# 第二阶段面试题

## 一、网络编程

### 1. OSI 七层模型及其作用?

**答案:**

应用层：提供用户服务，具体功能由特定的程序而定
表示层：数据的压缩优化，加密
会话层：建立应用级的连接，选择传输服务
传输层：提供不同的传输服务.流量控制
网络层：路由选择，网络互连
链路层：进行数据交换，具体消息的发送，链路连接
物理层：物理硬件，借口设定，网卡路由交换机等

### 2. 简述一下三次握手四次挥手的过程?

**答案:**

三次握手：tcp 传输在数据传输前建立连接的过程

1 客户端向服务器发送连接请求
2 服务器收到请求后，回复确认消息，表示允许连接
3 客户端收到服务器恢复，进行最终标志发送确认连接

---

四次挥手：tcp 传输在连接断开前进行断开确认的过程
1 主动方发动报文告知被动方要断开连接
2 被动方收到请求后立即返回报文告知已经准备断开
3 被动方准备就绪后再次发送报文告知可以断开
4 主动方发送消息，确认最终断开

### 3. TCP 和 UDP 的区别?

**答案:**

1、基于连接与无连接
2、TCP 要求系统资源较多，UDP 较少；
3、流模式（TCP）与数据报模式(UDP);
4、TCP 保证数据正确性，UDP 可能丢包
5、TCP 保证数据顺序，UDP 不保证

### 4. 什么是水平触发？什么是边缘触发?

**答案:**

**水平触发**:如果文件描述符已经就绪可以非阻塞的执行 IO 操作了,此时会触发通知.允许在任意时刻重

复检测 IO 的状态.select,poll 就属于水平触发.

**边缘触发**:如果文件描述符自上次状态改变后有新的 IO 活动到来,此时会触发通知.在收到一个 IO 事件通知后要尽可能多的执行 IO 操作,因为如果在一次通知中没有执行完 IO 那么就需要等到下一次新的 IO 活动到来才能获取到就绪的描述符.

## 5. 创建 TCP 连接的流程?

**答案:**

**服务器端:** 1）创建套接字 create；2）绑定端口号 bind；3）监听连接 listen；4）接受连接请求 accept，并返回新的套接字；5）用新返回的套接字 recv/send；6）关闭套接字

**客户端:** 1）创建套接字 create; 2）发起建立连接请求 connect; 3）发送/接收数据 send/recv；4）关闭套接字

## 6. 创建 UDP 连接流程?

**答案:**

**服务器端:** 1）创建套接字 create；2）绑定端口号 bind；3）接收/发送消息 recvfrom/sendto；4）关闭套接字。

**客户端:** 1）创建套接字 create；2）发送/接收消息 sendto/recvfrom；3）关闭套接字.

## 7. 什么是并发和并行?

**答案:**

并发是指一个处理器同时处理多个任务。
并行是指多个处理器或者是多核的处理器同时处理多个不同的任务。
并发是逻辑上的同时发生（simultaneous），而并行是物理上的同时发生。

## 8. 生产者与消费者模型的应用场景

**答案:**

生产者与消费者模式是通过一个容器来解决生产者与消费者的强耦合关系，生产者与消费者之间不直接进行通讯，而是利用阻塞队列来进行通讯，生产者生成数据后直接丢给阻塞队列，消费者需要数据则从阻塞队列获取，实际应用中，生产者与消费者模式则主要解决生产者与消费者生产与消费的速率不一致的问题，达到平衡生产者与消费者的处理能力，而阻塞队列则相当于缓冲区。

## 9. HTTP 常见的几种提交方法?

**答案:**

GET、HEAD、POST、PUT、DELETE、TRACE、CONNECT

### 10. GET 方法和 POST 请求的区别?

**答案:**

GET 在浏览器回退时是无害的，而 POST 会再次提交请求。
GET 产生的 URL 地址可以被加入到书签，而 POST 不可以。
GET 请求会被浏览器主动 cache，而 POST 不会，除非手动设置。
GET 请求只能进行 url 编码，而 POST 支持多种编码方式。
GET 请求参数会被完整保留在浏览器历史记录里，而 POST 中的参数不会被保留。
GET 请求在 URL 中传送的参数是有长度限制的，而 POST 么有。
对参数的数据类型，GET 只接受 ASCII 字符，而 POST 没有限制。
GET 比 POST 更不安全，因为参数直接暴露在 URL 上，所以不能用来传递敏感信息。
GET 参数通过 URL 传递，POST 放在 Request body 中。
大多数浏览器通常都会限制 url 长度在 2K 个字节，而大多数服务器最多处理 64K 大小的 url。

## 二、进程线程

### 1. 进程、线程、协程的区别是什么?

**答案:**

1. 进程：进程是资源分配的最小单位，程序运行时系统就会创建一个进程，并为他分配资源（独立的地址空间，建立数据表来维护代码段、堆栈段和数据段）。

2. 线程：程序执行时的最小单位，它是进程的一个执行流，线程是 cpu 调度的最小单元。一个进程可以由很多个线程组成，线程间共享进程的所有资源，每个线程有自己的堆栈和局部变量

3. 协程：又称为轻量级的线程 协程极高的执行效率（子程序切换不是线程切换）,不需要多线程的锁机制，一个协程遇到 IO 操作自动切换到其它协程

### 2. 进程线程的各有什么优缺点?

**答案:**

进程:
缺点: 正是因为进程创建时都有自己的独立地址空间，所以创建进程和切换进程消耗资源比较多
优点: 进程间的通信是以 ipc 通信的方式进行，相对于线程来说，他的健壮性更强，因为线程之间是独立的，一个进程的销毁不会影响另外一个进程。

线程:
缺点: 相对来说健壮性不够强，因为他是进程的执行流，那么一个线程死掉了可能会影响整个进
程。而且任何一个线程都可以修改进程中的变量，那么就容易把内容改乱。
优点: 线程间共享进程的资源，那么创建线程和切换线程消耗较小。

## 3. 什么时候用进程，什么时候用线程?

**答案:**

linux 系统函数 fork（）可以为当前进程创建一个子进程，那么，在一个进程接收到来自客户端的请
求之后，父进程可以创建子进程去处理客户端的请求。父进程只需要监控来自客户端得请求的到来，创建子进程去处理，这样就能实现高并发

对于线程来说，线程是 cpu 的最小执行单元，一个 cpu 同时只能执行一个线程，那么在多 cpu 的环境
下就可以允许多个线程同时执行，同样多线程也能实现高并发操作。但是容易造成数据混乱，那么多线程需要同步和互斥。同步和互斥也是编写多线程程序的难点。

## 4. 多个进程如何占用 CPU?

**答案:**

** 一个内核同一时间只能运行一个任务
** 多个进程对内核进行争夺，操作系统决定那个进程占有计算机核心。
**占有计算机的核心的进程我们称为该进程占有 CPU 的时间片

## 5. 怎么查看 进程的状态？进程有那些信息，如何保存？

**答案:**

ps -aux 查看进程的信息
PCB （进程控制块）在内存中开辟的一块空间，里面保存着进程的信息，会随着进程的创建而创建，随着进程的结束而消失

## 6. 并行和并发的区别

**答案:**

并发： 同时处理多个请求，但是内核采用轮询时间片的方式逐个访问，某一时间点实际只处理一个任务。IO 多路复用 协程 循环服务器。单线程
并行： 使用多个内核，同时执行多个任务 多进程 多线程

## 7. 谈谈你对多线程的理解

**答案:**

多线程：一个进程可以开启多个线程，每条线程可以并行（同时）执行不同的任务，多线程技术可以提高程序的执行效率，多线程中，所有变量都由所有线程共享，任何一个变量都可被任何一个
线程修改。线程之间共享数据的最大危险在于多个线程同时更改一个变量

缺点：1．需要用到同步互斥

2．可能受到 GIL 的影响，但是网络 IO 线程并发还是可以的

优点：资源消耗比较少

原理：同一时间内，CPU 只能处理一条线程，只有一条线程在执行（工作），多线程并发（同时）执

行，其实是 CPU 快速的在多线程之间调度（切换）。因为 CPU 切换特别快，造成了多线程并发

的假象

## 8. 谈谈你对 python 全局解释器锁的理解

**答案：**

python ---》支持多线程 ---》同步互斥 ---》加锁 ---》超级锁（全局解释器锁，把解释器锁住）----》在同一时刻，解释器只能解释一个线程 -----》大量 python 库为了省事沿用了这种方法 ---》python 多线程效率低下

GIL 问题：由于 python 的全局解释器锁造成 python 的多线程执行效率低下

解决方案：

* 不使用线程，使用多进程（Python 执行效率低）
* 不使用 c 、c++ 做解释器 C# java
* python 线程适合高用时的 IO 操作，网路 IO。不适合 cpu 密集型程序

## 9. 线程中，会产生资源争夺的现象，谈谈你对 资源争夺解决的办法

**答案：**

同步：同步是一种合作关系，为完成某个任务多进程或者多线程之间形成一种协调，按照条件次序执行，传递告知资源情况。这种协调可能是因为阻塞关系达成的

互斥：互斥是一种制约关系，但一个进程或线程进入到临界区会进行加锁操作，此时其他进程（线程）

在企图操作临界资源就会阻塞。只有当资源被释放才能进行操作。

## 10. 怎么利用多核 CPU 呢？

**答案：**

最简单的方法是多进程加协程，既充分利用多核，有充分发挥协程的高效率，可获得极高的性能

# 三、MySQL 基础

## 1. MySQL 如何优化?

**答案:**

1.优化索引、SQL 语句、分析慢查询;

2.设计表的时候严格按照数据库的设计范式来设计数据库;

3.我们还可以将我们的业务架构进行缓存，静态化和分布式;

4.不用全文索引，使用 Xunsearch,ES 或者云服务器上的索引;

5.如果效率还是不够好，可以采用主从方式将数据读写分离;

6.可以加上 memcached 缓存，将经常被访问到但不经常变化的数据放至 memcached 缓存服务器里面，这样的话能够节约磁盘 I/O;

7.还可以优化硬件，在硬件层面，我们可以使用更好的一些硬盘（固态硬盘），使用一些磁盘阵列技术（raid0，raid1，raid5 ） ?

- raid0：最简单的（两块硬件相加 100G+100G=200G）?
- raid1：镜像卷，把同样的数据下两份。可以随即从 A/B 里面读取，效率更高，硬盘坏了一块数据也不会丢失

## 2. SQL 中，drop、delete、truncate 有什么区别?

**答案:**

三者都是删除的意思，但是三者个有些区别

- delete 和 truncate 只删除表的数据不删除表的结构
- 速度 drop > truncate > delete
- 想删除部分数据时， delete 删除时要带上 where 语句
- 保留表而想删除所有的数据时用 truncate

## 3. 写 SQL 语句

| name | kecheng | fenshu |
|---|---|---|
| 张三 | 语文 | 81 |
| 张三 | 数学 | 75 |
| 李四 | 语文 | 76 |
| 李四 | 数学 | 90 |
| 王五 | 语文 | 81 |
| 王五 | 数学 | 100 |
| 王五 | 英语 | 90 |



**答案:**

1、每门课都大于 80 分的学生姓名

select name from table1 GROUP BY name having min(fenshu)>80;（推荐）

select DISTINCT name from table1 where name not in (select DISTINCT name from table1 where fenshu<=80)；

2、删除除了编号不同，其他都相同的冗余学生信息

delete from table2 where bianhao not in (select min(bianhao) from table2 GROUP BY xuehao,name,kechenghao,kecheng,fenshu)；

注意：这么写 MySQL 会报错，`You can't specify target table 'table2' for update in FROM clause`。这个错误表示不能在同一表中查询的数据作为同一表的更新数据。

应该这么写： delete from table2 where bianhao not in (select a.bianhao from (select * from table2 GROUP BY xuehao,name,kechenghao,kecheng,fenshu)a)

## 4. 一个叫 team 的表，里面只有一个字段 name,一共有 4 条纪录，分别是 a,b,c,d,对应四个球队，现在四个球队进行比赛，用一条 sql 语句显示所有可能的比赛组合。

**答案：**

select * from team a,team b where a.name>b.name;

## 5. 索引，主键，外键的区别?

**答案：**

定义：

主键--唯一标识一条记录，不能有重复的，不允许为空

外键--表的外键是另一表的主键， 外键可以有重复的， 可以是空值

索引--该字段没有重复值，但可以有一个空值

作用：

主键--用来保证数据完整性

外键--用来和其他表建立联系用的

索引--是提高查询排序的速度

个数：

主键--主键只能有一个

外键--一个表可以有多个外键

索引--一个表可以有多个唯一索引

## 6. 简述在 MySQL 数据库中 MyISAM 和 InnoDB 的区别?

**答案:**

区别于其他数据库的最重要的特点就是其插件式的表存储引擎。切记:存储引擎是基于表的,而不是数据库。

InnoDB 与 MyISAM 的区别:

InnoDB 存储引擎: 主要面向 OLTP(Online Transaction Processing,在线事务处理)方面的应用,是第一个完整支持 ACID 事务的存储引擎(BDB 第一个支持事务的存储引擎,已经停止开发)。

特点:

· 行锁设计、支持外键,支持事务,支持并发,锁粒度是支持 mvcc 得行级锁;

MyISAM 存储引擎: 是 MySQL 官方提供的存储引擎,主要面向 OLAP(Online Analytical Processing,在线分析处理)方面的应用。

特点:

不支持事务,锁粒度是支持并发插入得表级锁,支持表所和全文索引。操作速度快,不能读写操作太频繁;

## 7. 解释 MySQL 外连接、内连接与自连接的区别?

**答案:**

先说什么是交叉连接: 交叉连接又叫笛卡尔积,它是指不使用任何条件,直接将一个表的所有记录和另一个表中的所有记录一一匹配。

内连接 则是只有条件的交叉连接,根据某个条件筛选出符合条件的记录,不符合条件的记录不会出现在结果集中,即内连接只连接匹配的行。

外连接 其结果集中不仅包含符合连接条件的行,而且还会包括左表、右表或两个表中的所有数据行,这三种情况依次称之为左外连接,右外连接,和全外连接。

左外连接,也称左连接,左表为主表,左表中的所有记录都会出现在结果集中,对于那些在右表中并没有匹配的记录,仍然要显示,右边对应的那些字段值以 NULL 来填充。右外连接,也称右连接,右表为主表,右表中的所有记录都会出现在结果集中。左连接和右连接可以互换,MySQL 目前还不支持全外连接。

## 8. 分库分表有没有用到,怎么实现的?

**答案:**

目前,根据我们的业务量,还没有使用分库分表。但是我有在关注 MySQL 的分布式方案,以前 mysql 分布式比较常用的方法是用阿里巴巴的 cobar,将一张表水平拆分成多份分别放入不同的库来实现表的水平拆分,或将不同的表放入不同的库,但是后来发现 cobar 有一个问题一直不能很好的解决。目前,我关注到有很多人用 mycat 替换了 cobar

## 9. char 和 varchar 的区别?

**答案:**

是一种固定长度的类型，varchar 则是一种可变长度的类型，它们的区别是： char(M)类型的数据列里，每个值都占用 M 个字节，如果某个长度小于 M，MySQL 就会在它的右边用空格字符补足.（在检索操作中那些填补出来的空格字符将被去掉）在 varchar(M)类型的数据列里，每个值只占用刚好够用的字节再加上一个用来记录其长度的字节（即总长度为 L+1 字节）.

varchar 得适用场景:字符串列得最大长度比平均长度大很多 2.字符串很少被更新，容易产生存储碎片 3.使用多字节字符集存储字符串

Char 得场景:存储具有近似得长度（md5 值,身份证，手机号）,长度比较短小得字符串（因为 varchar 需要额外空间记录字符串长度），更适合经常更新得字符串，更新时不会出现页分裂得情况，避免出现存储碎片，获得更好的 io 性能

## 10. 什么是数据库事务?

**答案:**

1. 单个逻辑单元执行的一系列操作,这些操作要么全做要么全不做,是不可分割的.事务的开始和结束用户是可以控制的,如果没控制则由数据库默认的划分事务.事务具有以下性质:

(1)原子性

指一个事务要么全执行,要么全不执行.也就是说一个事务不可能执行到一半就停止了.比如:你去买东西,钱付掉了,东西没拿.这两步必须同时执行 ,要么都不执行.

(2)一致性

指事务的运行并不改变数据库中的一致性.比如 a+b=10;a 改变了,b 也应该随之改变.

(3)独立性

两个以上的事务不会出现交替运行的状态,因为这样可能导致数据的不一致

(4)持久性

事务运行成功之后数据库的更新是永久的,不会无缘无故的回滚

# 四、MongoDB

## 1. Redis 和 MongoDB 的优缺点?

**答案:**

MongoDB 和 Redis 都是 NoSQL，采用结构型数据存储。二者在使用场景中，存在一定的区别，这也主要由于二者在内存映射的处理过程，持久化的处理方法不同。 MongoDB 建议集群部署，更多的考虑到集群方案，Redis 更偏重于进程顺序写入，虽然支持集群，也仅限于主-从模式.

Redis 优点:

读写性能优异

支持数据持久化，支持 AOF 和 RDB 两种持久化方式

支持主从复制，主机会自动将数据同步到从机，可以进行读写分离。

数据结构丰富：除了支持 string 类型的 value 外还支持 string、hash、set、sortedset、list 等数据结构。

Redis 缺点:

Redis 不具备自动容错和恢复功能， 主机从机的宕机都会导致前端部分读写请求失败，需要等待机器重启或者手动切换前端的 IP 才能恢复。

主机宕机，宕机前有部分数据未能及时同步到从机， 切换 IP 后还会引入数据不一致的问题，降低了系统的可用性。Redis 的主从复制采用全量复制，复制过程中主机会 fork 出一个子进程对内存做一份快照， 并将子进程的内存快照保存为文件发送给从机，这一过程需要确保主机有足够多的空余内存。若快照文件较大，对集群的服务能力会产生较大的影响，而且复制过程是在从机新加入集群或者从机和主机网络断开重连时都会进行，也就是网络波动都会造成主机和从机间的一次全量的数据复制，这对实际的系统运营造成了不小的麻烦。

Redis 较难支持在线扩容，在集群容量达到上限时在线扩容会变得很复杂。为避免这一问题，运维人员在系统上线时必须确保有足够的空间， 这对资源造成了很大的浪费。

MongoDB 优点:

弱一致性（最终一致），更能保证用户的访问速度文档结构的存储方式， 能够更便捷的获取数
内置 GridFS，高效存储二进制大对象 (比如照片和视频)
支持复制集、主备、互为主备、自动分片等特性
动态查询
全索引支持,扩展到内部对象和内嵌数组

MongoDB 缺点：

不支持事务
MongoDB 占用空间过大
维护工具不够成熟

## 2. 怎么解决数据库高并发的问题?

**答案:**

解决数据库高并发：

1.分表分库
2.数据库索引
3.redis 缓存数据库
4.读写分离
5.负载均衡集群：将大量的并发请求分担到多个处理节点。由于单个处理节点的故障不影响整个服务，负载均衡集群同时也实现了高可用性。

## 3. 如何执行事物/加锁?

**答案:**

MongoDB 没有使用传统的锁或者复杂的带回滚的事务，因为它设计的宗旨是轻量，快速以及可预计的高性能。可以把它类比成 MySQL MylSAM 的自动提交模式。通过精简对事务的支持，性能得到了提升，特别是在一个可能会穿过多个服务器的系统里。

## 4. 如何理解 MongoDB 中的 GridFS 机制，MongoDB 如何使用 GridFS 来存储文件?

**答案:**

GridFS 是一种将大型文件存储在 MongoDB 中的文件规范。使用 GridFS 可以将大文件分隔成多个小文档存放，这样我们能够有效的保存大文档，而且解决了 BSON 对象有限制的问题。

## 5. MongoDB 支持存储过程吗？怎么用?

**答案:**

MongoDB 支持存储过程，它是 javascript 写的，保存在 db.system.js 表中。

## 6. 如果一个分片（Shard）停止或很慢的时候，发起一个查询会怎样?

**答案:**

如果一个分片停止了，除非查询设置了“Partial”选项，否则查询会返回一个错误。如 果一个分片响应很慢，MongoDB 会等待它的响应。

## 7. 你怎么比较 MongoDB、CouchDB 及 CouchBase?

**答案:**

MongoDB 和 CouchDB 都是面向文档的数据库。MongoDB 和 CouchDB 都是开源 N oSQL 数 据 库的最典型代表。 除了都以文档形式存储外它们没有其他的共同点。Mon goDB 和 CouchDB 在数据模型实现、接口、对象存储以及复制方法等方面有很多不 同。

## 8. MongoDB 成为最好 NoSQL 数据库的原因是什么?

**答案:**

以下特点使得 MongoDB 成为最好的 NoSQL 数据库:

面向文件

高性能

高可用性

易扩展性

丰富的查询语言

## 9. 分析器在 MongoDB 中的作用是什么?

**答案:**

MongoDB 中包括了一个可以显示数据库中每个操作性能特点的数据库分析器。通过这个分析器你可以找到比预期慢的查询(或写操作);利用这一信息，比如，可以确定是否需要添加索引。

## 10. 名字空间(namespace)是什么?

**答案:**

MongoDB 存储 BSON 对象在丛集(collection)中。数据库名字和丛集名字以句点连结起来叫做名字空间(namespace)。

# 五、正则表达式

## 1. python 中用分别用什么字符来匹配字符串的开头和末尾?

**答案:**

在 python 中，匹配字符串的开头用^,匹配字符串的末尾用$。

## 2. 用两种正则的方法表示 0-9 中的任意一个数字?

**答案:**

[0-9]和\d

## 3. python 中用正则匹配任意字母和数字的万能式子是什么?

**答案:**

[a-zA-Z0-9]

## 4. python 中 re 模块中的 match 与 search 的区别?

**答案:**

re.match 只匹配字符串的开始，如果字符串开始不符合正则表达式，则匹配失败，函数返回 None； 而 re.search 匹配整个字符串，直到找到一个匹配。

## 5. findall 方法在正则匹配中起到什么样的作用?

**答案:**

在字符串中找到正则表达式所匹配的所有子串，并返回一个列表，如果没有找到匹配的，则返回空列表。

**注意：** match 和 search 是匹配一次 findall 匹配所有。

语法格式为: findall(string[,pos[,endpos]])

参数:

string : 待匹配的字符串。

pos : 可选参数，指定字符串的起始位置，默认为 0。

endpos : 可选参数，指定字符串的结束位置，默认为字符串的长度。

# 六、git

## 1. 从版本库中删除文件的时候，如果发现删错了可以用什么命令将版本库中的文件进行一键还原?

**答案:**

```
git checkout --文件名
```

## 2. 如下两条命令的区别在于什么?

### ①： $ git branch -r

### ②：$ git branch –a

**答案:**

第一条命令的命令的意思是列出所有远程分支，第二条命令的意思是列出所有本地分支和远程分支。

## 3. 请分别指出以下几个名词的译名。

### ① Workspace

### ② Index / Stage

### ③ Repository

### 4 Remote

**答案:**

① 工作区； ②暂存区； ③仓库区（或本地仓库）； ④远程仓库

## 4. 简述以下 git 的工作流程。

**答案:**

1、在工作目录中修改某些文件

2、对修改后的文件进行快照，然后保存到暂存区域

3、提交更新，将保存在暂存区域的文件快照永久转储到 Git 目录中

## 5. fetch 和 merge 和 pull 的区别

**答案:**

pull 相当于 git fetch 和 git merge，即更新远程仓库的代码到本地仓库，然后将内容合并到当前分支。

git fetch：相当于是从远程获取最新版本到本地，不会自动 merge

git merge ： 将内容合并到当前分支

git pull：相当于是从远程获取最新版本并 merge 到本地