
group by   
user\_id,  
pay\_type  
) a  
;  
  
create database if not exists qfbap\_dm;  
  
CREATE TABLE if not exists `qfbap\_dm.dm\_user\_order`(  
 `user\_id` bigint,   
 `first\_order\_date` string,   
 `last\_order\_date` string,   
 `first\_diff\_day` int,   
 `last\_diff\_day` int,   
 `day30\_rr\_cnt` bigint,   
 `day30\_rr\_moneys` decimal(28,2),   
 `day60\_rr\_cnt` bigint,   
 `day60\_rr\_moneys` decimal(28,2),   
 `day90\_rr\_cnt` bigint,   
 `day90\_rr\_moneys` decimal(28,2),   
 `day30\_cnt` bigint,   
 `day30\_moneys` decimal(28,2),   
 `day60\_cnt` bigint,   
 `day60\_moneys` decimal(28,2),   
 `day90\_cnt` bigint,   
 `day90\_moneys` decimal(28,2),   
 `day90\_avgprice` decimal(38,16),   
 `max\_money` decimal(18,2),   
 `min\_money` decimal(18,2),   
 `all\_rr\_cnt` bigint,   
 `all\_rr\_moneys` decimal(28,2),   
 all\_rr\_avgprice decimal(28,2),  
 `all\_th\_cnt` bigint,   
 `all\_th\_moneys` decimal(28,2),   
 `all\_js\_cnt` bigint,   
 `all\_js\_moneys` decimal(28,2),   
 `max\_trade\_time` string,   
 `school\_cnt` bigint,   
 `company\_cnt` bigint,   
 `family\_cnt` bigint,   
 `hr025\_cnt` bigint,   
 `hr6212\_cnt` bigint,   
 `hr13215\_cnt` bigint,   
 `hr16220\_cnt` bigint,   
 `hr21223\_cnt` bigint,   
 `max\_area\_name` string,   
 `max\_pay\_type` string  
)  
partitioned by (dt string)  
;  
  
insert overwrite table qfbap\_dm.dm\_user\_order partition(dt = '${hiveconf:params}')  
select  
user\_id,  
max(first\_order\_date) first\_order\_date,  
max(last\_order\_date) last\_order\_date,  
max(first\_diff\_day) first\_diff\_day,  
max(last\_diff\_day) last\_diff\_day,  
max(day30\_rr\_cnt) day30\_rr\_cnt,  
max(day30\_rr\_moneys) day30\_rr\_moneys,  
max(day60\_rr\_cnt) day60\_rr\_cnt,  
max(day60\_rr\_moneys) day60\_rr\_moneys,  
max(day90\_rr\_cnt) day90\_rr\_cnt,  
max(day90\_rr\_moneys) day90\_rr\_moneys,  
max(day30\_cnt) day30\_cnt,  
max(day30\_moneys) day30\_moneys,  
max(day60\_cnt) day60\_cnt,  
max(day60\_moneys) day60\_moneys,  
max(day90\_cnt) day90\_cnt,  
max(day90\_moneys) day90\_moneys,  
max(day90\_avgprice) day90\_avgprice,  
max(max\_money) max\_money,  
min(min\_money) min\_money,  
sum(all\_rr\_cnt) all\_rr\_cnt,  
sum(all\_rr\_moneys) all\_rr\_moneys,  
sum(all\_rr\_moneys) / sum(all\_rr\_cnt) all\_rr\_avgprice,  
sum(all\_th\_cnt) all\_th\_cnt,  
sum(all\_th\_moneys) all\_th\_moneys,  
sum(all\_js\_cnt) all\_js\_cnt,  
sum(all\_js\_moneys) all\_js\_moneys,  
max(max\_trade\_time) max\_trade\_time,  
sum(school\_cnt) school\_cnt,  
sum(company\_cnt) company\_cnt,  
sum(family\_cnt) family\_cnt,  
sum(hr025\_cnt) hr025\_cnt,  
sum(hr6212\_cnt) hr6212\_cnt,  
sum(hr13215\_cnt) hr13215\_cnt,  
sum(hr16220\_cnt) hr16220\_cnt,  
sum(hr21223\_cnt) hr21223\_cnt,  
max(max\_area\_name) max\_area\_name,  
max(max\_pay\_type) max\_pay\_type  
from  
(  
select  
user\_id,  
max(case when uo.rn\_asc = 1 then uo.order\_date end) first\_order\_date,  
max(case when uo.rn\_desc = 1 then uo.order\_date end) last\_order\_date,  
datediff(current\_date(),max(case when uo.rn\_asc = 1 then uo.order\_date end)) first\_diff\_day,  
datediff(current\_date(),max(case when uo.rn\_desc = 1 then uo.order\_date end)) last\_diff\_day,  
count(case when uo.day30 = 1 and uo.trade\_type = 1 then uo.order\_id end) day30\_rr\_cnt,  
sum(case when uo.day30 = 1 and uo.trade\_type = 1 then uo.order\_money else 0 end) day30\_rr\_moneys,  
count(case when uo.day60 = 1 and uo.trade\_type = 1 then uo.order\_id end) day60\_rr\_cnt,  
sum(case when uo.day60 = 1 and uo.trade\_type = 1 then uo.order\_money else 0 end) day60\_rr\_moneys,  
count(case when uo.day90 = 1 and uo.trade\_type = 1 then uo.order\_id end) day90\_rr\_cnt,  
sum(case when uo.day90 = 1 and uo.trade\_type = 1 then uo.order\_money else 0 end) day90\_rr\_moneys,  
count(case when uo.day30 = 1 then uo.order\_id end) day30\_cnt,  
sum(case when uo.day30 = 1 then uo.order\_money else 0 end) day30\_moneys,  
count(case when uo.day60 = 1 then uo.order\_id end) day60\_cnt,  
sum(case when uo.day60 = 1 then uo.order\_money else 0 end) day60\_moneys,  
count(case when uo.day90 = 1 then uo.order\_id end) day90\_cnt,  
sum(case when uo.day90 = 1 then uo.order\_money else 0 end) day90\_moneys,  
sum(case when uo.day90 = 1 then uo.order\_money else 0 end) / count(case when uo.day90 = 1 then uo.order\_id end) day90\_avgprice,  
max(uo.order\_money) max\_money,  
min(uo.order\_money) min\_money,  
count(case when uo.trade\_type = 1 then uo.order\_id end) all\_rr\_cnt,  
sum(case when uo.trade\_type = 1 then uo.order\_money else 0 end) all\_rr\_moneys,  
count(case when uo.trade\_type = -1 then uo.order\_id end) all\_th\_cnt,  
sum(case when uo.trade\_type = -1 then uo.order\_money else 0 end) all\_th\_moneys,  
sum(case when uo.trade\_type = -2 then uo.goods\_amounts else 0 end) all\_js\_cnt,  
sum(case when uo.trade\_type = -2 then uo.goods\_moneys else 0 end) all\_js\_moneys,  
max(case when uo.trade\_type = -1 then uo.trade\_time end) max\_trade\_time,  
count(case when uo.user\_order\_flag = 1 then uo.order\_id end) school\_cnt,  
count(case when uo.user\_order\_flag = 2 then uo.order\_id end) company\_cnt,  
count(case when uo.user\_order\_flag = 3 then uo.order\_id end) family\_cnt,  
count(case when uo.hr025 = 1 then uo.order\_id end) hr025\_cnt,  
count(case when uo.hr6212 = 1 then uo.order\_id end) hr6212\_cnt,  
count(case when uo.hr13215 = 1 then uo.order\_id end) hr13215\_cnt,  
count(case when uo.hr16220 = 1 then uo.order\_id end) hr16220\_cnt,  
count(case when uo.hr21223 = 1 then uo.order\_id end) hr21223\_cnt,  
null as max\_area\_name,  
null as max\_pay\_type  
from   
(  
select   
od.\*,  
row\_number() over(distribute by user\_id sort by order\_date asc) rn\_asc,  
row\_number() over(distribute by user\_id sort by order\_date desc) rn\_desc,  
case when od.order\_day >= date\_sub('${hiveconf:params}',29) and od.order\_day <= '${hiveconf:params}' then 1 end day30,  
case when od.order\_day >= date\_sub('${hiveconf:params}',59) and od.order\_day <= '${hiveconf:params}' then 1 end day60,  
case when od.order\_day >= date\_sub('${hiveconf:params}',89) and od.order\_day <= '${hiveconf:params}' then 1 end day90,  
case when hour(od.order\_date) between 0 and 5 then 1 end hr025,  
case when hour(od.order\_date) between 6 and 12 then 1 end hr6212,  
case when hour(od.order\_date) between 13 and 15 then 1 end hr13215,  
case when hour(od.order\_date) between 16 and 20 then 1 end hr16220,  
case when hour(od.order\_date) between 21 and 23 then 1 end hr21223  
from qfbap\_dwd.dwd\_us\_order od  
) uo  
group by   
user\_id  
union all  
select  
user\_id,  
null first\_order\_date,  
null last\_order\_date,  
null first\_diff\_day,  
null last\_diff\_day,  
null day30\_rr\_cnt,  
null day30\_rr\_moneys,  
null day60\_rr\_cnt,  
null day60\_rr\_moneys,  
null day90\_rr\_cnt,  
null day90\_rr\_moneys,  
null day30\_cnt,  
null day30\_moneys,  
null day60\_cnt,  
null day60\_moneys,  
null day90\_cnt,  
null day90\_moneys,  
null day90\_avgprice,  
null max\_money,  
null min\_money,  
0 all\_rr\_cnt,  
0 all\_rr\_moneys,  
0 all\_th\_cnt,  
0 all\_th\_moneys,  
0 all\_js\_cnt,  
0 all\_js\_moneys,  
null max\_trade\_time,  
0 school\_cnt,  
0 company\_cnt,  
0 family\_cnt,  
0 hr025\_cnt,  
0 hr6212\_cnt,  
0 hr13215\_cnt,  
0 hr16220\_cnt,  
0 hr21223\_cnt,  
max(case when type = 1 and rn = 1 then content end) max\_area\_name,  
max(case when type = 2 and rn = 1 then content end) max\_pay\_type  
from qfbap\_dws.dws\_user\_order\_all   
where dt = '${hiveconf:params}'  
group by   
user\_id  
) b  
group by   
user\_id  
;

调度脚本

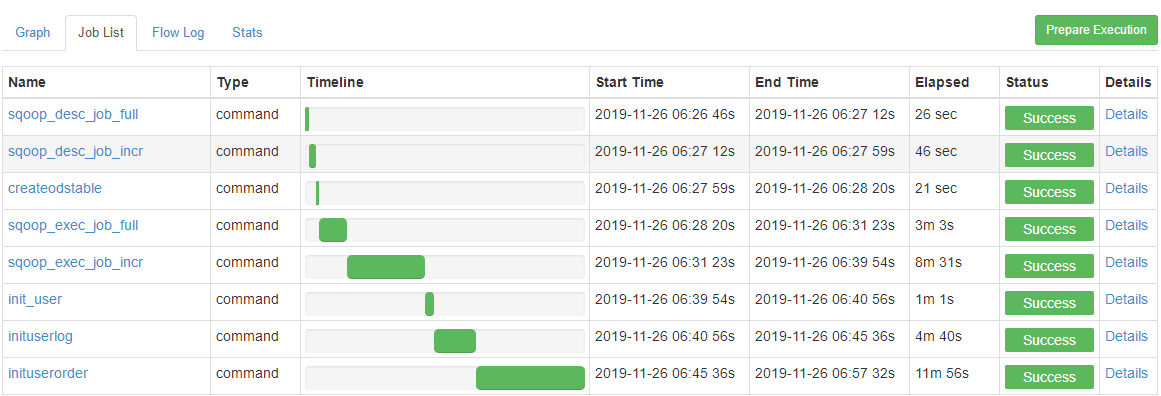
# 8inituserorder.sh  
args=$1  
dt=  
  
if [ ${#args} == 0 ]  
then  
dt=`date -d "2 days ago" "+%Y-%m-%d"`  
else  
dt=${#args}  
fi  
  
/usr/local/hive-1.1.0-cdh5.13.2/bin/hive --hiveconf params=${dt} -f 8inituserorder.hql

azkaban任务配置

#8inituserorder.job  
type=command  
command=/bin/bash 8inituserorder.sh  
dependencies=7inituserlog

完成后将以上所有文件打包到一个zip文件并上传到azkaban的任务

初始化调度任务运行如下



##### 5.5.4.2.2 定期任务

注意参数

导入全量数据

# 1sqoop\_exec\_job\_full.sh  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_user;  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_user\_extend;  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_user\_addr;

azkaban任务配置

#1sqoop\_exec\_job\_full.job  
type=command  
command=/bin/bash 1sqoop\_exec\_job\_full.sh

导入增量数据

# 2sqoop\_exec\_job\_incr.sh  
args=$1  
dt=  
  
if [ ${#args} == 0 ]  
then  
dt=`date -d "1 days ago" "+%Y-%m-%d"`  
else  
dt=${#args}  
fi  
  
  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_user\_order;   
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_order\_item;   
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_biz\_trade;   
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_order\_delivery;  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_user\_pc\_click\_log;  
/usr/local/sqoop-1.4.6-cdh5.13.2/bin/sqoop job --exec qfbap\_user\_app\_click\_log;  
  
/usr/local/hive-1.1.0-cdh5.13.2/bin/hive -e "load data inpath '/hive/tmp/ods\_us\_order/\*' overwrite into table qfbap\_ods.ods\_us\_order partition (dt = '${dt}')"  
/usr/local/hive-1.1.0-cdh5.13.2/bin/hive -e "load data inpath '/hive/tmp/ods\_order\_item/\*' overwrite into table qfbap\_ods.ods\_order\_item partition (dt = '${dt}')"  
/usr/local/hive-1.1.0-cdh5.13.2/bin/hive -e "load data inpath '/hive/tmp/ods\_biz\_trade/\*' overwrite into table qfbap\_ods.ods\_biz\_trade partition (dt = '${dt}')"  
/usr/local/hive-1.1.0-cdh5.13.2/bin/hive -e "load data inpath '/hive/tmp/ods\_order\_delivery/\*' overwrite into table qfbap\_ods.ods\_order\_delivery partition (dt = '${dt}')"  
/usr/local/hive-1.1.0-cdh5.13.2/bin/hive -e "load data inpath '/hive/tmp/ods\_user\_pc\_click\_log/\*' overwrite into table qfbap\_ods.ods\_user\_pc\_click\_log partition (dt = '${dt}')"  
/usr/local/hive-1.1.0-cdh5.13.2/bin/hive -e "load data inpath '/hive/tmp/ods\_user\_app\_click\_log/\*' overwrite into table qfbap\_ods.ods\_user\_app\_click\_log partition (dt = '${dt}')"

azkaban任务配置

#2sqoop\_exec\_job\_incr.job  
type=command  
command=/bin/bash 2sqoop\_exec\_job\_incr.sh  
dependencies=1sqoop\_exec\_job\_full

用户基本信息宽表指标分析

-- 3user\_basic.hql  
create database if not exists qfbap\_dwd;  
CREATE TABLE if not exists qfbap\_dwd.dwd\_user (  
 user\_id bigint ,  
 user\_name string ,  
 user\_gender tinyint ,  
 user\_birthday string ,  
 user\_age int ,  
 constellation string ,  
 province string ,  
 city string ,  
 city\_level tinyint ,  
 e\_mail string ,  
 op\_mail string ,  
 mobile bigint ,  
 num\_seg\_mobile int ,  
 op\_Mobile string ,  
 register\_time string ,  
 login\_ip string ,  
 login\_source string ,  
 request\_user string ,  
 total\_score decimal(18,2) ,  
 used\_score decimal(18,2) ,  
 is\_blacklist tinyint ,  
 is\_married tinyint ,  
 education string ,  
 monthly\_income decimal(18,2) ,  
 profession string ,  
 create\_date string ,  
 start\_date string,  
 end\_date string  
)   
partitioned by (state string,dt string)  
;  
  
insert into table qfbap\_dwd.dwd\_user partition(state = 'expired',dt='${hiveconf:params}')  
select   
ou.user\_id,  
ou.user\_name,  
ou.user\_gender,  
ou.user\_birthday,  
ou.user\_age,  
ou.constellation,  
ou.province,  
ou.city,  
ou.city\_level,  
ou.e\_mail,  
ou.op\_mail,  
ou.mobile,  
ou.num\_seg\_mobile,  
ou.op\_Mobile,  
ou.register\_time,  
ou.login\_ip,  
ou.login\_source,  
ou.request\_user,  
ou.total\_score,  
ou.used\_score,  
ou.is\_blacklist,  
ou.is\_married,  
ou.education,  
ou.monthly\_income,  
ou.profession,  
ou.create\_date,  
ou.start\_date,  
'${hiveconf:params}' end\_date  
from qfbap\_ods.ods\_user nu  
right join qfbap\_dwd.dwd\_user ou   
on   
 ou.state = 'active'  
 and ou.dt = '9999-12-31'  
 and nu.user\_id = ou.user\_id   
where nu.user\_id is null  
union all  
select   
ou.user\_id,  
ou.user\_name,  
ou.user\_gender,  
ou.user\_birthday,  
ou.user\_age,  
ou.constellation,  
ou.province,  
ou.city,  
ou.city\_level,  
ou.e\_mail,  
ou.op\_mail,  
ou.mobile,  
ou.num\_seg\_mobile,  
ou.op\_Mobile,  
ou.register\_time,  
ou.login\_ip,  
ou.login\_source,  
ou.request\_user,  
ou.total\_score,  
ou.used\_score,  
ou.is\_blacklist,  
ou.is\_married,  
ou.education,  
ou.monthly\_income,  
ou.profession,  
ou.create\_date,  
ou.start\_date,  
'${hiveconf:params}' end\_date  
from qfbap\_ods.ods\_user nu  
join qfbap\_dwd.dwd\_user ou   
on ou.state = 'active'  
 and ou.dt = '9999-12-31'  
 and nu.user\_id = ou.user\_id   
where nu.user\_name <> ou.user\_name  
;  
  
insert overwrite table qfbap\_dwd.dwd\_user partition(state = 'active',dt='9999-12-31')  
select   
nu.user\_id,  
nu.user\_name,  
nu.user\_gender,  
nu.user\_birthday,  
nu.user\_age,  
nu.constellation,  
nu.province,  
nu.city,  
nu.city\_level,  
nu.e\_mail,  
nu.op\_mail,  
nu.mobile,  
nu.num\_seg\_mobile,  
nu.op\_Mobile,  
nu.register\_time,  
nu.login\_ip,  
nu.login\_source,  
nu.request\_user,  
nu.total\_score,  
nu.used\_score,  
nu.is\_blacklist,  
nu.is\_married,  
nu.education,  
nu.monthly\_income,  
nu.profession,  
nu.create\_date,  
date\_add('${hiveconf:params}',1) start\_date,  
'9999-12-31' end\_date  
from qfbap\_ods.ods\_user nu  
join qfbap\_dwd.dwd\_user ou   
on   
 ou.state = 'active'  
 and ou.dt = '9999-12-31'  
 and nu.user\_id = ou.user\_id   
where nu.user\_name <> ou.user\_name  
union all  
select   
nu.user\_id,  
nu.user\_name,  
nu.user\_gender,  
nu.user\_birthday,  
nu.user\_age,  
nu.constellation,  
nu.province,  
nu.city,  
nu.city\_level,  
nu.e\_mail,  
nu.op\_mail,  
nu.mobile,  
nu.num\_seg\_mobile,  
nu.op\_Mobile,  
nu.register\_time,  
nu.login\_ip,  
nu.login\_source,  
nu.request\_user,  
nu.total\_score,  
nu.used\_score,  
nu.is\_blacklist,  
nu.is\_married,  
nu.education,  
nu.monthly\_income,  
nu.profession,  
nu.create\_date,  
date\_add('${hiveconf:params}',1) start\_date,  
'9999-12-31' end\_date  
from qfbap\_ods.ods\_user nu  
left join qfbap\_dwd.dwd\_user ou   
on   
 ou.state = 'active'  
 and ou.dt = '9999-12-31'  
 and nu.user\_id = ou.user\_id   
where ou.user\_id is null  
union all  
select   
ou.user\_id,  
ou.user\_name,  
ou.user\_gender,  
ou.user\_birthday,  
ou.user\_age,  
ou.constellation,  
ou.province,  
ou.city,  
ou.city\_level,  
ou.e\_mail,  
ou.op\_mail,  
ou.mobile,  
ou.num\_seg\_mobile,  
ou.op\_Mobile,  
ou.register\_time,  
ou.login\_ip,  
ou.login\_source,  
ou.request\_user,  
ou.total\_score,  
ou.used\_score,  
ou.is\_blacklist,  
ou.is\_married,  
ou.education,  
ou.monthly\_income,  
ou.profession,  
ou.create\_date,  
ou.start\_date,  
ou.end\_date  
from qfbap\_ods.ods\_user nu  
join qfbap\_dwd.dwd\_user ou   
on   
 ou.state = 'active'   
 and ou.dt = '9999-12-31'  
 and nu.user\_id = ou.user\_id   
where nu.user\_name = ou.user\_name  
;  
  
CREATE TABLE if not exists qfbap\_dwd.dwd\_user\_extend (  
 user\_id bigint ,  
 user\_gender bigint ,  
 is\_pregnant\_woman tinyint ,  
 is\_have\_children tinyint ,  
 is\_have\_car tinyint ,  
 phone\_brand string ,  
 phone\_brand\_level string ,  
 phone\_cnt int ,  
 change\_phone\_cnt int ,  
 is\_maja tinyint ,  
 majia\_account\_cnt int ,  
 loyal\_model string ,  
 shopping\_type\_model string ,  
 weight int ,  
 height int ,  
 dw\_date string  
)  
;  
  
insert overwrite table qfbap\_dwd.dwd\_user\_extend  
select   
user\_id,  
user\_gender,  
is\_pregnant\_woman,  
is\_have\_children,  
is\_have\_car,  
phone\_brand,  
phone\_brand\_level,  
phone\_cnt,  
change\_phone\_cnt,  
is\_maja,  
majia\_account\_cnt,  
loyal\_model,  
shopping\_type\_model,  
weight,  
height,  
current\_timestamp() dw\_date  
from qfbap\_ods.ods\_user\_extend  
;  
  
create database if not exists qfbap\_dm;  
  
CREATE TABLE if not exists qfbap\_dm.dm\_user(  
 user\_id bigint,   
 user\_name string,   
 user\_gender tinyint,   
 user\_birthday string,   
 user\_age int,   
 constellation string,   
 province string,   
 city string,   
 city\_level tinyint,   
 e\_mail string,   
 op\_mail string,   
 mobile bigint,   
 num\_seg\_mobile int,   
 op\_mobile string,   
 register\_time string,   
 login\_ip string,   
 login\_source string,   
 request\_user string,   
 total\_score decimal(18,2),   
 used\_score decimal(18,2),   
 is\_blacklist tinyint,   
 is\_married tinyint,   
 education string,   
 monthly\_income decimal(18,2),   
 profession string,   
 create\_date string,   
 is\_pregnant\_woman tinyint,   
 is\_have\_children tinyint,   
 is\_have\_car tinyint,   
 phone\_brand string,   
 phone\_brand\_level string,   
 phone\_cnt int,   
 change\_phone\_cnt int,   
 is\_maja tinyint,   
 majia\_account\_cnt int,   
 loyal\_model string,   
 shopping\_type\_model string,   
 weight int,   
 height int)  
;  
  
insert overwrite table qfbap\_dm.dm\_user  
select   
us.user\_id,  
us.user\_name,  
us.user\_gender,  
us.user\_birthday,  
us.user\_age,  
us.constellation,  
us.province,  
us.city,  
us.city\_level,  
us.e\_mail,  
us.op\_mail,  
us.mobile,  
us.num\_seg\_mobile,  
us.op\_Mobile,  
us.register\_time,  
us.login\_ip,  
us.login\_source,  
us.request\_user,  
us.total\_score,  
us.used\_score,  
us.is\_blacklist,  
us.is\_married,  
us.education,  
us.monthly\_income,  
us.profession,  
us.create\_date,  
ue.is\_pregnant\_woman,  
ue.is\_have\_children,  
ue.is\_have\_car,  
ue.phone\_brand,  
ue.phone\_brand\_level,  
ue.phone\_cnt,  
ue.change\_phone\_cnt,  
ue.is\_maja,  
ue.majia\_account\_cnt,  
ue.loyal\_model,  
ue.shopping\_type\_model,  
ue.weight,  
ue.height  
from   
(select \* from qfbap\_dwd.dwd\_user where state = 'active' and dt = '9999-12-31') us  
left join qfbap\_dwd.dwd\_user\_extend ue   
on  
ue.user\_id = us.user\_id   
;

调度脚本

# 3user\_basic.sh  
args=$1  
dt=  
if [ ${#args} == 0 ]  
then  
dt=`date -d "2 days ago" "+%Y-%m-%d"`  
else  
dt=${#args}  
fi  
  
/usr/local/hive-1.1.0-cdh5.13.2/bin/hive --hiveconf params=${dt} -f 3user\_basic.hql

azkaban任务配置

#3user\_basic.job  
type=command  
command=/bin/bash 3user\_basic.sh  
dependencies=2sqoop\_exec\_job\_incr

用户访问模型

-- user\_visit.hql  
create database if not exists qfbap\_dwd;  
  
CREATE TABLE if not exists qfbap\_dwd.dwd\_user\_pc\_pv (  
log\_id bigint ,  
user\_id bigint ,  
session\_id string ,  
cookie\_id string ,  
in\_time string,  
out\_time string,  
stay\_time bigint,  
pv bigint,  
visit\_os string ,  
browser\_name string ,  
visit\_ip string ,  
province string ,  
city string ,  
dw\_date string  
)  
partitioned by (dt string)  
;  
  
insert overwrite table qfbap\_dwd.dwd\_user\_pc\_pv partition(dt='${hiveconf:params}')  
select  
max(log\_id) log\_id,  
user\_id,  
session\_id,  
cookie\_id,  
min(visit\_time) in\_time,  
max(visit\_time) out\_time,  
unix\_timestamp(max(visit\_time)) - unix\_timestamp(min(visit\_time)) stay\_time,  
count(1) pv,  
visit\_os,  
browser\_name,  
visit\_ip,  
province,  
city,  
current\_timestamp() dw\_date  
from qfbap\_ods.ods\_user\_pc\_click\_log   
where dt = '${hiveconf:params}'  
group by   
user\_id,  
session\_id,  
cookie\_id,  
visit\_os,  
browser\_name,  
visit\_ip,  
province,  
city,  
dt  
;  
  
CREATE external TABLE if not exists qfbap\_dwd.dwd\_user\_app\_pv (  
 log\_id bigint ,  
 user\_id bigint ,  
 imei string ,  
 log\_time string ,  
 log\_hour string,  
 visit\_os string ,  
 os\_version string ,  
 app\_name string ,  
 app\_version string ,  
 device\_token string ,  
 visit\_ip string ,  
 province string ,  
 city string ,  
 dw\_date string  
)   
partitioned by (dt string)  
;  
  
insert overwrite table qfbap\_dwd.dwd\_user\_app\_pv partition (dt = '${hiveconf:params}')  
select  
log\_id,  
user\_id,  
imei,  
log\_time,  
hour(log\_time) log\_hour,  
visit\_os,  
os\_version,  
app\_name,  
app\_version,  
device\_token,  
visit\_ip,  
province,  
city,  
current\_timestamp() dw\_date  
from qfbap\_ods.ods\_user\_app\_click\_log  
where dt = '${hiveconf:params}'  
;  
  
create database if not exists qfbap\_dws;  
create table if not exists qfbap\_dws.dws\_user\_visit\_month1  
(  
 user\_id bigint ,  
 type string,  
 content string,  
 cnt bigint,  
 rn bigint,  
 dw\_date string  
)  
partitioned by (dt string)  
stored as orc  
;  
  
insert overwrite table qfbap\_dws.dws\_user\_visit\_month1 partition(dt = '${hiveconf:params}')  
select   
user\_id,  
type,  
content,  
cnt,  
row\_number() over(distribute by user\_id,type sort by cnt desc) rn,  
current\_timestamp() dw\_date  
from   
(  
select  
user\_id,  
'visit\_ip' as type,  
visit\_ip as content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv   
where dt >= date\_sub('${hiveconf:params}',29) and dt <= '${hiveconf:params}'  
group by   
user\_id,  
visit\_ip  
union all  
select  
user\_id,  
'cookie\_id' as type,  
cookie\_id as content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv   
where dt >= date\_sub('${hiveconf:params}',29) and dt <= '${hiveconf:params}'  
group by   
user\_id,  
cookie\_id  
union all  
select  
user\_id,  
'browser\_name' as type,  
browser\_name as content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv   
where dt >= date\_sub('${hiveconf:params}',29) and dt <= '${hiveconf:params}'  
group by   
user\_id,  
browser\_name  
union all  
select  
user\_id,  
'visit\_os' as type,  
visit\_os as content,  
sum(pv) cnt  
from qfbap\_dwd.dwd\_user\_pc\_pv   
where dt >= date\_sub('${hiveconf:params}',29) and dt <= '${hiveconf:params}'  
group by   
user\_id,  
visit\_os  
) pc  
;  
  
create database if not exists qfbap\_dm;  
CREATE TABLE if not exists qfbap\_dm.dm\_user\_visit(  
user\_id bigint,   
last\_time string,   
last\_session string,   
last\_cookie string,   
last\_pv bigint,   
last\_browser string,   
last\_os string,   
first\_time string,   
first\_session string,   
first\_cookie string,   
first\_pv bigint,   
first\_browser string,   
first\_os string,   
cnt7 bigint,   
cnt15 bigint,   
cnt30 bigint,   
cnt60 bigint,   
cnt90 bigint,   
day30 bigint,   
pv30 bigint,   
avgpv30 double,   
pv025 bigint,   
pv627 bigint,   
pv829 bigint,   
pv10211 bigint,   
pv12213 bigint,   
pv14215 bigint,   
pv16217 bigint,   
pv18219 bigint,   
pv20221 bigint,   
pv22223 bigint,   
ipcnt bigint,   
topip string,   
ckcnt bigint,   
topck string,   
topbr string,   
topos string,   
last\_apptime string,   
last\_appname string,   
last\_appos string,   
last\_appip string,   
first\_apptime string,   
first\_appname string,   
first\_appos string,   
first\_appip string,   
appcnt7 bigint,   
appcnt15 bigint,   
appcnt30 bigint,   
appcnt60 bigint,   
appcnt90 bigint,   
apppv025 bigint,   
apppv627 bigint,   
apppv829 bigint,   
apppv10211 bigint,   
apppv12213 bigint,   
apppv14215 bigint,   
apppv16217 bigint,   
apppv18219 bigint,   
apppv20221 bigint,   
apppv22223 bigint,   
maxip string,   
maxprovince string,   
maxcity string,   
minip string,   
minprovince string,   
mincity string)  
partitioned by (dt string)  
;  
  
insert overwrite table qfbap\_dm.dm\_user\_visit partition(dt = '${hiveconf:params}')  
select   
user\_id,   
last\_time,   
last\_session,   
last\_cookie,   
last\_pv,   
last\_browser,   
last\_os,   
first\_time,   
first\_session,   
first\_cookie,   
first\_pv,   
first\_browser,   
first\_os,   
cnt7,   
cnt15,   
cnt30,   
cnt60,   
cnt90,   
day30,   
pv30,   
avgpv30,   
pv025,   
pv627,   
pv829,   
pv10211,   
pv12213,   
pv14215,   
pv16217,   
pv18219,   
pv20221,   
pv22223,   
ipcnt,   
topip,   
ckcnt,   
topck,   
topbr,   
topos,   
last\_apptime,   
last\_appname,   
last\_appos,   
last\_appip,   
first\_apptime,   
first\_appname,   
first\_appos,   
first\_appip,   
appcnt7,   
appcnt15,   
appcnt30,   
appcnt60,   
appcnt90,   
apppv025,   
apppv627,   
apppv829,   
apppv10211,   
apppv12213,   
apppv14215,   
apppv16217,   
apppv18219,   
apppv20221,   
apppv22223,  
case when last\_time > last\_apptime then last\_ip else last\_appip end maxip,  
case when last\_time > last\_apptime then last\_province else last\_appprovince end maxprovince,  
case when last\_time > last\_apptime then last\_city else last\_appcity end maxcity,  
case when first\_time < first\_apptime then first\_ip else first\_appip end minip,  
case when first\_time < first\_apptime then first\_province else first\_appprovince end minprovince,  
case when first\_time < first\_apptime then first\_city else first\_appcity end mincity  
from   
(  
select  
us.user\_id,  
max(case when pc.rn\_desc = 1 then pc.in\_time end) as last\_time,  
max(case when pc.rn\_desc = 1 then pc.session\_id end) as last\_session,  
max(case when pc.rn\_desc = 1 then pc.cookie\_id end) as last\_cookie,  
max(case when pc.rn\_desc = 1 then pc.pv end) as last\_pv,  
max(case when pc.rn\_desc = 1 then pc.browser\_name end) as last\_browser,  
max(case when pc.rn\_desc = 1 then pc.visit\_os end) as last\_os,  
max(case when pc.rn\_desc = 1 then pc.visit\_ip end) as last\_ip,  
max(case when pc.rn\_desc = 1 then pc.province end) as last\_province,  
max(case when pc.rn\_desc = 1 then pc.city end) as last\_city,  
max(case when pc.rn\_asc = 1 then pc.in\_time end) as first\_time,  
max(case when pc.rn\_asc = 1 then pc.session\_id end) as first\_session,  
max(case when pc.rn\_asc = 1 then pc.cookie\_id end) as first\_cookie,  
max(case when pc.rn\_asc = 1 then pc.pv end) as first\_pv,  
max(case when pc.rn\_asc = 1 then pc.browser\_name end) as first\_browser,  
max(case when pc.rn\_asc = 1 then pc.visit\_os end) as first\_os,  
max(case when pc.rn\_asc = 1 then pc.visit\_ip end) as first\_ip,  
max(case when pc.rn\_asc = 1 then pc.province end) as first\_province,  
max(case when pc.rn\_asc = 1 then pc.city end) as first\_city,  
count(case when pc.dt7 = 1 then pc.session\_id end) as cnt7,  
count(case when pc.dt15 = 1 then pc.session\_id end) as cnt15,  
count(case when pc.dt30 = 1 then pc.session\_id end) as cnt30,  
count(case when pc.dt60 = 1 then pc.session\_id end) as cnt60,  
count(case when pc.dt90 = 1 then pc.session\_id end) as cnt90,  
count(distinct substring(case when pc.dt30 = 1 then pc.in\_time end,1,10)) day30,  
sum(case when pc.dt30 = 1 then pc.pv end) pv30,  
sum(case when pc.dt30 = 1 then pc.pv end) / count(distinct substring(case when pc.dt30 = 1 then pc.in\_time end,1,10)) avgpv30,  
sum(case when pc.dt30 = 1 and pc.hr025 = 1 then pc.pv end) pv025,  
sum(case when pc.dt30 = 1 and pc.hr627 = 1 then pc.pv end) pv627,  
sum(case when pc.dt30 = 1 and pc.hr829 = 1 then pc.pv end) pv829,  
sum(case when pc.dt30 = 1 and pc.hr10211 = 1 then pc.pv end) pv10211,  
sum(case when pc.dt30 = 1 and pc.hr12213 = 1 then pc.pv end) pv12213,  
sum(case when pc.dt30 = 1 and pc.hr14215 = 1 then pc.pv end) pv14215,  
sum(case when pc.dt30 = 1 and pc.hr16217 = 1 then pc.pv end) pv16217,  
sum(case when pc.dt30 = 1 and pc.hr18219 = 1 then pc.pv end) pv18219,  
sum(case when pc.dt30 = 1 and pc.hr20221 = 1 then pc.pv end) pv20221,  
sum(case when pc.dt30 = 1 and pc.hr22223 = 1 then pc.pv end) pv22223,  
count(case when mon.type = 'visit\_ip' then mon.content end) ipcnt,  
max(case when mon.type = 'visit\_ip' and mon.rn = 1 then mon.content end) topip,  
count(case when mon.type = 'cookie\_id' then mon.content end) ckcnt,  
max(case when mon.type = 'cookie\_id' and mon.rn = 1 then mon.content end) topck,  
max(case when mon.type = 'browser\_name' and mon.rn = 1 then mon.content end) topbr,  
max(case when mon.type = 'visit\_os' and mon.rn = 1 then mon.content end) topos,  
max(case when app.rn\_desc = 1 then app.log\_time end) last\_apptime,  
max(case when app.rn\_desc = 1 then app.app\_name end) last\_appname,  
max(case when app.rn\_desc = 1 then app.visit\_os end) last\_appos,  
max(case when app.rn\_desc = 1 then app.visit\_ip end) last\_appip,  
max(case when app.rn\_desc = 1 then app.province end) last\_appprovince,  
max(case when app.rn\_desc = 1 then app.city end) last\_appcity,  
max(case when app.rn\_asc = 1 then app.log\_time end) first\_apptime,  
max(case when app.rn\_asc = 1 then app.app\_name end) first\_appname,  
max(case when app.rn\_asc = 1 then app.visit\_os end) first\_appos,  
max(case when app.rn\_asc = 1 then app.visit\_ip end) first\_appip,  
max(case when app.rn\_asc = 1 then app.province end) first\_appprovince,  
max(case when app.rn\_asc = 1 then app.city end) first\_appcity,  
count(case when app.dt7 = 1 then app.user\_id end) as appcnt7,  
count(case when app.dt15 = 1 then app.user\_id end) as appcnt15,  
count(case when app.dt30 = 1 then app.user\_id end) as appcnt30,  
count(case when app.dt60 = 1 then app.user\_id end) as appcnt60,  
count(case when app.dt90 = 1 then app.user\_id end) as appcnt90,  
sum(case when app.dt30 = 1 and app.hr025 = 1 then 1 end) apppv025,  
sum(case when app.dt30 = 1 and app.hr627 = 1 then 1 end) apppv627,  
sum(case when app.dt30 = 1 and app.hr829 = 1 then 1 end) apppv829,  
sum(case when app.dt30 = 1 and app.hr10211 = 1 then 1 end) apppv10211,  
sum(case when app.dt30 = 1 and app.hr12213 = 1 then 1 end) apppv12213,  
sum(case when app.dt30 = 1 and app.hr14215 = 1 then 1 end) apppv14215,  
sum(case when app.dt30 = 1 and app.hr16217 = 1 then 1 end) apppv16217,  
sum(case when app.dt30 = 1 and app.hr18219 = 1 then 1 end) apppv18219,  
sum(case when app.dt30 = 1 and app.hr20221 = 1 then 1 end) apppv20221,  
sum(case when app.dt30 = 1 and app.hr22223 = 1 then 1 end) apppv22223  
from qfbap\_dm.dm\_user us  
left join   
(select   
user\_id,  
session\_id,  
cookie\_id,  
visit\_os,  
browser\_name,  
visit\_ip,  
province,  
city,  
pv,  
in\_time,  
out\_time,  
stay\_time,  
row\_number() over(distribute by user\_id sort by in\_time asc) as rn\_asc,  
row\_number() over(distribute by user\_id sort by in\_time desc) as rn\_desc,  
case when dt >= date\_sub('${hiveconf:params}',6) then 1 end as dt7,  
case when dt >= date\_sub('${hiveconf:params}',14) then 1 end as dt15,  
case when dt >= date\_sub('${hiveconf:params}',29) then 1 end as dt30,  
case when dt >= date\_sub('${hiveconf:params}',59) then 1 end as dt60,  
case when dt >= date\_sub('${hiveconf:params}',89) then 1 end as dt90,  
case when hour(in\_time) between 0 and 5 then 1 end as hr025,  
case when hour(in\_time) between 6 and 7 then 1 end as hr627,  
case when hour(in\_time) between 8 and 9 then 1 end as hr829,  
case when hour(in\_time) between 10 and 11 then 1 end as hr10211,  
case when hour(in\_time) between 12 and 13 then 1 end as hr12213,  
case when hour(in\_time) between 14 and 15 then 1 end as hr14215,  
case when hour(in\_time) between 16 and 17 then 1 end as hr16217,  
case when hour(in\_time) between 18 and 19 then 1 end as hr18219,  
case when hour(in\_time) between 20 and 21 then 1 end as hr20221,  
case when hour(in\_time) between 22 and 23 then 1 end as hr22223  
from qfbap\_dwd.dwd\_user\_pc\_pv  
where dt >= date\_sub('${hiveconf:params}',89) and dt <= '${hiveconf:params}'  
) pc on us.user\_id = pc.user\_id  
left join qfbap\_dws.dws\_user\_visit\_month1 mon on mon.dt = '${hiveconf:params}' and mon.user\_id = us.user\_id   
left join   
(  
select   
user\_id,  
imei,  
log\_time,  
log\_hour,  
visit\_os,  
os\_version,  
app\_name,  
app\_version,  
device\_token,  
visit\_ip,  
province,  
city,  
row\_number() over(distribute by user\_id sort by log\_time asc) rn\_asc,  
row\_number() over(distribute by user\_id sort by log\_time desc) rn\_desc,  
case when dt >= date\_sub('${hiveconf:params}',6) then 1 end dt7,  
case when dt >= date\_sub('${hiveconf:params}',14) then 1 end dt15,  
case when dt >= date\_sub('${hiveconf:params}',29) then 1 end dt30,  
case when dt >= date\_sub('${hiveconf:params}',59) then 1 end dt60,  
case when dt >= date\_sub('${hiveconf:params}',89) then 1 end dt90,  
case when log\_hour between 0 and 5 then 1 end as hr025,  
case when log\_hour between 6 and 7 then 1 end as hr627,  
case when log\_hour between 8 and 9 then 1 end as hr829,  
case when log\_hour between 10 and 11 then 1 end as hr10211,  
case when log\_hour between 12 and 13 then 1 end as hr12213,  
case when log\_hour between 14 and 15 then 1 end as hr14215,  
case when log\_hour between 16 and 17 then 1 end as hr16217,  
case when log\_hour between 18 and 19 then 1 end as hr18219,  
case when log\_hour between 20 and 21 then 1 end as hr20221,  
case when log\_hour between 22 and 23 then 1 end as hr22223  
from qfbap\_dwd.dwd\_user\_app\_pv   
where dt >= date\_sub('${hiveconf:params}',89) and dt <= '${hiveconf:params}'  
) app on us.user\_id = app.user\_id   
group by   
us.user\_id  
) all  
;

调度脚本

# 4user\_visit.sh  
args=$1  
dt=  
if [ ${#args} == 0 ]  
then  
dt=`date -d "1 days ago" "+%Y-%m-%d"`  
else  
dt=${#args}  
fi  
  
/usr/local/hive-1.1.0-cdh5.13.2/bin/hive --hiveconf params=${dt} -f user\_visit.hql

azkaban任务配置

#4user\_visit.job  
type=command  
command=/bin/bash 4user\_visit.sh  
dependencies=3user\_basic

用户订单模型

-- 5user\_order.hql  
create database if not exists qfbap\_dwd;  
  
CREATE TABLE if not exists qfbap\_dwd.dwd\_us\_order (  
 order\_id bigint ,  
 order\_no string ,  
 order\_date string ,  
 order\_day string,  
 user\_id bigint ,  
 user\_name string ,  
 order\_money decimal(18,2) ,  
 order\_type int ,  
 order\_status int ,  
 pay\_status int ,  
 pay\_type int ,  
 order\_source string ,  
 update\_time string ,  
 goods\_amounts int,  
 goods\_moneys decimal(18,2),  
 trade\_type tinyint,  
 trade\_time string,  
 area\_name string,  
 user\_order\_flag tinyint,  
 dw\_date string  
)  
partitioned by (dt string)  
;  
  
insert overwrite table qfbap\_dwd.dwd\_us\_order partition(dt = '${hiveconf:params}')  
select  
od.order\_id,  
od.order\_no,  
od.order\_date,  
from\_unixtime(to\_unix\_timestamp(od.order\_date),'yyyy-MM-dd') order\_day,  
od.user\_id,  
od.user\_name,  
od.order\_money,  
od.order\_type,  
od.order\_status,  
od.pay\_status,  
od.pay\_type,  
od.order\_source,  
od.update\_time,  
oi.goods\_amounts,  
oi.goods\_moneys,  
ot.trade\_type,  
ot.trade\_time,  
ood.area\_name,  
oua.user\_order\_flag,  
current\_timestamp() dw\_date  
from qfbap\_ods.ods\_us\_order od  
left join   
(  
select   
order\_id,  
sum(goods\_amount) goods\_amounts,  
sum(goods\_amount \* curr\_price) goods\_moneys   
from qfbap\_ods.ods\_order\_item  
where dt = '${hiveconf:params}'  
group by order\_id  
) oi   
on oi.order\_id = od.order\_id  
left join qfbap\_ods.ods\_biz\_trade ot   
on ot.dt = '${hiveconf:params}' and ot.order\_id = od.order\_id  
left join qfbap\_ods.ods\_order\_delivery ood  
on ood.dt = '${hiveconf:params}' and ood.order\_id = od.order\_id  
left join qfbap\_ods.ods\_user\_addr oua  
on oua.addr\_id = ood.addr\_id  
;  
  
create database if not exists qfbap\_dws;  
create table if not exists qfbap\_dws.dws\_user\_order\_all  
(  
 user\_id bigint,  
 type string,  
 content string,  
 cnt bigint,  
 rn bigint,  
 dw\_date string  
)  
partitioned by (dt string)  
stored as orc  
;  
  
insert overwrite table qfbap\_dws.dws\_user\_order\_all partition(dt = '${hiveconf:params}')  
select  
user\_id,  
type,  
content,  
cnt,  
row\_number() over(distribute by user\_id,type sort by cnt desc) rn,  
current\_timestamp() dw\_date  
from  
(  
select   
user\_id,  
type,  
content,  
sum(cnt) cnt  
from  
(  
select  
user\_id,  
1 type,   
area\_name content,  
count(1) cnt  
from qfbap\_dwd.dwd\_us\_order  
where dt = '${hiveconf:params}'  
group by   
user\_id,  
area\_name  
union all  
select  
user\_id,  
2 type,   
pay\_type content,  
count(1) cnt  
from qfbap\_dwd.dwd\_us\_order  
where dt = '${hiveconf:params}'  
group by   
user\_id,  
pay\_type  
union all  
select  
user\_id,  
type,  
content,  
cnt  
from qfbap\_dws.dws\_user\_order\_all oa  
where oa.dt = date\_sub('${hiveconf:params}',1)  
) a  
group by  
user\_id,  
type,  
content  
) b  
;  
  
create database if not exists qfbap\_dm;  
  
CREATE TABLE if not exists `qfbap\_dm.dm\_user\_order`(  
 `user\_id` bigint,   
 `first\_order\_date` string,   
 `last\_order\_date` string,   
 `first\_diff\_day` int,   
 `last\_diff\_day` int,   
 `day30\_rr\_cnt` bigint,   
 `day30\_rr\_moneys` decimal(28,2),   
 `day60\_rr\_cnt` bigint,   
 `day60\_rr\_moneys` decimal(28,2),   
 `day90\_rr\_cnt` bigint,   
 `day90\_rr\_moneys` decimal(28,2),   
 `day30\_cnt` bigint,   
 `day30\_moneys` decimal(28,2),   
 `day60\_cnt` bigint,   
 `day60\_moneys` decimal(28,2),   
 `day90\_cnt` bigint,   
 `day90\_moneys` decimal(28,2),   
 `day90\_avgprice` decimal(38,16),   
 `max\_money` decimal(18,2),   
 `min\_money` decimal(18,2),   
 `all\_rr\_cnt` bigint,   
 `all\_rr\_moneys` decimal(28,2),   
 all\_rr\_avgprice decimal(28,2),  
 `all\_th\_cnt` bigint,   
 `all\_th\_moneys` decimal(28,2),   
 `all\_js\_cnt` bigint,   
 `all\_js\_moneys` decimal(28,2),   
 `max\_trade\_time` string,   
 `school\_cnt` bigint,   
 `company\_cnt` bigint,   
 `family\_cnt` bigint,   
 `hr025\_cnt` bigint,   
 `hr6212\_cnt` bigint,   
 `hr13215\_cnt` bigint,   
 `hr16220\_cnt` bigint,   
 `hr21223\_cnt` bigint,   
 `max\_area\_name` string,   
 `max\_pay\_type` string  
)  
partitioned by (dt string)  
;  
  
insert overwrite table qfbap\_dm.dm\_user\_order partition(dt = '${hiveconf:params}')  
select  
user\_id,  
max(first\_order\_date) first\_order\_date,  
max(last\_order\_date) last\_order\_date,  
max(first\_diff\_day) first\_diff\_day,  
max(last\_diff\_day) last\_diff\_day,  
max(day30\_rr\_cnt) day30\_rr\_cnt,  
max(day30\_rr\_moneys) day30\_rr\_moneys,  
max(day60\_rr\_cnt) day60\_rr\_cnt,  
max(day60\_rr\_moneys) day60\_rr\_moneys,  
max(day90\_rr\_cnt) day90\_rr\_cnt,  
max(day90\_rr\_moneys) day90\_rr\_moneys,  
max(day30\_cnt) day30\_cnt,  
max(day30\_moneys) day30\_moneys,  
max(day60\_cnt) day60\_cnt,  
max(day60\_moneys) day60\_moneys,  
max(day90\_cnt) day90\_cnt,  
max(day90\_moneys) day90\_moneys,  
max(day90\_avgprice) day90\_avgprice,  
max(max\_money) max\_money,  
min(min\_money) min\_money,  
sum(all\_rr\_cnt) all\_rr\_cnt,  
sum(all\_rr\_moneys) all\_rr\_moneys,  
sum(all\_rr\_moneys) / sum(all\_rr\_cnt) all\_rr\_avgprice,  
sum(all\_th\_cnt) all\_th\_cnt,  
sum(all\_th\_moneys) all\_th\_moneys,  
sum(all\_js\_cnt) all\_js\_cnt,  
sum(all\_js\_moneys) all\_js\_moneys,  
max(max\_trade\_time) max\_trade\_time,  
sum(school\_cnt) school\_cnt,  
sum(company\_cnt) company\_cnt,  
sum(family\_cnt) family\_cnt,  
sum(hr025\_cnt) hr025\_cnt,  
sum(hr6212\_cnt) hr6212\_cnt,  
sum(hr13215\_cnt) hr13215\_cnt,  
sum(hr16220\_cnt) hr16220\_cnt,  
sum(hr21223\_cnt) hr21223\_cnt,  
max(max\_area\_name) max\_area\_name,  
max(max\_pay\_type) max\_pay\_type  
from  
(  
select  
user\_id,  
max(case when uo.rn\_asc = 1 then uo.order\_date end) first\_order\_date,  
max(case when uo.rn\_desc = 1 then uo.order\_date end) last\_order\_date,  
datediff(current\_date(),max(case when uo.rn\_asc = 1 then uo.order\_date end)) first\_diff\_day,  
datediff(current\_date(),max(case when uo.rn\_desc = 1 then uo.order\_date end)) last\_diff\_day,  
count(case when uo.day30 = 1 and uo.trade\_type = 1 then uo.order\_id end) day30\_rr\_cnt,  
sum(case when uo.day30 = 1 and uo.trade\_type = 1 then uo.order\_money else 0 end) day30\_rr\_moneys,  
count(case when uo.day60 = 1 and uo.trade\_type = 1 then uo.order\_id end) day60\_rr\_cnt,  
sum(case when uo.day60 = 1 and uo.trade\_type = 1 then uo.order\_money else 0 end) day60\_rr\_moneys,  
count(case when uo.day90 = 1 and uo.trade\_type = 1 then uo.order\_id end) day90\_rr\_cnt,  
sum(case when uo.day90 = 1 and uo.trade\_type = 1 then uo.order\_money else 0 end) day90\_rr\_moneys,  
count(case when uo.day30 = 1 then uo.order\_id end) day30\_cnt,  
sum(case when uo.day30 = 1 then uo.order\_money else 0 end) day30\_moneys,  
count(case when uo.day60 = 1 then uo.order\_id end) day60\_cnt,  
sum(case when uo.day60 = 1 then uo.order\_money else 0 end) day60\_moneys,  
count(case when uo.day90 = 1 then uo.order\_id end) day90\_cnt,  
sum(case when uo.day90 = 1 then uo.order\_money else 0 end) day90\_moneys,  
sum(case when uo.day90 = 1 then uo.order\_money else 0 end) / count(case when uo.day90 = 1 then uo.order\_id end) day90\_avgprice,  
max(case when uo.day1 = 1 then uo.order\_money end) max\_money,  
min(case when uo.day1 = 1 then uo.order\_money end) min\_money,  
count(case when uo.day1 = 1 and uo.trade\_type = 1 then uo.order\_id end) all\_rr\_cnt,  
sum(case when uo.day1 = 1 and uo.trade\_type = 1 then uo.order\_money else 0 end) all\_rr\_moneys,  
count(case when uo.day1 = 1 and uo.trade\_type = -1 then uo.order\_id end) all\_th\_cnt,  
sum(case when uo.day1 = 1 and uo.trade\_type = -1 then uo.order\_money else 0 end) all\_th\_moneys,  
sum(case when uo.day1 = 1 and uo.trade\_type = -2 then uo.goods\_amounts else 0 end) all\_js\_cnt,  
sum(case when uo.day1 = 1 and uo.trade\_type = -2 then uo.goods\_moneys else 0 end) all\_js\_moneys,  
max(case when uo.day1 = 1 and uo.trade\_type = -1 then uo.trade\_time end) max\_trade\_time,  
count(case when uo.day1 = 1 and uo.user\_order\_flag = 1 then uo.order\_id end) school\_cnt,  
count(case when uo.day1 = 1 and uo.user\_order\_flag = 2 then uo.order\_id end) company\_cnt,  
count(case when uo.day1 = 1 and uo.user\_order\_flag = 3 then uo.order\_id end) family\_cnt,  
count(case when uo.day1 = 1 and uo.hr025 = 1 then uo.order\_id end) hr025\_cnt,  
count(case when uo.day1 = 1 and uo.hr6212 = 1 then uo.order\_id end) hr6212\_cnt,  
count(case when uo.day1 = 1 and uo.hr13215 = 1 then uo.order\_id end) hr13215\_cnt,  
count(case when uo.day1 = 1 and uo.hr16220 = 1 then uo.order\_id end) hr16220\_cnt,  
count(case when uo.day1 = 1 and uo.hr21223 = 1 then uo.order\_id end) hr21223\_cnt,  
null as max\_area\_name,  
null as max\_pay\_type  
from   
(  
select   
od.\*,  
row\_number() over(distribute by user\_id sort by order\_date asc) rn\_asc,  
row\_number() over(distribute by user\_id sort by order\_date desc) rn\_desc,  
case when od.order\_day >= date\_sub('${hiveconf:params}',29) and od.order\_day <= '${hiveconf:params}' then 1 end day30,  
case when od.order\_day >= date\_sub('${hiveconf:params}',59) and od.order\_day <= '${hiveconf:params}' then 1 end day60,  
case when od.order\_day >= date\_sub('${hiveconf:params}',89) and od.order\_day <= '${hiveconf:params}' then 1 end day90,  
case when od.order\_day = '${hiveconf:params}' then 1 end day1,  
case when hour(od.order\_date) between 0 and 5 then 1 end hr025,  
case when hour(od.order\_date) between 6 and 12 then 1 end hr6212,  
case when hour(od.order\_date) between 13 and 15 then 1 end hr13215,  
case when hour(od.order\_date) between 16 and 20 then 1 end hr16220,  
case when hour(od.order\_date) between 21 and 23 then 1 end hr21223  
from qfbap\_dwd.dwd\_us\_order od  
where dt >= date\_sub('${hiveconf:params}',89) and dt <= '${hiveconf:params}'  
) uo  
group by   
user\_id  
union all  
select  
user\_id,  
null first\_order\_date,  
null last\_order\_date,  
null first\_diff\_day,  
null last\_diff\_day,  
null day30\_rr\_cnt,  
null day30\_rr\_moneys,  
null day60\_rr\_cnt,  
null day60\_rr\_moneys,  
null day90\_rr\_cnt,  
null day90\_rr\_moneys,  
null day30\_cnt,  
null day30\_moneys,  
null day60\_cnt,  
null day60\_moneys,  
null day90\_cnt,  
null day90\_moneys,  
null day90\_avgprice,  
null max\_money,  
null min\_money,  
0 all\_rr\_cnt,  
0 all\_rr\_moneys,  
0 all\_th\_cnt,  
0 all\_th\_moneys,  
0 all\_js\_cnt,  
0 all\_js\_moneys,  
null max\_trade\_time,  
0 school\_cnt,  
0 company\_cnt,  
0 family\_cnt,  
0 hr025\_cnt,  
0 hr6212\_cnt,  
0 hr13215\_cnt,  
0 hr16220\_cnt,  
0 hr21223\_cnt,  
max(case when type = 1 and rn = 1 then content end) max\_area\_name,  
max(case when type = 2 and rn = 1 then content end) max\_pay\_type  
from qfbap\_dws.dws\_user\_order\_all   
where dt = '${hiveconf:params}'  
group by   
user\_id  
union all  
select  
user\_id,  
null first\_order\_date,  
null last\_order\_date,  
null first\_diff\_day,  
null last\_diff\_day,  
null day30\_rr\_cnt,  
null day30\_rr\_moneys,  
null day60\_rr\_cnt,  
null day60\_rr\_moneys,  
null day90\_rr\_cnt,  
null day90\_rr\_moneys,  
null day30\_cnt,  
null day30\_moneys,  
null day60\_cnt,  
null day60\_moneys,  
null day90\_cnt,  
null day90\_moneys,  
null day90\_avgprice,  
dm.max\_money,  
dm.min\_money,  
dm.all\_rr\_cnt,  
dm.all\_rr\_moneys,  
dm.all\_th\_cnt,  
dm.all\_th\_moneys,  
dm.all\_js\_cnt,  
dm.all\_js\_moneys,  
dm.max\_trade\_time,  
dm.school\_cnt,  
dm.company\_cnt,  
dm.family\_cnt,  
dm.hr025\_cnt,  
dm.hr6212\_cnt,  
dm.hr13215\_cnt,  
dm.hr16220\_cnt,  
dm.hr21223\_cnt,  
null max\_area\_name,  
null max\_pay\_type  
from qfbap\_dm.dm\_user\_order dm  
where dt = date\_sub('${hiveconf:params}',1)  
) b  
group by   
user\_id  
;

调度脚本

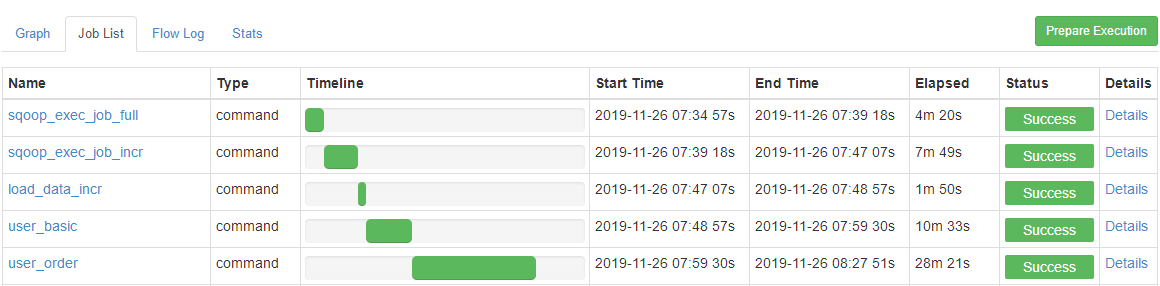
# 5user\_order.sh  
args=$1  
dt=  
if [ ${#args} == 0 ]  
then  
dt=`date -d "1 days ago" "+%Y-%m-%d"`  
else  
dt=${#args}  
fi  
  
/usr/local/hive-1.1.0-cdh5.13.2/bin/hive --hiveconf params=${dt} -f 5user\_order.hql

azkaban任务配置

#5user\_order.job  
type=command  
command=/bin/bash 5user\_order.sh  
dependencies=4user\_visit

完成后将以上所有文件打包到一个zip文件并上传到azkaban的任务

定期任务运行如下



# 六 知识点扩展

## 6.1 概念模型

概念模型用于信息世界的建模,是现实世界到信息世界的第一层抽象 。为了把现实世界中的具体事物抽象、组织为某一数据库管理系统支持的数据模型，人们常常首先将现实世界抽象为信息世界，然后将信息世界转换为机器世界。也就是说，首先把现实世界中的客观对象抽象为某一种信息结构，这种信息结构并不依赖于具体的计算机系统，不是某一个数据库管理系统（DBMS）支持的数据模型，而是概念级的模型，称为概念模型。

概念数据模型是面向用户、面向现实世界的数据模型，是与DBMS无关的。它主要用来描述一个单位的概念化结构。采用概念数据模型，数据库设计人员可以在设计的开始阶段，把主要精力用于了解和描述现实世界上，而把涉及DBMS的一些技术性的问题推迟到设计阶段去考虑。

由于概念模型用于信息世界的建模型，是现实世界到信息世界的第一层抽象，是用户与数据库设计人员之间进行交流的语言，因此概念模型一方面应该具有较强的语义表达能力，能够方便、直接地表达应用中的各种语义知识，另一方面它还应该简单、清晰、易于用户理解。

然而不同的人对同一个场景进行研究，可能提炼出来的概念模型都不一样，所以说这是颇受主观认识影响的一个过程。概念模型的质量对整个系统的影响至关紧要，因为，所谓的面向对象，就是从这里开始。一般来说，构建概念模型的过程与程序员的关系并不大。最适合进行这项活动的人，应该是那些有较深资历的领域专家，极端一点，甚至可以就是最为熟悉自身业务流程的客户代表。只要稍稍学习简单的建模知识，他们就可以胜任了。

### 6.1.1 建模过程

1、运用概念目录列表或名词性短语找出问题领域中的候选概念；

2、绘制概念到概念模型图中；

3、为概念添加关联关系；

4、为概念添加属性。

### 6.1.2 模型设计

概念模型不依赖于具体的计算机系统，他是纯粹反映信息需求的概念结构。

建模是在需求分析结果的基础上展开，常常要对数据进行抽象处理。常用的数据抽象方法是‘聚集’和‘概括’。

E-R方法是设计概念模型时常用的方法。用设计好的ER图再附以相应的说明书可作为阶段成果。

概念模型设计可分三步完成：

#### 6.1.2.1 局部模型

① 确定局部概念模型的范围；

② 定义实体；

③ 定义联系；

④ 确定属性；

⑤ 逐一画出所有的局部ER图，并附以相应的说明文件。

#### 6.1.2.2 全局模型

建立全局E-R图的步骤如下：

① 确定公共实体类型；

② 合并局部E-R图；

③ 消除不一致因素；

④ 优化全局E-R图；

⑤ 画出全局E-R图，并附以相应的说明文件。

#### 6.1.2.3 模型评审

概念模型的评审分两部分进行：

第一部分是用户评审。

第二部分是开发人员评审。

### 6.1.2 E-R图

E-R图主要是由实体、属性和联系三个要素构成的。在E-R图中，使用了下面四种基本的图形符号。

#### 6.1.2.1 成分

在ER图中有如下四个成分：

矩形框：表示实体，在框中记入实体名。

菱形框：表示联系，在框中记入联系名。

椭圆形框：表示实体或联系的属性，将属性名记入框中。对于主属性名，则在其名称下划一下划线。

连线：实体与属性之间；实体与联系之间；联系与属性之间用直线相连，并在直线上标注联系的类型。（对于一对一联系，要在两个实体连线方向各写1； 对于一对多联系，要在一的一方写1，多的一方写N；对于多对多关系，则要在两个实体连线方向各写N,M。

#### 6.1.2.2 构图要素

构成E-R图的3个基本要素是实体、属性和联系，其表示方法为：

##### 6.1.2.2.1 实体

一般认为，客观上可以相互区分的事物就是实体，实体可以是具体的人和物，也可以是抽象的概念与联系。关键在于一个实体能与另一个实体相区别，具有相同属性的实体具有相同的特征和性质。用实体名及其属性名集合来抽象和刻画同类实体**。**在E-R图中用**矩形**表示，矩形框内写明实体名；比如学生**张三**、学生**李四**都是实体。如果是弱实体的话，在矩形外面再套实线矩形。

##### 6.1.2.2.2 属性

实体所具有的某一特性，一个实体可由若干个属性来刻画。属性不能脱离实体，属性是相对实体而言的。在E-R图中用**椭圆形**表示，并用无向边将其与相应的实体连接起来；比如学生的姓名、学号、性别、都是属性。如果是多值属性的话，在椭圆形外面再套实线椭圆。如果是派生属性则用虚线椭圆表示。



##### 6.1.2.2.3 联系

联系也称关系，信息世界中反映实体内部或实体之间的关联。实体内部的联系通常是指组成实体的各属性之间的联系；实体之间的联系通常是指不同实体集之间的联系。在E-R图中用**菱形**表示，菱形框内写明联系名，并用无向边分别与有关实体连接起来，同时在无向边旁标上联系的类型（1 : 1，1 : n或m : n）。比如老师给学生授课存在授课关系，学生选课存在选课关系。如果是弱实体的联系则在菱形外面再套菱形。

#### 6.1.2.3 一般性约束

实体-联系数据模型中的联系型，存在3种一般性约束：一对一约束（联系）、一对多约束（联系）和多对多约束（联系），它们用来描述实体集之间的数量约束：

(1) 一对一联系(1 ∶1)

对于两个实体集A和B，若A中的每一个值在B中至多有一个实体值与之对应，反之亦然，则称实体集A和B具有一对一的联系。

一个学校只有一个正校长，而一个校长只在一个学校中任职，则学校与校长之间具有一对一联系。

(2) 一对多联系(1 : N)

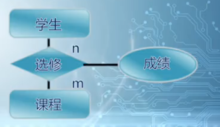
对于两个实体集A和B，若A中的每一个值在B中有多个实体值与之对应，反之B中每一个实体值在A中至多有一个实体值与之对应，则称实体集A和B具有一对多的联系。



例如，某校教师与课程之间存在一对多的联系“教”，即每位教师可以教多门课程，但是每门课程只能由一位教师来教。一个专业中有若干名学生，而每个学生只在一个专业中学习，则专业与学生之间具有一对多联系

(3) 多对多联系(M ∶N)

对于两个实体集A和B，若A中每一个实体值在B中有多个实体值与之对应，反之亦然，则称实体集A与实体集B具有多对多联系



实际上，一对一联系是一对多联系的特例，而一对多联系又是多对多联系的特例。 联系是随着数据库语义而改变的，假如有如下3种语义规定：

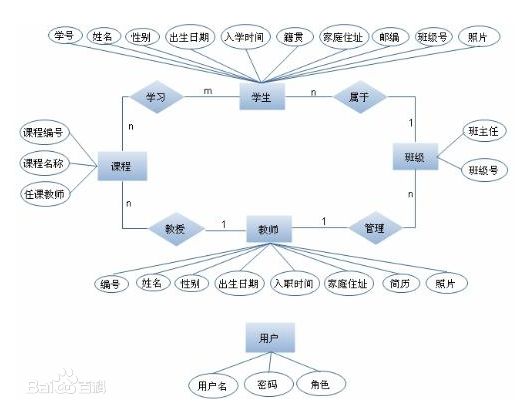
例如，一个部门有一个经理，而每个经理只在一个部门任职，则部门与经理的联系是一对一的。

一个员工可以同时是多个部门的经理，而一个部门只能有一个经理，则这种规定下“员工”与“部门”之间的“管理”联系就是1：n的联系了。

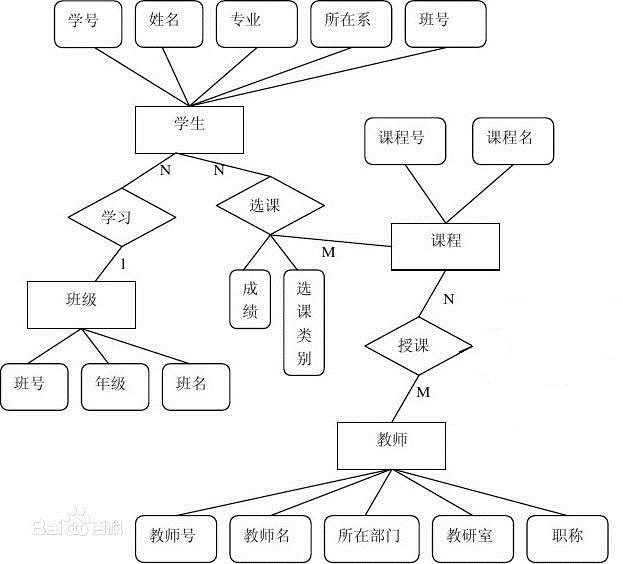
一个员工可以同时在多个部门工作，而一个部门有多个员工在其中工作，则“员工”与“部门”的“工作”联系为m:n联系。

### 6.1.3 实例

**实例图形1**



**实例图形2**



## 6.2 逻辑模型

### 6.2.1 逻辑模型定义

借助相对抽象、逻辑统一且结构稳健的结构，实现数据仓库系统所要求的数据存储目标，支持大量的分析应用，是实现业务智能的重要基础，同时也是数据管理分析的工具和交流的有效手段。

对于企业，**逻辑数据模型**（简称LDM）就是企业基础数据的一部分，它是企业数据资产的全面的、准确的描述，是数据整合的核心或目的。数据整合就是将不同来源的数据整合到一个统一定义、统一形式的LDM中。

### 6.2.2 模型分类

1. 层次数据模型
2. 网状数据模型
3. 关系数据模型
4. 面向对象模型

#### 6.2.2.1.层次模型

层次模型（Hierarchical Model）是最早出现的数据模型，它是采用层次数据结构来组织数据的数据模型。层次模型可以简单、直观地表示信息世界中实体、实体的属性以及实体之间的一对多联系。它使用记录类型来描述实体；使用字段来描述属性；使用结点之间的连线表示实体之间的联系。

**（1）层次模型的概念**

层次数据结构也称树型结构，树中的每个结点代表一种记录类型。满足以下两个条件的数据模型称为层次模型：

1）只有一个结点没有双亲结点（双亲结点也称父结点），该结点称为根结点。

2）根结点以外的其他结点有且只有一个双亲结点。

层次模型可以很自然地表示家族结构、行政组织结构等。

**（2）层次模型的三要素**

1）数据结构：使用记录类型表示实体，使用结点之间的连线表示一对多的联系。

2）数据操作：包括结点的查询和结点的更新（如插入、删除和修改）操作。

3）完整性约束：一个模型只有一个根结点；其他结点只能有一个双亲结点；结点之间是一对多的联系。

**（3）层次模型的优缺点**

**层次模型**的优点是结构简单、清晰，容易理解，结点之间联系简单，查询效率高。缺点主要有以下几点：

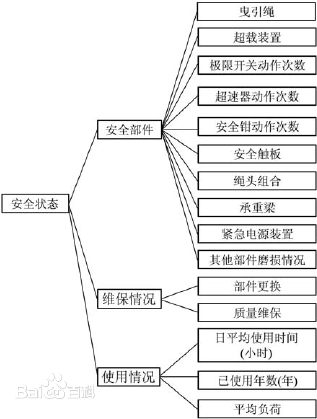
1）不能表示一个结点有多个双亲的情况。

2）不能直接表示多对多的联系，需要将多对多联系分解成多个一对多的联系。常用的分解方法是冗余结点法和虚拟结点法。

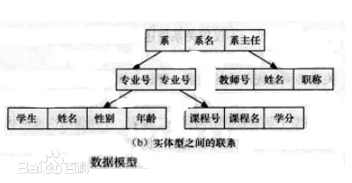
3）插入、删除限制多。比如，删除父结点则相应的子结点也被同时删除等。具体内容可参考“数据结构”课程中树的相关操作。

4）必须要经过父结点，才能查询子结点。因为在层次模型中，没有一个子结点的记录值能够脱离父结点的记录值而独立存在。

**实例图形1**



**实例图形2**



#### 6.2.2.2.网状模型

网状模型（Network Model）采用网状结构，能够直接描述一个结点有多个父结点以及结点之间为多对多联系的情形。

**（1）网状模型的概念**

网状模型是满足以下两个条件的基本层次联系的集合：

1）允许有一个以上的结点无双亲结点。

2）一个结点可以有多于一个的双亲结点。

实际上，层次模型是网状模型的一个特例。网状模型去掉了层次模型中的限制，允许多个结点没有双亲结点，允许结点有多个双亲结点，还允许结点之间存在多对多的联系。使用网状模型可以表示多对多联系。例如，如上图实例2所示，通过引入一个成绩的联结记录来表示学生和课程之间多对多的联系。网状模型中子结点与双亲结点的联系可以不唯一，但需要为每个联系进行命名，实例2中成绩结点有两个双亲结点：课程和学生。将课程与成绩的联系命名为“课程-成绩”，将学生与成绩的联系命名为“学生-成绩”。

**（2）网状模型的三要素**

1）数据结构：使用记录类型表示实体，使用字段来描述实体的属性，每个记录类型可包含若干个字段，使用结点之间的连线表示一对多的联系。

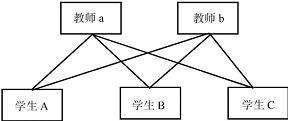
2）数据操作：包括结点的查询和结点的更新操作。

3）完整性约束：支持码的概念，用于唯一标识记录的数据项的集合；保证一个联系中双亲结点与子结点之间是一对多联系；支持双亲记录和子女记录之间的某些约束条件，如只删除双亲结点等。

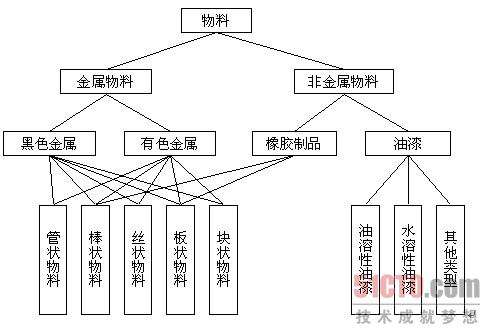
**（3）网状模型的优缺点**

网状模型具有良好的性能，存取效率较高。相比层次模型，网状模型中结点之间的联系具有灵活性，能表示事物之间的复杂联系，更适合描述客观世界。网状模型虽然有效克服了层次模型不方便表达多对多联系的缺点，但因为结构复杂，实现网状数据库管理系统比较困难。并且其所提供的DDL语言复杂，不容易学习和掌握。此外，由于实体间的联系本质上是通过存取路径来表现，因而，应用程序在访问数据时还需要指定存取路径。

**实例图形1**

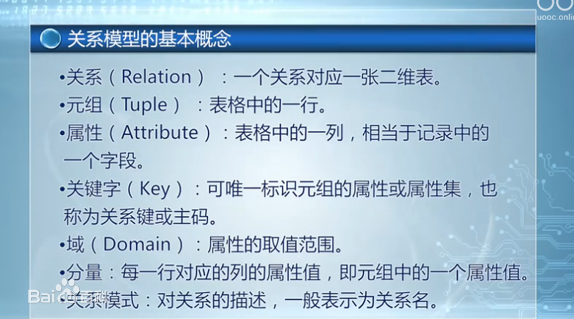


**实例图形2**



#### 6.2.2.3.关系模型

层次模型和网状模型的共同缺点是通过存取路径实现记录之间的联系，应用程序在访问数据时必须选择适当的存取路径，用户必须了解系统结构的细节，这样加重了编写应用程序的负担。另外，不支持集合处理，即没有提供一次处理多个记录的功能。 关系模型（Relational Model）在1970年由IBM公司的E.F.Codd首次提出。关系模型可以描述一对一、一对多和多对多的联系，并向用户隐藏存取路径，大大提高了数据的独立性以及程序员的工作效率。此外，关系模型建立在严格的数学概念和数学理论基础之上，支持集合运算。关系模型由关系数据结构、关系操作和完整性约束三部分组成。在关系模型中，实体和实体之间的联系均由关系来表示。



**（1）关系的定义：** 关系模型是一种简单的二维表格结构，每个二维表称做一个关系，一个二维表的表头，即所有列的标题称为一个元组，每一列数据称为一个属性，列标题称属性名。同一个关系中不允许出现重复元组和相同属性名的属性。

**（2）数据库体系结构**

① 外模式：或子模式、应用模式、局部模式等，它是对数据库在某个方面局部应用所涉及数据的[逻辑结构](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E7%BB%93%E6%9E%84/9663235)和特征的描述，它是终端用户和应用程序员所见到的数据库。

②模式：或概念模式、逻辑模式、全局模式等。它是对整个数据库逻辑结构和特征的描述，用户以DBMS支持的逻辑数据模型为基础。

③ 内模式：或存储模式、物理模式等。它是对整个数据库的存储结构和特征的描述。

**（3）DBMS的主要功能**

①、 数据定义和操纵

②、 数据库管理控制

③、 数据库辅助服务

④、 提供使用数据库工具

⑤、 建立和维护数据字典

**（4）关系运算**

①关系数据结构域：域是具有相同特性的数据集合。

笛卡儿积：笛卡儿积是定义在一组域上的集合，假定一组域用D1、D2、……Dn表示，则它们的笛卡儿积表示为：D1*D2*……\*Dn。

关系：关系到笛卡儿积的一个子集，若笛卡儿积具有n个域，则该笛卡儿积上的关系被称为n元关系。

码：码又称为键、关键字等。

候选码：关系中能惟一标识每个元组的最少属性或属性组被称为该关系的候选码。

主码：从候选码中选择一个作为该关系的主码，数据库系统将按主码标识和排序每个元组。

②关系完整性

实体完整性

实体完整性规则：关系的主码不能取空值，或者说任何关系中每个元组的主码不能为空。

参照完整性

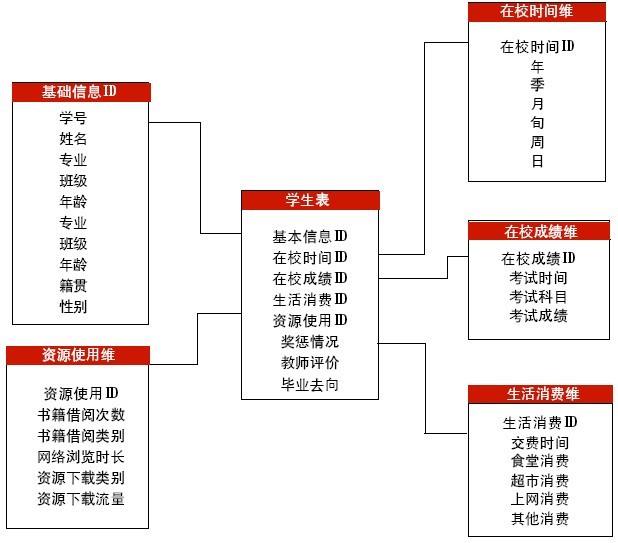
参照完整性规则：在两个参照和被参照关系中，参照关系中每个元组的外码或者为空，或者等于被参照关系中某个元组的主码。

③关系运算

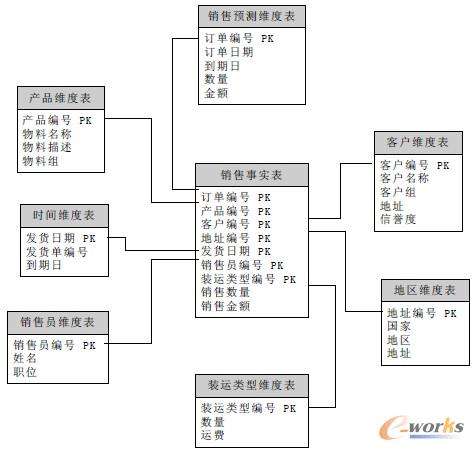
**（5）传统的集合运算**

包括：并、交、差和笛卡儿积等四种运算。

**实例图形1**



**实例图形2**



#### 6.2.2.4.面向对象模型

面向对象模型是一种新兴的数据模型，也是目前最重要的模型思路。它采用面向对象的方法来设计数据库。面向对象的数据库存储对象是以对象为单位，每个对象包含对象的属性和方法，具有类和继承等特点。

在面向对象数据库的设计中，我们将客观世界中的实体抽象成为对象。面向对象的方法中一个基本的信条是"任何东西都是对象"。对象可以定义为对一组信息及其操作的描述。对象之间的相互操作都得通过发送消息和执行消息完成，消息是对象之间的接口。严格地讲，在面向对象模型中，实体的任何属性都必须表示为相应对象中的一个变量和一对消息。变量用来保存属性值，一个消息用来读取属性值，另一个消息则用来更新这个值。

数据库中通常有很多相似的对象。"相似"是指它们响应相同的消息使用相同的方法，并有相同名称和类型的变量。对每个这样的对象单独进行定义是很浪费的，因此我们将相似的对象分组形成了一个"类"。类是相似对象的集合。类中的每个对象也称为类的实例。一个类中的所有对象共享一个公共的定义，尽管它们对变量所赋予的值不同。面向对象数据模型中类的概念相当于E-R模型中实体集的概念。

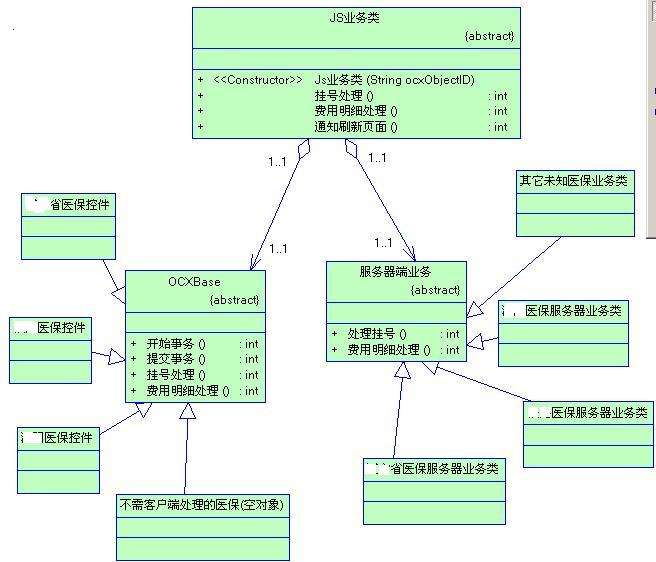
继承性允许不同类的对象共享它们公共部分的结构和特性。继承性可以用超类和子类的层次联系实现。一个子类可以继承某一个超类的结构和特性，这称为"单继承性"；一个子类也可以继承多个超类的结构和特性，这称为"多继承性"。继承性是数据间的泛化/细化联系，是一种"is a"联系。

面向对象系统提供一种"对象标识符"的概念来标识对象。OID与对象的物理存储位置无关，也与数据的描述方式和值无关。OID是惟一的。在对象创建的瞬间，由系统赋给对象，它在系统内是惟一的，在对象的生存期间，标识是不能改变的。如果要将数据转移到另外一个不同的数据库系统中，则标识符必须进行转化。

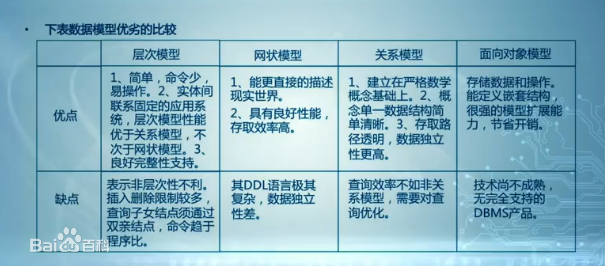
对象包含不同类的对象之间可能存在着包含关系）。包含其它对象的对象称为复合对象。包含关系可以有多层，形成类包含层次图。包含是一种"是一部分"（is part of）联系，因此包含与继承是两种不同的数据联系。

目前，一种结合关系数据库和面向对象特点的数据库为那些希望使用具有面向对象特征的关系数据库用户提供了一条捷径。这种数据库系统称为"对象关系数据库"，它是在传统关系数据模型基础上，提供元组、数组、集合一类丰富的数据类型以及处理新的数据类型操作能力，并且有继承性和对象标识等面向对象特点。

**实例图形**



**模型对比**



## 6.3 事实表

* 主要包含了描述特定商业事件的数据，即某些特定 商业事件的度量值。
* 一般情况下，事实表中的数据不允许修改，新的数 据只是简单地添加进事实表中。

发生在现实世界中操作型事件，其产生的可度量数值，存储的表统称事实表。例如交易表

一般有以下几种事实：

1. 可加、半可加、不可加事实 。

* 可加，例如pv(点击量) ； 半可加，例如数值差额，uv(用户量)；不可加，例如比率。

1. 事实一致性。不同事实表中的事实，应保证事实的定义是相同的，且具有相同的命名，如果不兼容，则须用不同命名方式，便于应用。
2. 事务事实表

* 事务事实表记录的事务层面的事实，保存的是最原子的数据，也称“原子事实表”。  
  事务事实表中的数据在事务事件发生后产生，数据的粒度通常是每个事务一条记录。一旦事务被提交，事实表数据被插入，数据就不再进行更改，其更新方式为增量更新。  
  事务事实表的日期维度记录的是事务发生的日期，它记录的事实是事务活动的内容。用户可以通过事务事实表对事务行为进行特别详细的分析。   
    
  通过事务事实表，还可以建立聚集事实表，为用户提供高性能的分析。

1. 周期事实。某天、某周等周期性，周期内未发生过程，也会有null或0等事实；

* 周期快照事实表以具有规律性的、可预见的时间间隔来记录事实，时间间隔如每天、每月、每年等等。典型的例子如销售日快照表、库存日快照表等。  
   周期快照事实表的粒度是每个时间段一条记录，通常比事务事实表的粒度要粗，是在事务事实表之上建立的聚集表。周期快照事实表的维度个数比事务事实表要少，但是记录的事实要比事务事实表多。  
    
   周期快照事实表的日期维度通常是记录时间段的终止日，记录的事实是这个时间段内一些聚集事实值。事实表的数据一旦插入即不能更改，其更新方式为增量更新。

1. 累计事实，开始与结束之间可预测步骤内的度量事件；

* 累积快照事实表和周期快照事实表有些相似之处，它们存储的都是事务数据的快照信息。但是它们之间也有着很大的不同，周期快照事实表记录的确定的周期的数据，而累积快照事实表记录的不确定的周期的数据。   
   累积快照事实表代表的是完全覆盖一个事务或产品的生命周期的时间跨度，它通常具有多个日期字段，用来记录整个生命周期中的关键时间点。另外，它还会有一个用于指示最后更新日期的附加日期字段。由于事实表中许多日期在首次加载时是不知道的，所以必须使用代理关键字来处理未定义的日期，而且这类事实表在数据加载完后，是可以对它进行更新的，来补充随后知道的日期信息。

1. 无事实的事实，比如：某天学生参加课程的事件；

* 在事实表中，通常会保存十个左右的维度外键和多个度量事实，度量事实是事实表的关键所在。在非事实型事实表中没有这些度量事实，只有多个维度外键。非事实型事实表通常用来跟踪一些事件或者说明某些活动的范围。下面举例来进行说明。  
    
   第一类非事实型事实表是用来跟踪事件的事实表。例如：学生注册事件，学校需要对学生按学期进行跟踪。维度表包括学期维度、课程维度、系维度、学生维度、注册专业维度和取得学分维度，而事实表是由这些维度的主键组成，事实只有注册数，并且恒为1。这样的事实表可以回答大量关于大学开课注册方面的问题，主要是回答各种情况下的注册数。  
    
   第二类非事实型事实表是用来说明某些活动范围的事实表。例如：促销范围事实表。通常销售事实表可以回答如促销商品的销售情况，但是对于那些没有销售出去的促销商品没法回答。这时，通过建立促销范围事实表，将商场需要促销的商品单独建立事实表保存。然后，通过这个促销范围事实表和销售事实表即可得出哪些促销商品没有销售出去。这样的促销范围事实表只是用来说明促销活动的范围，其中没有任何事实度量。

1. 聚集事实，聚合，提高查询性能；

* 细节事实表(detailed fact tables)中每条记录表示单一事实，  
  而聚集事实表(aggregated fact tables)中每条记录则聚合了多条事实。  
  从表的字段上看，细节事实表通常有设置TID属性，而聚集事实表则无。  
    
  两种事实表各有优缺点，细节事实表查询灵活但是响应速度相对慢，而聚集事实表虽然提高了查询速度，但使查询功能受到一定限制。

1. 合并事实，同粒度表进行合并；

## 6.4 维度表

### 6.4.1 维度定义

维度：看成观察数据的角度。

维度表：维度表可以看成是用户用来分析一个事实的窗口，它里面的数据应该是对事实的各个方面描述，比如时间维度表，它里面的数据就是一些日，周，月，季，年，日期等数据，维度表只能是事实表的一个分析角度。

### 6.4.2 分层维度

外链维表：有的维度是多个方向描述，此时可将多个维度分层级，形成规范的三范式。比如：地域维度，可能分层国家、省、市等三个维度，而将国家形成一个维度表、将省形成一个维度表、将市形成一个维度表，其中省、市维度表就是外链维度表。外链维表典型的就是雪花模型。

### 6.4.3 半维半实

既可以看着是维度表，也可以看着是事实的表，可以称之为半维半实，比如：订单表。

### 6.4.4 维度表内容

维度表有:

时间维度表,每个数据仓库都需要一个时间维度表。

地理维度表:描述位置信息的数据,如国家,省份,城市,区县,邮编等

产品维度表:描述产品及其属性

人员维度表:描述人员相关信息,部门员工表等

范围维度表:描述分段数据的信息等,比如信用等级、支付类型、支付方式、年龄段等

### 6.4.5 时间维度的另一种设计

创建日期时间的维度表，以年为维度，以季度为维度，以月为维度，以天为维度，各自创建时间维度

--创建年维度表  
create table your\_db.your\_table\_name(  
 year\_id int comment '年份ID:yyyy',  
 year\_name string comment '年份名称,2019年',  
 is\_leap\_year tinyint comment '是否为闰月：1是,0否',  
 start\_date string comment '开始日期',  
 end\_date string comment '结束日期',  
 day\_num int comment '多少天',  
 lunar\_year\_animal string comment '生肖',  
 lunar\_year string comment '阴历的天干地支'  
 ) row format delimited fields terminated by ','   
 stored as textfile;  
  
--创建季度维度表  
-- quarter\_id, quarter\_order, quarter\_name, season\_name, start\_date, end\_date, day\_num, year\_id  
  
create table if not exists table\_name(  
 quarter\_id int comment 'quarterID [20191,20192,20193,201904]',  
 quarter\_order int comment '1,2,3,4',  
 quarter\_name string comment 'quarter name',  
 season\_name string comment 'season name',  
 start\_date string comment 'start date',  
 end\_date string comment 'end date',  
 day\_num int comment 'day number',  
 year\_id int comment 'year id'   
 ) row format delimited fields terminated by ','   
 stored as textfile;  
  
--创建月份维度表  
-- month\_id, month\_order, month\_name, start\_date, end\_date, day\_num, quarter\_id, year\_id  
create table table\_name(  
 month\_id int comment 'monthID [201901~201912]',  
 month\_order int comment '1~12',  
 month\_name string comment 'month name',  
 start\_date string comment 'start date',  
 end\_date string comment 'end date',  
 day\_num string comment 'day number',  
 quarter\_id int comment 'quarter id',  
 year\_id int comment 'year id'   
 ) row format delimited fields terminated by ','   
 stored as textfile;

## 6.5 数据粒度

### 6.5.1 数据粒度定义

粒度是指数据仓库的数据单位中保存数据的细化或综合程度的级别。

粒度影响存放在数据仓库中的数据量的大小，同时影响数据仓库所能回答查询问题 的细节程度，是设计数据仓库的一个最重要方面。

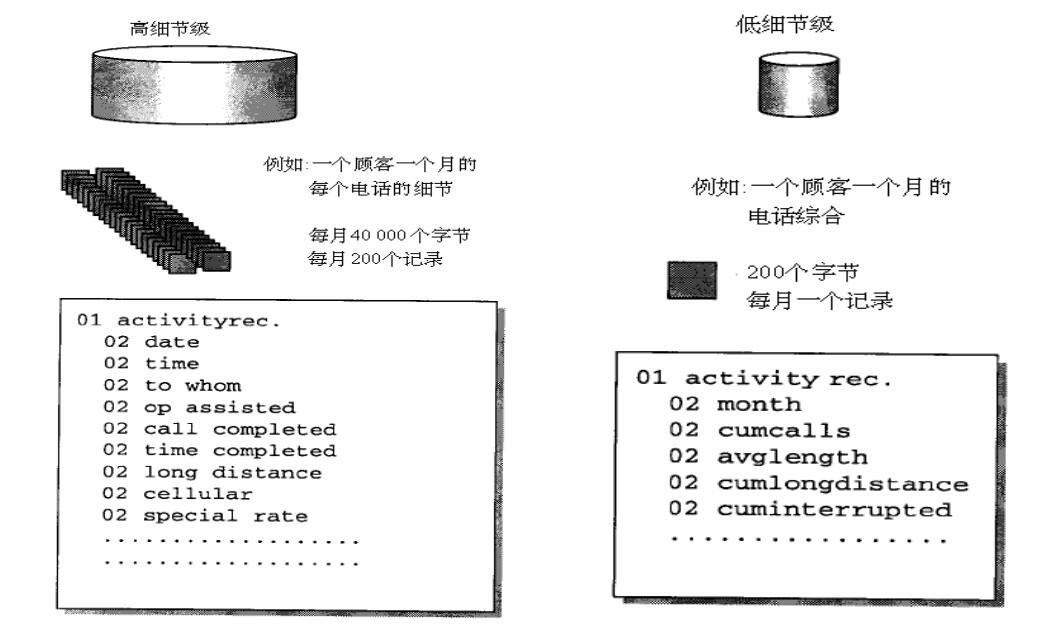
### 6.5.2 数据粒度分类

粒度可以分为两种形式：

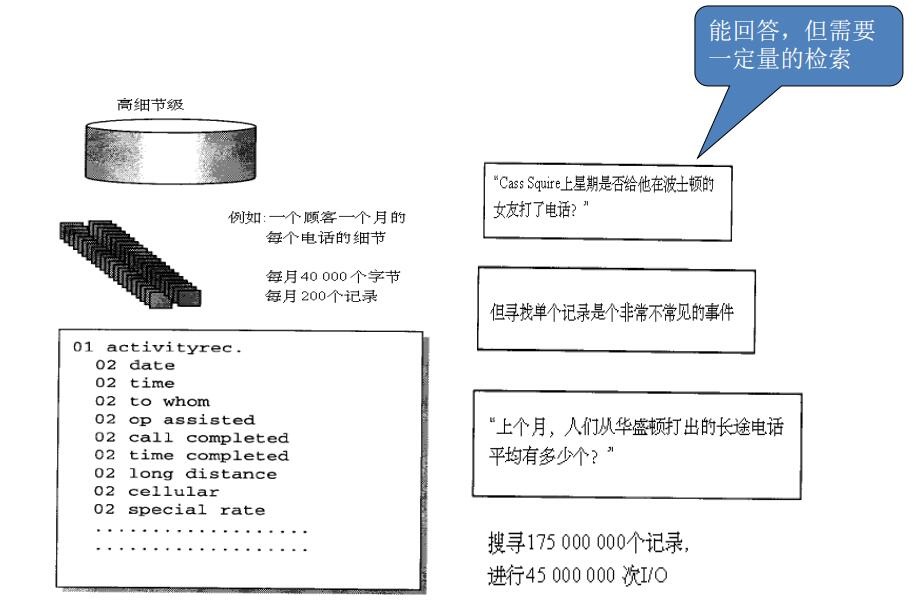
* 按时间段综合数据的粒度
* 按采样率高低划分的样本数据库

### 6.5.3 数据粒度例子

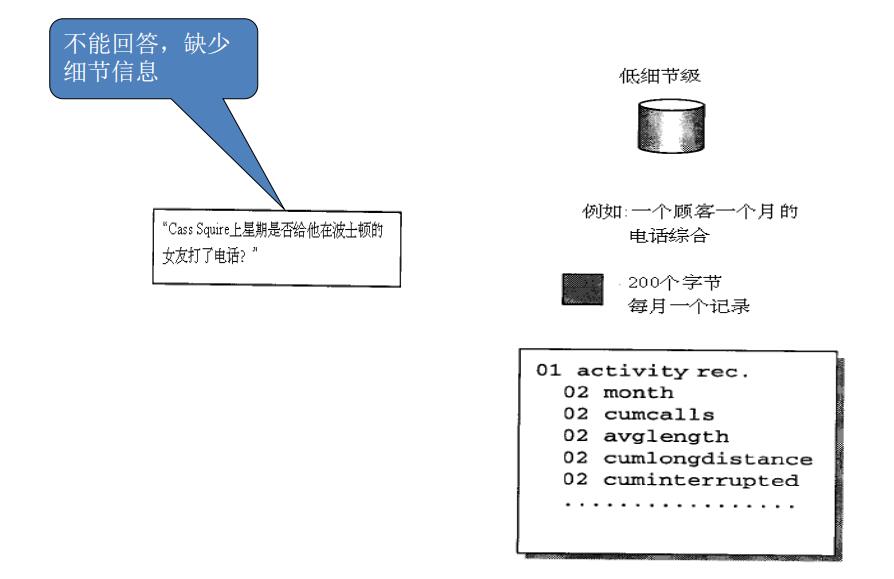
数据粒度的一个例子：



高细节级粒度：



低细节级粒度：



数据粒度的权衡：

