# 数据库三范式

一范式：每列或者说每个字段都不可分解，具有原子性

二范式：非主键列要完全依赖主键列，不能只是依赖主键的一部分。即表必须有主键，且没有包含在主键中的列必须完全依赖于主键

（防止出现行冗余，避免出现多行重复数据）

三范式：非主键列要直接依赖主键列，而不是通过依赖非主键列间接依赖主键列。

（防止出现列冗余，避免出现无关的列，无关的多列信息应放在其他表中）

MYSQL：

索引B+树（使用），Hash(对区间查询不友好)

聚簇索引：索引和数据存放在一起，同一个文件中。

非聚簇索引：索引和数据存分开存放，存放在不同的文件中。

MYSQL中的两种存储引擎：

InnoDB属于聚簇索引，MYISAM属于非聚簇索引

InnoDB存储引擎存储数据时数据文件本身就是按B+树索引结构组织的，所以必须要有主键作为索引key值依据，若没有手动指定主键，数据库后台会自动选择一个唯一标识的字段建立主键；若表中没有唯一标识的列，则在表中默认添加一列作为主键字段，维护索引+数据文件。

SQL慢查询：

开启慢查询日志slow-log-log=1并设置慢查询时间阈值long qwuery time=xxx，设置慢查询日志文件slow-query-log-file=xxx。当SQL语句执行时间超过指定的阈值时记录查询日志。

慢查询优化：优化在某些查询速度特别慢（慢查询日志中执行时间大于一定程度的SQL查询语句）

SQL语句优化

索引优化

MVCC-多版本并发控制

SQL注入

网络七层/五层模型各层的作用

微服务架构：将一个单一的应用程序开发为一组小型服务的方法，每个服务都独立运行在自己的进程中，服务间通信采用“轻量级“通信机制，通常使用http资源的API来实现。这些服务围绕业务能力构建并且可通过全自动部署机制独立部署，这些服务共用一个最小的集中式的管理，服务可用不同的语言开发，使用不同的数据存储技术，即不受技术栈的限制。

微服务特征：

1. 每个服务独立运行在自己的进程中
2. 一系列运行的微服务共同构建起整个系统
3. 每个微服务可独立开发，开发过程中只要关注一个当前服务业务模块的特定功能，如支付服务、订单服务、用户管理服务等。
4. 可使用不同的技术栈开发不同的服务
5. 微服务之间直接通过轻量级的通信机制进行通信，如Rest API
6. 全自动的部署机制

微服务适用场景：

1. 大型项目
2. 有快速迭代的要求
3. 访问压力过大的网站

微服务拆分：

1.依据技术栈驱动设计

2.面向对象驱动设计

按职责划分（订单服务、用户服务、商品服务等）

按通用性划分（把一些通用功能做成微服务，如用户中心、支付在中心、消息中心，一个中台由若干个微服务构成）

Zookper，本质就是一个封装了很多算法的软件，投票推选leader，

一个分布式协调框架。

1. 发布订阅管理，作为注册中心
2. 分布式/集群管理
3. 安装 -> 启动zkServer -> 防火墙设置zkServer 端口通行

# 网络IO模型

## BIO：阻塞IO

Server端在accept和read或write时都是阻塞模式，主线程负责accept客户端发来的连接请求，若没有连接则一直阻塞在accept处，针对某个客户端发来的请求，起线程读取相应socket的数据作处理，若一直没有数据发来，则该线程阻塞在这里一直等待，直到socket中有数据传来。

1. 主线程accpet客户端的连接，每发来一个客户端连接，则创建一个线程来处理，这样客户端发来大量连接的情况下，会有非常多的线程，消耗系统资源；
2. 使用线程池进行优化，每当客户端发来一个连接时，将请求封装成任务发送给线程池调度，线程池管理线程，实现线程的复用，用少量的线程处理很多的客户端请求
3. 但是，用线程池能处理的客户端请求也是有限的

## NIO：非阻塞IO

Server端的线程或进行在处理socket时不再阻塞，而是想方法使用事件通知的机制，当有socket中接收到数据可被读取时通知和处理该socket的程序（线程或进程）去读取。

Select 模型，当有socket中传来数据时，返回一个大于等于1的整数，但是此时只是知道有socket中有数据，并不知道是哪个socket中传来的数据，程序要去遍历所有的socket，找出有数据的socket读取数据。Select模型通过一个数组管理被监控的socket。

Epoll模型，维护一个双端队列-就绪list，链接已经有数据可被处理的socket引用，此时不需要再像select那样遍历所有的socket；管理所有的socket使用一个红黑树结构。

等待队列 和 工作队列 中存放的是进程或线程的引用

## 额外：

网络数据传输到网卡后，网卡会发送一个中断信号给CPU，CPU会去优先处理将这些数据拷贝到内存和socket中~ 硬件有数据产生时会发送一个中断信号给CPU，CPU会优先处理，此种优先级比较高。

# HTTP

## HTTP1.0

无状态连接，每一个请求-响应都要三次握手建立连接->请求响应->断开连接

## HTTP1.1

Keep-Alive可以建立长连接，同一个连接被多个请求响应复用

HTTP1.0和HTTP1.1数据都是串行传输的，文本传输

## HTTP2

多路复用，数据可并行传输，使用帧为数据标记顺序，以二进制流传输，提高资源传输响应。