# 一 数据仓库

## 1.数据仓库分层

### 1.2 主题层

FDM主题层:  
团体

事件

协议

产品

财务

渠道

资产

公共

汇总

报表

## 2.遇到的困难

## 3.平时工作中如优化重复性的工作

## 4.给一个需求场景 简述如何设计开发

# 二 大数据

# 三 数据治理

# 四 Java基础

# 五 版本控制工具

$ git merge dev-20191119

# 六、面试记录

## PAJF面试

## 1. 挑选项目经历中最有价值的一个讲出来，如果让你重构，你会怎么样做才能做得更好？

## 2.数据库基础知识

文章参考地址：

<https://wenku.baidu.com/view/c73b15b17e21af45b207a850.html>

<https://www.jianshu.com/p/c65c4ec0fb3f>

<https://blog.csdn.net/dosthing/article/details/87954213>

（1） 数据库设计的三范式 原则及为了解决什么问题

为了建立冗余较小、结构合理的数据库，设计数据库时必须遵循一定的规则。在关系型数据库中这种规则就称为范式。范式是符合某一种设计要求的总结。要想设计一个结构合理的关系型数据库，必须满足一定的范式

在实际开发中最为常见的设计范式有三个：

1）.第一范式（确保每列保持原子性）

第一范式是最基本的范式。如果数据库表中的所有字段都是不可分解的原子值，就说明了该数据库表满足了第一范式。

实例：如以下订单表，买家地址列并不符合第一范式，需要继续拆分



上表所示的订单遵循了第一范式的要求，这样对用户使用城市进行分类的时候就非常方便，也提高了数据的性能

2）.第二范式（确保表中的每列都和主键相关）

第二范式在第一范式的基础之上更进一步。第二范式需要确保数据库中的每一列和主键相关，而不能只与主键的某一部分相关（主要针对联合主键而言）。也就是说在一个数据库表中，只能保存一种数据，不可以把多种数据保存在同一张数据库表中。

将上述订单表拆分成多张表，拆分后如下图



3）.第三范式（确保每列都和主键列直接相关，而不是间接相关）

第三范式需要确保数据表中的每一列数据都和主键直接相关，而不能间接相关，列和列之间不存在相互依赖关系。要求一个关系中不包含已在其它关系已包含的非主关键字信息

上述表中，虽然单价和订购数量可以计算出总价，但是单价是直接和商品相关的，并不是和订单直接相关，所以不满足第三范式，需要继续拆分



**（2）数据库的事务性**

除了数据库设计的三大范式之外，事务处理也是保证数据完整性的重要手段。事务是单独的工作单元，该单元可以包含多个操作以完成一个完整的任务。锁是在多用户环境中对数据访问的限制。事务和锁确保了数据的完整性。

**事务处理**

提交commit，当所有的操作步骤都被完整执行后，称该事务被提交。

回滚rollback，由于某一操作步骤执行失败，导致所有都没有被提交，则事务必须回滚，即回到事务执行前的状态。

**事务ACID属性**

事务处理的特性，每一个事务都有他们所共有的特性，叫做ACID特性，分别是原子性atomicity、一致性consistence、隔离性Isolation、持久性Durability

1）原子性，事务的原子性表示事务执行过程中，把事务作为一个工作单元处理，一个工作单元可能包含若干个操作步骤，每个操作步骤都必须完成才算完成，若因任何原因导致其中的一个步骤操作失败，则所有操作步骤失败，前面的步骤必须回滚。

2）一致性，事务的一致性保证数据处于一致状态。如果事务开始时处于一致状态，则事务结束时也应处于一致状态，不管事务成功还是失败。

3）隔离性，事务的隔离性保证事务访问的任何数据都不会受到其他事务所做的任何改变的影响，直到该事务完成。

4）持久性，事务的持久性保证假如事务执行，则它在系统中产生的结果应该是持久的。

（2） 数据库设计的拉链表 必须字段

（3） 数据库调优如何看调优的，怎么解决的，看什么字段

（4）给一个场景如何开发

（5）常用数据库语句，如何写，分组排序取最大/小

## 3.JAVA会不会

（1）spring /springMVC 的区别，有没有接触过springboot

## 4.SHELL脚本

（1） 定时语句/命令怎么写

（2） 查找一个文件中某个字符串出现的次数

## 5.其他

（1） 平时 有没有打建过系统环境，SVN,服务器之类的，如何搭建的

## 6.为什么要换工作，平时有没有对新技术进行钻研