# 专业问答

## 数据结构

1. **为什么需要数据结构**

数据结构是一种在计算机中存储和组织数据的特定的方式，其作用是使数据可以被更高效的使用。

A data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently.

1. **头指针和头结点的区别**

**头指针：**是指向第一个节点存储位置的指针，具有标识作用，头指针是链表的必要元素，无论链表是否为空，头指针都存在。

**头结点：**是放在第一个元素节点之前，便于在第一个元素节点之前进行插入和删除的操作，头结点不是链表的必须元素，可有可无，头结点的数据域也可以不存储任何信息。

1. **B树和B+树的区别**

* 在B+树中，具有n个关键字的节点只含有n棵子树，即**每个关键字对应1棵子树**；而在B树中，具有n个关键字的节点只含有n+1棵子树。
* 在B+树中，叶结点包含信息，所有非叶子结点仅起到索引作用，非叶子结点中的每个索引项只含有对应子树的最大关键字和指向该子树的指针，不包含有该关键字对应记录的存储地址。
* 在B+树中，叶结点包含了全部关键字，即在非叶结点中出现的关键字也会出现在叶节点中；而在B树中，叶结点包含的关键字和其他结点包含的关键字是不重复的。

1. **排序算法的时间复杂度？**



1. **层次遍历用到了什么数据结构**

队列

1. **满二叉树和完全二叉树的区别**
2. 一个二叉树，如果每一个层的结点数都达到最大值，则这个二叉树就是满二叉树。也就是说，如果一个二叉树的层数为K，且结点总数是(2^k) -1 ，则它就是**满二叉树**。
3. 设二叉树的深度为h，除第 h 层外，其它各层 (1～h-1) 的结点数都达到最大个数，第 h 层所有的结点都连续集中在最左边，这就是**完全二叉树**。
4. **树的遍历有哪些**

前序遍历、中序遍历、后序遍历和层序遍历。

## 计算机组成原理

1. **冯诺依曼计算机的工作原理特点**

早期冯诺依曼模型以运算器为核心，包括：运算器、控制器、存储器和输入输出设备。随着微电子技术的进步，同时计算机需要处理的信息量也与日俱增，大量I/O设备的速度和CPU的速度差距悬殊，故以运算器为中心的结构不能满足计算机发展的要求。现代计算机已发展为以存储器为中心，使I/O操作尽可能绕过CPU，直接在I/O设备和存储器之间完成，以提高系统的整体运算效率。

1. **时钟周期**

通常为节拍脉冲或T周期，即主频的倒数，它是CPU 中最小的时间单位，每个动作至少需要1 个时钟周期。

1. **引入总线结构有什么好处？**
2. 简化了系统结构，便于系统设计制造。
3. 大大减少了连线数目，便于布线，减小体积，提高系统的可靠性。
4. 便于接口设计，所有与总线连接的设备均采用类似的接口。
5. 便于系统的扩充、更新与灵活配置，易于实现系统的模块化。
6. 便于设备的软件设计，所有接口的软件对不同的接口地址进行操作。
7. 便于故障诊断和维修，同时也能降低成本。
8. **总线相关概念**
9. **系统总线按照传输信息的不同，分成哪几类？是单向的，还是双向的？**
10. 分成数据总线、地址总线以及控制总线。
11. 数据总线：各个功能部件之间传送数据信息，双向传输；
12. 地址总线：用来指明数据总线上，源数据或目的数据所在的主存单元的地址。单向：由CPU发 出；
13. 控制总线：用来发送各种控制信号。对于控制总线中的单根线，是单向的，即只能由一个部件 发向另一个部件。而一组控制总线中，有输入也有输出，因此，控制总线也可以看成是双向的。
14. **什么是总线宽度、总线带宽、总线复用、信号线数？**
15. 总线宽度：数据总线的根数，一般是8的倍数。是衡量计算机系统性能的重要指标；
16. 总线带宽：即总线数据传输速率，总线上每秒能够传输的最大字节量；
17. 总线复用：一条信号线上分时传送两种信号。例如数据总线和地址总线的分时复用；
18. 信号线数：地址总线、数据总线和控制总线三种总线的线数之和。
19. **异步通信与同步通信的主要区别是什么，说明通信双方如何联络。**

同步通信和异步通信的主要区别是前者有公共时钟，总线上的所有设备按统一的时序，统一的传输周期进行信息传输，通信双方按约定好的时序联络。后者没有公共时钟，没有固定的传输周期，采用应答方式通信，具体的联络方式有不互锁、半互锁和全互锁三种。不互锁方式通信双方没有相互制约关系；半互锁方式通信双方有简单的制约关系；全互锁方式通信双方有完全的制约关系。其中全互锁通信可靠性最高。

1. **外围设备要通过接口与CPU相连的原因主要有：**
2. 一台机器通常配有多台外设，它们各自有其设备号（地址），通过接口可实现对设备的选择；
3. I/O设备种类繁多，速度不一，与CPU速度相差可能很大，通过接口可实现数据缓冲，达到速度匹配；
4. I/O设备可能串行传送数据，而CPU一般并行传送，通过接口可实现数据串并格式转换；
5. I/O设备的入/出电平可能与CPU的入/出电平不同，通过接口可实现电平转换；
6. CPU启动I/O设备工作，要向外设发各种控制信号，通过接口可传送控制命令；
7. I/O设备需将其工作状况（“忙”、“就绪”、“错误”、“中断请求”等）及时报告CPU，通过接口可监视设备的工作状态，并保存状态信息，供CPU查询；

归纳起来，接口应具有选址的功能、传送命令的功能、反映设备状态的功能以及传送数据的功能（包括缓冲、数据格式及电平的转换）。

1. **DMA接口主要由哪些部件组成？在数据交换过程中它应完成哪些功能？**

答：DMA接口主要由数据缓冲寄存器、主存地址计数器、字计数器、设备地址寄存器、中断机构和DMA控制逻辑等组成。在数据交换过程中，DMA接口的功能有：

1. 向CPU提出总线请求信号；
2. 当CPU发出总线响应信号后，接管对总线的控制；
3. 向存储器发地址信号（并能自动修改地址指针）；
4. 向存储器发送读/写等控制信号，进行数据传送；
5. 修改字计数器，并根据传送字数，判断DMA传送是否结束；
6. 发DMA结束信号，向CPU申请程序中断，报告一组数据传送完毕。
7. **什么叫指令？什么叫指令系统？**

指令是计算机执行某种操作的命令，也就是常说的机器指令。一台机器中所有机器指令的集合，称这台计算机的指令系统。

1. **一次程序中断大致可分为哪几个阶段？**

一次程序中断大致可分为五个阶段。中断请求（1分）中断判优（1分）中断响应（1分）中断服务（1分）中断返回（1分）

1. **比较水平微指令与垂直微指令的优缺点。**
2. 水平型微指令并行操作能力强、效率高、灵活性强，垂直型微指令则较差。
3. 水平型微指令执行一条指令的时间短，垂直型微指令执行时间长。
4. 由水平型微指令解释指令的微程序，具有微指令字比较长，但微程序短的特点，而垂直型微指令正好相反。
5. 水平型微指令用户难以掌握，而垂直型微指令与指令比较相似，相对来说比较容易掌握。
6. **指令和数据均存放在内存中，计算机如何从时间和空间上区分它们是指令还是数据？**

时间上讲，取指令事件发生在“取指周期”，取数据事件发生在“执行周期”。从空间上讲，从内存读出的指令流流向控制器（指令寄存器）。从内存读出的数据流流向运算器（通用寄存器）。

1. **虚拟存储器**

**虚拟存储器的基本概念**

虚拟存储器是指具有请求调入和置换功能，能从逻辑上对主存容量加以扩存的一种存储器系统。

**页式虚拟存储器**

页式管理：是把虚拟存储空间和实际空间等分成固定大小的页，各虚拟页可装入主存中的不同实际页面位置。页式存储中，处理机逻辑地址由虚页号和页内地址两部分组成，实际地址也分为页号和页内地址两部分，由地址映射机构将虚页号转换成主存的实际页号。

页式管理用一个页表，包括页号，每页在主存中起始位置，装入位等。页表是虚拟页号与物理页号的映 射表，页式管理由操作系统进行,对应用程序员是透明的。

**段式虚拟存储器**

段式管理：把主存按段分配的存储管理方式。它是一种模块化的存储管理方式，每个用户程序模块可分到一个段，该程序模块只能访问分配给该模块的段所对应的主存空间。段长可以任意设定，并可放大和缩小。

系统中通过一个段表指明各段在主存中的位置。段表中包括段名(段号)、段起点、装入位和段长等。段表本身也是一个段，段一般是按程序模块分的。

**段页式虚拟存储器**

段页式管理:是上述两种方法的结合，它将存储空间按逻辑模块分成段,每段又分成若干个页，访存通过一个段表和若干个页表进行。段的长度必须是页长的整数倍，段的起点必须是某一页的起点。

**TLB(快表)**

在虚拟存储器中进行地址变换时，需要虚页号变换成主存中实页号的内部地址变换这一过程。

缓存时首先要到主存查页表，然后才能根据主存物理地址访问主存的存取指令或数据。因此采用虚拟存储器机制后，访存的次数增加了。为了减少访存的次数，往往将页表中最活跃的几个页表项复制到高速缓存中。这种在高速缓存中的页表项称为快表（translation look aside buffer）。

查表时，根据虚页表同时查找快表和慢表,当在快表中查到该虚页号时,就能很快找到对应的实页号，将其送入主存实地址寄存器，同时使慢表的查找作废，这时主存的访问速度没降低多少。如果在快表中查不到，则经过一个访主存的时间延迟后，将从慢表中查到的实页送入实地址寄存器，同时将此虚页号和对应的实页号送入快表。

1. **cache和寄存器有什么区别？**

**寄存器**是中央处理器内的组成部份。寄存器是有限存贮容量的高速存贮部件，它们可用来暂存指令、数据和位址。在中央处理器的控制部件中，包含的寄存器有指令寄存器(IR)和程序计数器(PC)。在中央处理器的算术逻辑单元中，包含的寄存器有累加器(ACC)。

Cache是高速缓冲存储器，是位于CPU与主内存间的一种容量较小但速度很高的存储器。其作用是解决CPU和主存之间速度差异问题。使用Cache改善系统性能的依据是程序的**局部性原理**。

1. **什么是虚短，什么是虚断？**

**虚短：**理解成短路，运放处于线性状态时，把两输入端视为等电位，即运放正输入端和负输入端的电压相等；

**虚断：**理解成断路，运放处于线性状态时，把两输入端视为开路，即流入正负输入端的电流为零；

总结一句话：虚短用来得到电压相等；虚断用来得到电流为0。

## 操作系统

1. **进程和线程的区别**
2. 进程（Process）是操作系统进行资源分配和调度的基本单位，线程（Thread）是CPU调度和分派的基本单位；
3. 线程依赖于进程而存在，一个进程至少有一个线程；
4. 进程有自己的独立地址空间，线程共享所属进程的地址空间；
5. 进程是拥有系统资源的一个独立单位，而线程自己基本上不拥有系统资源，只拥有一点在运行中必不可少的资源(如程序计数器,一组寄存器和栈)，和其他线程共享本进程的相关资源如内存、I/O、CPU等；
6. 在进程切换时，涉及到整个当前进程CPU环境的保存环境的设置以及新被调度运行的CPU环境的设置，而线程切换只需保存和设置少量的寄存器的内容，并不涉及存储器管理方面的操作，可见，进程切换的开销远大于线程切换的开销；
7. 线程之间的通信更方便，同一进程下的线程共享全局变量等数据，而进程之间的通信需要以进程间通信(IPC)的方式进行；
8. 多线程程序只要有一个线程崩溃，整个程序就崩溃了，但多进程程序中一个进程崩溃并不会对其它进程造成影响，因为进程有自己的独立地址空间，因此多进程更加健壮。
9. **进程与程序的区别**
10. 程序是永存的；进程是暂时的，是程序在数据集上的一次执行，有创建有撤销，存在是暂时的；
11. 程序是静态的观念，进程是动态的观念；
12. 进程具有并发性，而程序没有；
13. 进程是竞争计算机资源的基本单位，程序不是。
14. 进程和程序不是一一对应的：一个程序可对应多个进程即多个进程可执行同一程序；一个进程可以执行一个或几个程序。
15. **产生死锁的必要条件**

* 互斥条件：指进程对所分配到的资源进行排它性使用，即在一段时间内某资源只由一个进程占用；
* 请求和保持条件：指进程已经保持至少一个资源，但又提出了新的资源请求，而该资源已被其它进程占有，此时请求进程阻塞，但又对自己已获得的其它资源保持不放；
* 不剥夺条件：指进程已获得的资源，在未使用完之前，不能被剥夺，只能在使用完时由自己释放；
* 环路等待条件：指在发生死锁时，必然存在一个进程资源的环形链。

1. **死锁的解决方法**

一般来说，处理死锁问题有三种方法：

1. 通过协议来预防或避免死锁，确保系统不会进入死锁状态，比如说银行家算法；
2. 可以允许系统进入死锁状态，然后检测它，并加以恢复，比如说哲学家进餐问题的解决；
3. 可以忽视这个问题，认为死锁不可能在系统内发生，比如说鸵鸟算法。
4. **同步与互斥的区别**

* 互斥：是指散布在不同任务之间的若干程序片断，当某个任务运行其中一个程序片段时，其它任务就不能运行它们之中的任一程序片段，只能等到该任务运行完这个程序片段后才可以运行，最基本的场景就是对资源的同时写，为了保持资源的一致性，往往需要进行互斥访问。
* 同步：是指散布在不同任务之间的若干程序片断，它们的运行必须严格按照规定的某种先后次序来运行，这种先后次序依赖于要完成的特定的任务，最基本的场景就是任务之间的依赖，比如A任务的运行依赖于B任务产生的数据。

显然，同步是一种更为复杂的互斥，而互斥是一种特殊的同步。也就是说互斥是两个任务之间不可以同时运行，他们会相互排斥，必须等待一个线程运行完毕，另一个才能运行，而同步也是不能同时运行，但他是必须要按照某种次序来运行相应的线程（也是一种互斥）！因此互斥具有唯一性和排它性，但互斥并不限制任务的运行顺序，即任务是无序的。而同步的任务之间则有顺序关系。

总结：区别就是同步需要“按照规定的某种先后次序来运行“。

1. **并发与并行**

**并发（concurrency）：**指在同一时刻只能有一条指令执行，但多个进程指令被快速的轮换执行，使得在宏观上具有多个进程同时执行的效果，但在微观上并不是同时执行的，只是把时间分成若干段，使多个进程快速交替的执行。

**并行（parallellism）：**指在同一时刻，有多条指令在多个处理器上同时执行。

并发和并行都可以是很多个线程，就看这些线程能不能同时被(多个)CPU执行，如果可以就说明是并行,而并发是多个线程被(一个)CPU轮流切换着执行。

1. **用信号量可以做什么操作？**

用信号量及PV操作来实现进程的同步和互斥。PV操作属于进程的低级通信。

https://blog.csdn.net/wing\_7/article/details/79517768

## 计算机网络

1. **基带信号与宽带信号的传输各有什么特点？**

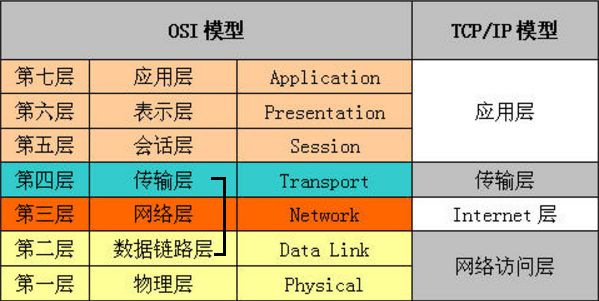
基带信号将数字1和0直接用两种不同的电压表示，然后送到线路上传输。宽带信号是将基带信号调制后形成的频分复用模拟信号。采用基带信号传输，一条电缆只能传输一路数字信号，而采用宽带信号传输，一条电缆中可同时传送多路的数字信号，提高了线路的利用率。

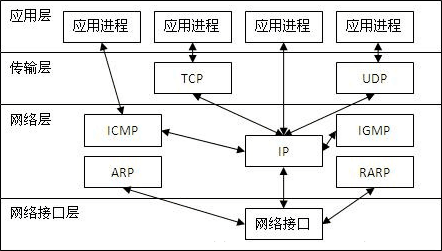
1. **对称密钥体制与公钥密码体制的特点各如何？**

在对称密钥体制中，它的加密密钥与解密密钥的密码体制是相同的，且收发双方必须共享密钥，对称密码的密钥是保密的，没有密钥，解密就不可行，知道算法和若干密文不足以确定密钥。公钥密码体制中，它使用不同的加密密钥和解密密钥，且加密密钥是向公众公开的，而解密密钥是需要保密的，发送方拥有加密或者解密密钥，而接收方拥有另一个密钥。

两个密钥之一也是保密的，无解密密钥，解密不可行，知道算法和其中一个密钥以及若干密文不能确定另一个密钥。

1. **OSI和TCP/IP模型**





1. **TCP协议**

TCP是在不可靠的IP层之上实现的可靠的数据传输协议，它主要解决传输的可靠、有序、无丢失 和不重复问题。TCP 是TCP/IP 体系中非常复杂的一个协议，主要**特点**如下：

1. TCP是面向连接的传输层协议。
2. 每条TCP连接只能有两个端点，每条TCP连接只能是点对点的（一对一）。
3. TCP 提供可靠的交付服务，保证传送的数据无差错、不丢失、不重复且有序。
4. TCP 提供全双工通信，允许通信双方的应用进程在任何时候都能发送数据，为此TCP连接的两端都设有发送缓存和接收缓存，用来临时存放双向通信的数据。发送缓存用来暂时存放以下数据：
   1. 发送应用程序传送给发送方TCP准备发送的数据；
   2. TCP已发送但尚未收到确认的数据。接收缓存用来暂时存放以下数据：
      1. 按序到达但尚未被接收应用程序收取的数据；
      2. 不按序到达的数据。

**TCP连接的建立**

在TCP 连接建立的过程中，要解决以下三个问题：

1. 要使每一方都能够确知对方的存在。
2. 要允许双方协商一些参数（如最大窗口值、是否使用窗口扩大选项、时间戳选项及服务质量等）。
3. 能够对运输实体资源（如缓存大小、连接表中的项目等）进行分配。

**三次握手建立连接**

第一步：客户机的TCP 首先向服务器的TCP 发送一个连接请求报文段。这个特殊的报文段中不含 应用层数据，其首部中的SYN标志位被置为1。另外，客户机会随机选择一个起始序号seq = x（连接请求报文不携带数据，但要消耗一个序号）。

第二步：服务器的TCP收到连接请求报文段后，如同意建立连接，就向客户机发回确认，并为该 TCP 连接分配TCP缓存和变量。在确认报文段中， SYN 和ACK 位都被置为1, 确认号字段的值为 x+ 1, 并且服务器随机产生起始序号seq= y( 确认报文不携带数据，但也要消耗一个序号）。确认 报文段同样不包含应用层数据。

第三步：当客户机收到确认报文段后，还要向服务器给出确认，并且也要给该连接分配缓存和变量。这个报文段的ACK 标志位被置1, 序号字段为x+ 1, 确认号字段ack=y+ 1。该报文段可以携带数据，若不携带数据则不消耗序号。

成功进行以上三步后，就建立了TCP 连接，接下来就可以传送应用层数据。TCP提供的是全双工通信，因此通信双方的应用进程在任何时候都能发送数据。另外，值得注意的是，服务器端的资 源是在完成第二次握手时分配的，而客户端的资源是在完成第三次握手时分配的，这就使得服务 器易于受到SYN 洪泛攻击。

**四次握手释放连接**

第一步：客户机打算关闭连接时，向其TCP发送一个连接释放报文段，并停止发送数据，主动关闭TCP 连接，该报文段的FIN标志位被置1, seq= u, 它等于前面已传送过的数据的最后一个字节的 序号加1 (FIN 报文段即使不携带数据，也要消耗一个序号）。TCP 是全双工的，即可以想象为一条TCP 连接上有两条数据通路。发送FIN报文时，发送FIN的一端不能再发送数据，即关闭了其中 一条数据通路，但对方还可以发送数据。

第二步：服务器收到连接释放报文段后即发出确认，确认号是ack = u + 1, 而这个报文段自己的序号是V, 等于它前面已传送过的数据的最后一个字节的序号加1。此时，从客户机到服务器这个方向的连接就释放了，TCP 连接处于半关闭状态。但服务器若发送数据，客户机仍要接收，即从服务器到客户机这个方向的连接并未关闭。

第三步：若服务器已经没有要向客户机发送的数据，就通知TCP释放连接，此时其发出FIN= 1的连接释放报文段。

第四步：客户机收到连接释放报文段后，必须发出确认。在确认报文段中，ACK 字段被置为1, 确 认号ack= w + 1，序号seq= u + 1。此时TCP 连接还未释放，必须经过时间等待计时器设置的时间2MSL 后，A才进入连接关闭状态。

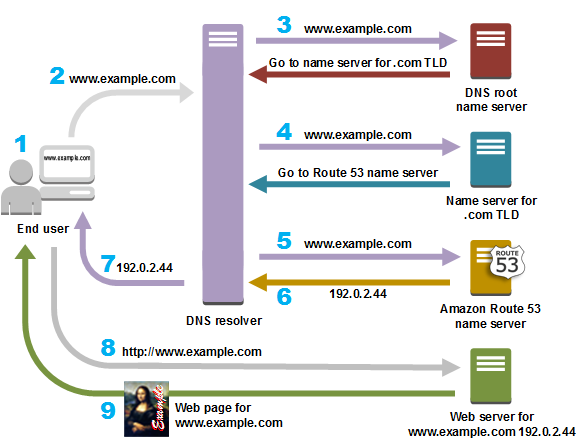
1. **说明为什么在传输连接建立时要使用三次握手。说明如不这样做可能会出现什么情况？**

我们知道，3次握手完成两个重要的功能，既要双方做好发送数据的准备工作（双方都知道彼此已准备好），也要允许双方就初始序列号进行协商，这个序列号在握手过程中被发送和确认。

现在把三次握手改成仅需要两次握手，死锁是可能发生的。作为例子，考虑计算机A和B之间的通信，假定B给A发送一个连接请求分组，A收到了这个分组，并发送了确认应答分组。按照两次握手的协定，A认为连接已经成功地建立了，可以开始发送数据分组。可是，B在A的应答分组在传输中被丢失的情况下，将不知道A是否已准备好，不知道A建议什么样的序列号，B甚至怀疑A是否收到自己的连接请求分组。在这种情况下，B认为连接还未建立成功，将忽略A发来的任何数据分组，只等待连接确认应答分组。而A在发出的分组超时后，重复发送同样的分组。这样就形成了死锁。

1. **TCP和UDP的区别**
2. TCP面向连接（如打电话要先拨号建立连接）。UDP是无连接的，即发送数据之前不需要建立连接。
3. TCP提供可靠的服务，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达。UDP尽最大努力交付，即不保证可靠交付。
4. TCP传输效率相对较低。UDP传输效率高，适用于对高速传输和实时性有较高的通信或广播通信。
5. TCP连接只能是点到点、一对一的。UDP支持一对一，一对多，多对一和多对多的交互通信。
6. **DNS服务的工作过程**
7. 在浏览器中输入www.example.com域名，操作系统会先检查自己本地的hosts文件是否有这个网址映射关系，如果有，就先调用这个IP地址映射，完成域名解析。
8. 如果hosts里没有这个域名的映射，则查找本地DNS解析器缓存，是否有这个网址映射关系，如果有，直接返回，完成域名解析。
9. 如果hosts与本地DNS解析器缓存都没有相应的网址映射关系，首先会找TCP/IP参数中设置的首选DNS服务器，在此我们叫它本地DNS服务器，此服务器收到查询时，如果要查询的域名，包含在本地配置区域资源中，则返回解析结果给客户机，完成域名解析，此解析具有权威性。
10. 如果要查询的域名，不由本地DNS服务器区域解析，但该服务器已缓存了此网址映射关系，则调用这个IP地址映射，完成域名解析，此解析不具有权威性。
11. 如果本地DNS服务器本地区域文件与缓存解析都失效，则根据本地DNS服务器的设置（是否设置转发器）进行查询，如果未用转发模式，本地DNS就把请求发至13台根DNS，根DNS服务器收到请求后会判断这个域名(.com)是谁来授权管理，并会返回一个负责该顶级域名服务器的一个IP。本地DNS服务器收到IP信息后，将会联系负责.com域的这台服务器。这台负责.com域的服务器收到请求后，如果自己无法解析，它就会找一个管理.com域的下一级DNS服务器地址(http://example.com)给本地DNS服务器。当本地DNS服务器收到这个地址后，就会找http:// example.com域服务器，重复上面的动作，进行查询，直至找到www. example.com主机。
12. 如果用的是转发模式，此DNS服务器就会把请求转发至上一级DNS服务器，由上一级服务器进行解析，上一级服务器如果不能解析，或找根DNS或把转请求转至上上级，以此循环。不管是本地DNS服务器用是是转发，还是根提示，最后都是把结果返回给本地DNS服务器，由此DNS服务器再返回给客户机。

**一个动态域名解析的流程**



1. **C/S和P2P的区别**

P2P：对等网络中的计算机之间可以互相通信和共享资源，

C/S：客户机/服务器结构的网络中可共享资源放在一台专用计算机-服务器上，工作站之间不互相直接共享资源。

P2P网络比较灵活，适用于工作组级的小型网络，当网络规模较大时，其管理和安全性都变得比较困难，此时宜采用客户机/服务器结构。

1. **计算机网络中的时延**
2. **处理时延：**差错检测、确定输出链路；
3. **排队时延：**等待输出链路可用，取决于路由器堵塞程度；
4. **传输时延：**将所有分组的比特推向链路（即传输，或者说发射）所需要的时间；
5. **传播时延：**链路起点到到终点传播所需要的时间。
6. **中继器和放大器的区别？**
7. 中继器起信号转发作用，相当于传声筒的作用，不仅仅是针对数据的转发，高级的中继例如ATM异步帧中继，还能针对会话、应用服务进行中继传送；无线网络、光纤网络是使用中继最多的网络，且中继不一定是一对一的传送，还可以是多对多传送，例如中国移动的多个基站；中继的过程中，有配套的调度协议和调度算法参与，甚至是纠错算法，传送效率提升算法；
8. 放大器只是单纯的信号放大，高级的放大器能够做大只放大信号，同时抑制噪声和其他干扰，普通的放大器是全部放大，可能导致通讯速率下降或者通讯中断。放大器对传送的信号一般是不加管理的，只是单纯的放大信号、抑制噪音和干扰。

两种设备的作用都是用于网络延伸和范围扩大，但是应用场合和原理完全不同。

## 数据库

1. **为什么会有数据库？**

文件管理系统的弊端：

1. **数据访问困难：**如果查询的条件进行了新的变化（如从查地址变成了查年龄），没有已有的程序支持数据访问。
2. **数据孤立：**数据存放在不同的地方，没有集中起来，而且存放数据的结构可能是不同的，.c \.java \.py文件等等。也就是说，没有一个程序能快速的检索到这些数据。
3. **完整性问题：**这个感觉和数据访问有点像，但是其实应该不是。数据必须要有一些约束，如人的性别应该是男女，而不能出现外星人。人的岁数必须是正整数。当我们根据某些约束，编好了程序之后。如果，添加新的约束，那么程序将很难修改。
4. **数据冗余和不一致：** 同一份数据存在了不同的地方，如一个老师必然它的数据在教务系统存了一份，在人事系统也存了一份，这样相同的数据存在多个地方，就会存储浪费，访问麻烦。而且导致数据不一致的问题，你修改了某一个文件的信息时，不一定修改了另一个的文件的信息，那么数据就不一致。

数据库是为了解决文件管理的弊端。

1. **SQL语言的四种类型**
2. 数据库定义语言(DDL)：定义数据库中Table的名称、标题(内含的属性名称及对该属性的值的要求)等；
3. 数据库操作语言(DML)：向数据库的Table中增加/删除/更新数据及对数据进行查询、检索、统计等；
4. 数据库控制语言(DCL)：控制数据库中数据的使用---哪些用户可以使用,哪些不可以；
5. 事务控制语言(TCL)：有事务的设置保存点、提交、回滚等功能。
6. **事务**

**概念：**事务指的是满足 ACID 特性的一组操作，可以通过 Commit 提交一个事务，也可以使用 Rollback 进行回滚。

**ACID特性：**

* 原子性 (Atomicity)：事务被视为不可分割的最小单元，事务的所有操作要么全部提交成功，要么全部失败回滚。回滚可以用回滚日志来实现，回滚日志记录着事务所执行的修改操作， 在回滚时反向执行这些修改操作即可。
* 一致性 (Consistency)：数据库在事务执行前后都保持一致性状态。在一致性状态下，所有事务对一个数据的读取结果都是相同的。
* 隔离性 (Isolation)：一个事务所做的修改在最终提交以前，对其它事务是不可见的。
* 持久性 (Durability)：一旦事务提交，则其所做的修改将会永远保存到数据库中。即使系统发生崩溃，事务执行的结果也不能丢失。使用重做日志来保证持久性。

事务的 ACID 特性概念简单，但不是很好理解，主要是因为这几个特性不是一种平级关系：

* 只有满足一致性，事务的执行结果才是正确的。
* 在无并发的情况下，事务串行执行，隔离性一定能够满足。此时只要能满足原子性，就一定能满足一致性。
* 在并发的情况下，多个事务并行执行，事务不仅要满足原子性，还需要满足隔离性，才能满足一致性。事务满足持久性是为了能应对数据库崩溃的情况。

1. **MySQL Oracle是基于什么数据模型?**

关系模型

**数据库的三种数据模型分别是什么？**

* 层次模型

层次模型是数据库系统最早使用的一种模型，它的数据结构是一棵“有向树”。根结点在最上端，层次最高，子结点在下，逐层排列。层次模型的特征是：

1. 有且只有一个根结点；
2. 其他结点有且仅有一个父结点。

* 网状模型

网状模型以网状结构表示实体与实体之间的联系。网中的每一个结点代表一个记录类型，联系用链接指针来实现。网状模型可以表示多个从属关系的联系，也可以表示数据间的交叉关系，即数据间的横向关系与纵向关系，它是层次模型的扩展。网状模型可以方便地表示各种类型的联系，但结构复杂，实现的算法难以规范化。其特征是：

1. 允许结点有多于一个父结点；
2. 可以有一个以上的结点没有父结点。

* 关系模型

关系模型以二维表结构来表示实体与实体之间的联系，它是以关系数学理论为基础的。关系模型的数据结构是一个“二维表框架”组成的集合。每个二维表又可称为关系。在关系模型中，操作的对象和结果都是二维表。关系模型是目前最流行的数据库模型。支持关系模型的数据库管理系统称为关系数据库管理系统，Access就是一种关系数据库管理系统。特征：

1. 描述的一致性，不仅用关系描述实体本身，而且也用关系描述实体之间的联系；
2. 可直接表示多对多的联系；
3. 关系必须是规范化的关系，即每个属性是不可分的数据项，不许表中有表；
4. 关系模型是建立在数学概念基础上的，有较强的理论依据。
5. **数据库两段锁以及实现**

两段锁协议是指每个事务的执行可以分为两个阶段：生长阶段 (加锁阶段) 和衰退阶段 (解锁阶段)。

两段封锁法可以这样来实现：事务开始后就处于加锁阶段，一直到执行ROLLBACK和COMMIT之前都是加锁阶段。ROLLBACK和COMMIT使事务进入解锁阶段，即在ROLLBACK和COMMIT模块中 DBMS释放所有封锁。

1. **数据库存储过程**

存储过程（Stored Procedure）是在大型数据库系统中，一组为了完成特定功能的SQL语句集，它存储在数据库中，一次编译后永久有效，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。存储过程是数据库中的一个重要对象。在数据量特别庞大的情况下利用存储过程能达到倍速的效率提升。

1. **数据库查询优化**
2. **使用索引**

应尽量避免全表扫描，首先应考虑在 where 及order by, group by涉及的列上建立索引。

1. **优化SQL语句**
2. **通过 explain(查询优化神器)用来查看SQL语句的执行效果**
3. **任何地方都不要使用select \* from t**
4. 索引列不能参与计算，保持列“干净”。
5. 查询尽可能使用limit减少返回的行数，减少数据传输时间和带宽浪费。
6. **优化数据库对象**
7. **硬件优化**

CPU优化、内存优化、硬盘I/O优化（使用磁盘阵列(raid)，调整磁盘调度算法）。

1. **应用优化**

使用数据库连接池、使用查询缓存。

<https://www.jianshu.com/p/3985b1f9d9b4>

1. **数据库的逻辑结构**

数据库逻辑结构是从逻辑的角度分析数据库的构成，即创建数据库后形成的逻辑概念之间的关系。

SQL的逻辑结构主要应用于面向用户的数据组织和管理，与传统关系模型术语相比，在SQL中，视图对应外模式，基本表对应模式，存储文件对应内模式。

1. **视图和基本表的区别**
2. 视图是一些SQL语句的集合，它可以查看基本数据表。
3. 表它是物理存在的，就好像成计算机中的文件。而视图它又是虚拟的内存表，就好像成Windows的快捷方式！
4. 表存在存储设备（硬盘）里，视图不会将数据存在存储设备之中，其只保存SELECT语句。我们从视图中读取数据时，视图会在内部执行该SELECT语句并创建出一张临时表。

简言之，表中存储的是实际数据，而视图中保存的是从表中取出数据所使用的SELECT语句。

