**第一章 Python综合**

## 1.对不定长参数的理解？

**不定长参数有两种：\*args和\*\*kwargs；**

**\*args：是不定长参数，用来将参数打包成tuple给函数体调用；**

**\*\*kwargs:是关键字参数，打包关键字参数成dict给函数体调用在定义函数的时候不确定要传入的参数个数会有多少个的时候就可以使用不定长参数作为形参。**

## 2.对缺省参数的理解？

**缺省参数指在调用函数的时候没有传入参数的情况下,调用默认的参数,在调用函数的同时赋值时,所传入的参数会替代默认参数。**

## 3.什么是lambda函数？它有什么好处？写一个匿名函数求两个数的和。

**lambda函数是匿名函数；使用lambda函数能创建小型匿名函数。这种函数得名于省略了用def声明函数的标准步骤；**

## f = lambda x,y:x+y

## print(f(2017,2018))

## 4.深拷贝和浅拷贝的区别？

**（1）python中的拷贝分为浅拷贝和深拷贝，若不特殊说明拷贝一般是浅拷贝**

**（2）浅拷贝是将对象顶层拷贝，拷贝了引用，并没有拷贝内容，原对象改变新对象也跟着改变**

**（3）深拷贝是对一个对象的所有层次的拷贝（递归），但是修改原来的值，新对象不受影响**

**（4）浅拷贝对于可变类型和不可变类型是不同的，对于可变类型只拷贝顶层，不可变类型依然是原来的对象**

## 5.什么是可变、不可变？元组里添加字典，会改变id吗？

**（1）可变不可变指的是内存中的值是否可以被改变，不可变类型指的是对象所在内存块里面的值不可以改变，有数值、字符串、元组；可变类型则是可以改变，主要有列表、字典**

**（2）元组的顶层元素中包含可变类型，在可变类型中修改或添加字典id不会改变**

## 6.生成器、迭代器的区别？

**（1）在Python中，一边循环一边计算的机制，称为生成器：generator，生成器是可以迭代对象，但是生成器可以通过send传值返回到前面；**

**（2）迭代器对象从集合的第一个元素开始访问，直到所有的元素被访问完结束。迭代器只能往前不会后退。**

1. **迭代器是一个可以记住遍历的位置的对象。**

## 7.range（）和xrange（）函数的用法？

**（1）range(start,stop,step)函数按照从start到stop每个step生成一个数值，生成的是列表对象，一次性将所有数据都返回；**

**（2）xrange(start,stop,step)函数按照从start到stop每个step生成一个数值，返回的是可迭代对象，每次调用返回其中的一个值**

## 8.对装饰器的理解，写出一个计时器记录方法执行性能的装饰器。

**装饰器本质上是一个Python函数，它可以让其他函数在不需要做任何代码变动的前提下增加额外功能，装饰器的返回值也是一个函数对象**

**import time**

**def timeit(func):**

**def wrapper():**

**start = time.clock()**

**end =time.clock()**

**print 'used:', end - start**

**return wrapper**

**@timeit**

**def foo():**

**print 'in foo()'foo()**

## 9.什么单例模式，其应用场景都有哪些？

**（1）确保某一个类只有一个实例，而且自行实例化并向整个系统提供这个实例，这个类称为单例类，单例模式是一种对象创建型模式。**

**Windows的Task Manager（任务管理器）、Recycle Bin（回收站）、网站计数器**

**（2）单例模式应用的场景一般发现在以下条件下：**

**资源共享的情况下，避免由于资源操作时导致的性能或损耗等。如上述中的日志文件，应用配置。**

**控制资源的情况下，方便资源之间的互相通信。如线程池等**

## 10．并行（parallel）和并发（concurrency）？线程是并发还是并行，进程是并发还是并行？

**（1）线程是并发，进程是并行；进程之间相互独立，是系统分配资源的最小单位，同一个线程中的所有线程共享资源**

**（2）并行：同一时刻多个任务同时在运行**

**（3）并发：在同一时间间隔内多个任务都在运行，但是并不会在同一时刻同时运行，存在交替执行的情况**

**（4）实现并行的库有：multiprocessing**

**（5）实现并发的库有：threading**

**（6）程序需要执行较多的读写、请求和回复任务的需要大量的IO操作，IO密集型操作使用并发更好**

**CPU运算量大的程序程序，使用并行会更好**

1. 请简单描述fork，Process，Pool个自的特点（从父进程和子进程的执行方面描述）

fork 父进程不会等子进程，父子进程都要做事；

Process 父进程会等子进程，父子进程都要做事；

Pool 父进程等子进程，父进程不做事；

## 11.什么是线程安全，什么是互斥锁？

**（1）每个对象都对应于一个可称为" 互斥锁" 的标记，这个标记用来保证在任一时刻，只能有一个线程访问该对象。**

**（2）同一个进程中的多线程之间是共享系统资源的,多个线程同时对一个对象进行操作,一个线程操作尚未结束,另一个线程已经对其进行操作,导致最终结果出现错误,此时需要对被操作对象添加互斥锁,保证每个线程对该对象的操作都得到正确的结果**

## 12.python的内存管理机制及调优手段？

**内存管理机制：引用计数、垃圾回收、内存池**

**（1）引用计数**

**引用计数是一种非常高效的内存管理手段, 当一个Python对象被引 用时其引用计数增加1, 当其不再被一个变量引用时则计数减1. 当引用计数等于0时对象被删除.**

**（2）垃圾回收**

1. **引用计数**

**引用计数也是一种垃圾收集机制，而且也是一种最直观，最简单的垃圾收集技术。当Python的某个对象的引用计数降为0时，说明没有任何引用指向该对象，该对象就成为要被回收的垃圾了。比如某个新建对象，它被分配给某个引用，对象的引用计数变为1。如果引用被删除，对象的引用计数为0，那么该对象就可以被垃圾回收。  
不过如果出现循环引用的话，引用计数机制就不再起有效的作用了**

1. **标记清除**

**如果两个对象的引用计数都为1，但是仅仅存在他们之间的循环引用，那么这两个对象都是需要被回收的，也就是说，它们的引用计数虽然表现为非0，但实际上有效的引用计数为0。所以先将循环引用摘掉，就会得出这两个对象的有效计数。**

1. **分代回收**

**从前面“标记-清除”这样的垃圾收集机制来看，这种垃圾收集机制所带来的额外操作实际上与系统中总的内存块的数量是相关的，当需要回收的内存块越多时，垃圾检测带来的额外操作就越多，而垃圾回收带来的额外操作就越少；反之，当需回收的内存块越少时，垃圾检测就将比垃圾回收带来更少的额外操作。**

**这三种情况触发垃圾回收1调用gc.collect() 2，GC达到阀值时 3，程序退出时**

**（3）内存池**

**1.Python的内存机制呈现金字塔形状，-1，-2层主要有操作系统进行操作；**

**2.第0层是C中的malloc，free等内存分配和释放函数进行操作；**

**3.第1层和第2层是内存池，有Python的接口函数PyMem\_Malloc函数实现，当对象小于256K时有该层直接分配内存；**

**4.第3层是最上层，也就是我们对Python对象的直接操作；**

**Python在运行期间会大量地执行malloc和free的操作，频繁地在用户态和核心态之间进行切换，这将严重影响Python的执行效率。为了加速Python的执行效率，Python引入了一个内存池机制，用于管理对小块内存的申请和释放。   
 Python内部默认的小块内存与大块内存的分界点定在256个字节，当申请的内存小于256字节时，PyObject\_Malloc会在内存池中申请内存；当申请的内存大于256字节时，PyObject\_Malloc的行为将蜕化为malloc的行为。当然，通过修改Python源代码，我们可以改变这个默认值，从而改变Python的默认内存管理行为。**

**（4）调优手段（了解）**

**1.手动垃圾回收**

**2.调高垃圾回收阈值**

**3.避免循环引用（手动解循环引用和使用弱引用）**

## 13．read、readline和readlines的区别

**read:读取整个文件**

**readline：读取下一行，使用生成器方法**

**readlines：读取整个文件到一个迭代器以供我们遍历**

## 14．什么是死锁以及怎么解决?

**死锁：在线程间共享多个资源的时候，如果两个线程分别占有一部分资源并且同时等待对方的资源，就会造成死锁**

**给互斥锁添加超时时间**

**程序设计时要尽量避免（银行家算法）**

## 15．解释一下什么是闭包

**在函数内部再定义一个函数，并且这个函数用到了外边函数的变量，那么将这个函数以及用到的一些变量称之为闭包**

## 16．描述Python GIL的概念，以及它对Python多线程的影响？

**GIL：全局解释器锁。每个线程在执行的过程都需要先获取GIL，保证同一时刻只有一个线程可以执行字节码。**

**线程释放GIL锁的情况：**

**在IO操作等可能会引起阻塞的system call之前,可以暂时释放GIL,但在执行完毕后,必须重新获取GIL**

**Python 3.x使用计时器（执行时间达到阈值后，当前线程释放GIL）或Python 2.x，tickets计数达到100**

**Python使用多进程是可以利用多核的CPU资源的。**

**多线程爬取比单线程性能有提升，因为遇到IO阻塞会自动释放GIL锁**

## 17．Python中有日志吗?怎么使用?

**有日志；**

**python自带logging模块，调用logging.basicConfig()方法，配置需要的日志等级和相应的参数，python解释器会按照配置的参数生成相应的日志。**

## 第二章 网络编程

## 1．Tcp协议和udp协议有什么区别？

（1）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  
 （2）TCP提供可靠的服务。也就是说，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  
 （3）TCP面向字节流，实际上是TCP把数据看成一连串无结构的字节流;UDP是面向报文的，UDP没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如IP电话，实时视频会议等）

（4）每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信  
 （5）TCP首部开销20字节;UDP的首部开销小，只有8个字节  
 （6）TCP的逻辑通信信道是全双工的可靠信道，UDP则是不可靠信道

## 2．Post和get区别？

（1）GET请求，请求的数据会附加在URL之后，以?分割URL和传输数据，多个参数用&连接。URL的编码格式采用的是ASCII编码，而不是uniclde，即是说所有的非ASCII字符都要编码之后再传输。

POST请求：POST请求会把请求的数据放置在HTTP请求包的包体中。上面的item=bandsaw就是实际的传输数据。

因此，GET请求的数据会暴露在地址栏中，而POST请求则不会。

（2）传输数据的大小

在HTTP规范中，没有对URL的长度和传输的数据大小进行限制。但是在实际开发过程中，对于GET，特定的浏览器和服务器对URL的长度有限制。因此，在使用GET请求时，传输数据会受到URL长度的限制。

对于POST，由于不是URL传值，理论上是不会受限制的，但是实际上各个服务器会规定对POST提交数据大小进行限制，Apache、IIS都有各自的配置。

（3）安全性

POST的安全性比GET的高。这里的安全是指真正的安全，而不同于上面GET提到的安全方法中的安全，上面提到的安全仅仅是不修改服务器的数据。比如，在进行登录操作，通过GET请求，用户名和密码都会暴露再URL上，因为登录页面有可能被浏览器缓存以及其他人查看浏览器的历史记录的原因，此时的用户名和密码就很容易被他人拿到了。除此之外，GET请求提交的数据还可能会造成Cross-site request frogery攻击。

## 3．cookie 和session 的区别？

**（1）cookie数据存放在客户的浏览器上，session数据放在服务器上。**

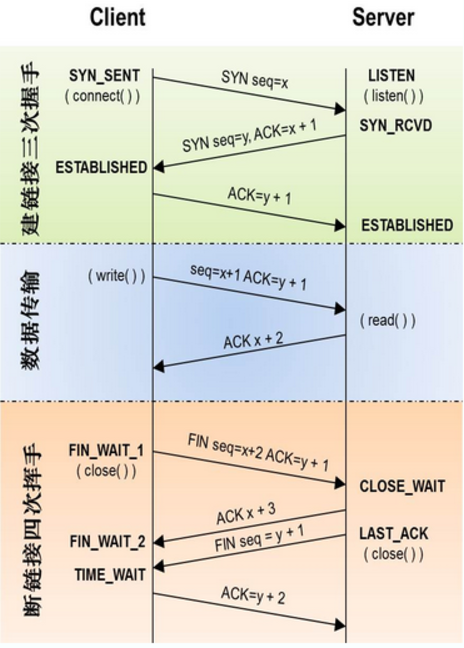
**（2）cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗考虑到安全应当使用session。**

**（3）session会在一定时间内保存在服务器上。当访问增多，会比较占用服务器的性能考虑到减轻服务器性能方面，应当使用COOKIE。**

**（4）单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。**

**（5）建议：  
   将登陆信息等重要信息存放为SESSION  
   其他信息如果需要保留，可以放在COOKIE中**

## 4．请简单说一下三次握手和四次挥手？什么是2msl？为什么要这样做？



**（1）2MSL即两倍的MSL，TCP的TIME\_WAIT状态也称为2MSL等待状态，**

**（2）当TCP的一端发起主动关闭，在发出最后一个ACK包后，即第3次握手完成后发送了第四次握手的ACK包后就进入了TIME\_WAIT状态，**

**必须在此状态上停留两倍的MSL时间，等待2MSL时间主要目的是怕最后一个 ACK包对方没收到，那么对方在超时后将重发第三次握手的FIN包，主动关闭端接到重发的FIN包后可以再发一个ACK应答包。**

**（3）在TIME\_WAIT状态 时两端的端口不能使用，要等到2MSL时间结束才可继续使用。**

**（4）当连接处于2MSL等待阶段时任何迟到的报文段都将被丢弃。**

**不过在实际应用中可以通过设置 SO\_REUSEADDR选项达到不必等待2MSL时间结束再使用此端口。**

## 5．什么是僵尸进程和孤儿进程,怎么避免僵尸进程?

**（1）孤儿进程：父进程退出，子进程还在运行的这些子进程都是孤儿进程，孤儿进程将被init进程(进 程号为1)所收养，并由init进程对它们完成状态收集工作。**

**（2）僵尸进程：进程使用fork创建子进程，如果子进程退出，而父进程并没有调用wait或waitpid获取子进程的状态信息，那么子进程的进程描述符仍然保存在系统中的这些进程是僵尸进程。**

**（3）避免僵尸进程的方法：**

**1.fork两次用孙子进程去完成子进程的任务。**

**2.用wait()函数使父进程阻塞。**

**3.使用信号量，在signal handler中调用 waitpid，这样父进程不用阻塞。**

## 6.长链接与短链接的区别？

（1）短连接

连接->传输数据->关闭连接

HTTP是无状态的，浏览器和服务器每进行一次HTTP操作，就建立一次连接，但任务结束后就中断连接。短连接是指SOCKET连接后发送后接收完数据后马上断开连接。

（2）长连接

连接->传输数据->保持连接->传输数据->....->关闭连接

长连接指建立SOCKET连接后不管是否使用都保持连接，但安全性较差。

## 7.HTTP协议状态码有什么用，都表示什么意思？

作用：告诉用户当前操作所处的状态

200 OK 服务器成功处理了请求（这个是我们见到最多的）

301/302 Moved Permanently（重定向）请求的URL已移走。Response中应该包含一个Location URL, 说明资源现在所处的位置

304 Not Modified（未修改）客户的缓存资源是最新的， 要客户端使用缓存

404 Not Found 未找到资源

501 Internal Server Error服务器遇到一个错误，使其无法对请求提供服

## 8、死锁

（1）原因: 1.竞争资源2.程序推进顺序不当

（2）必要条件：1.互斥条件2.请求和保持条件3.不剥夺条件4.环路等待条件

（3）处理死锁基本方法：

1、预防死锁(摒弃除1以外的条件)

2、避免死锁(银行家算法)

3、检测死锁(资源分配图)

（4）解除死锁:1.剥夺死锁 2.撤销进程

## 9.从输入http://www.baidu.com/到页面返回，中间都是发生了什么？

**(浏览器发送一个请求到返回一个页面的具体过程)**

**第一步，解析域名，找到ip**

**浏览器会缓存DNS一段时间，一般2-30分钟不等，如果有缓存，直接返回ip，否则下一步。**

**缓存中无法找到ip，浏览器会进行一个系统调用，查询hosts文件。如果找到，直接返回ip，否则下一步。**

**进行1 和2 本地查询无果，只能借助于网络，路由器一般都会有自己的DNS缓存，ISP服务商DNS缓存，这时一般都能够得到相应的ip，如果还是无果，只能借助于DNS递归解析了。**

**这时ISP的DNS服务器就会开始从根域名服务器开始递归搜索，从.com 顶级域名服务器，到baidu的域名服务器。   
到这里，浏览器就获得网络ip，在DNS解析过程中，常常解析出不同的IP。**

**第二步，浏览器于网站建立TCP连接**

**浏览器利用ip直接网站主机通信，浏览器发出TCP连接请求，主机返回TCP应答报文，浏览器收到应答报文发现ACK标志位为1，表示连接请求确认，浏览器返回TCP（）确认报文，主机收到确认报文，三次握手，TCP连接建立完成。**

**第三步， 浏览器发起默认的GET请求**

**浏览器向主机发起一个HTTP-GET方法报文请求，请求中包含访问的URL，也就是**[**http://www.baidu.com/**](http://www.baidu.com/)**还有User-Agent用户浏览器操作系统信息，编码等，值得一提的是Accep-Encoding和Cookies项。Accept-Encoding一般采用gzip，压缩之后传输html文件，Cookies如果是首次访问，会提示服务器简历用户缓存信息，如果不是，可以利用Cookies对应键值，找到相应缓存，缓存里面存放着用户名，密码和一些用户设置项**

**第四步，显示页面或返回其他**

**返回状态码200 OK，表示服务器可以响应请求，返回报文，由于在报头中Content-type为“text/html”,浏览器以HTML形式呈现，而不是下载文件。   
但是对于大型网站存在多个主机站点，往往不会直接返回请求页面，而是重定向。返回的状态码就不是 200 OK， 而是301,302以3开头的重定向吗。浏览器在获取了重定向响应后，在响应报文中Location项找到重定向地址，浏览器重新第一步访问即可。**

**第三章 数据库**

## 1.说一下mysql数据库存储的原理？

储存过程是一个可编程的函数，它在数据库中创建并保存。它可以有SQL语句和一些特殊的控制结构组成。当希望在不同的应用程序或平台上执行相同的函数，或者封装特定功能时，存储过程是非常有用的。数据库中的存储过程可以看做是对编程中面向对象方法的模拟。它允许控制数据的访问方式。存储过程通常有以下优点：

a、存储过程能实现较快的执行速度。

b、存储过程允许标准组件是编程。

c、存储过程可以用流程控制语句编写，有很强的灵活性，可以完成复杂的判断和较复杂的运算。

d、存储过程可被作为一种安全机制来充分利用。

e、存储过程能够减少网络流量。

## 2.事务的特性？

（1）原子性(Atomicity)：事务中的全部操作在数据库中是不可分割的，要么全部完成，要么均不执行。

（2）一致性(Consistency)：几个并行执行的事务，其执行结果必须与按某一顺序串行执行的结果相一致。

（3）隔离性(Isolation)：事务的执行不受其他事务的干扰，事务执行的中间结果对其他事务必须是透明的。

（4）持久性(Durability)：对于任意已提交事务，系统必须保证该事务对数据库的改变不被丢失，即使数据库出现故障

## 3．使用Mysql数据库中的事务的方法？

MYSQL 事务处理主要有两种方法：

1、用 BEGIN, ROLLBACK, COMMIT来实现

BEGIN 开始一个事务

ROLLBACK 事务回滚

COMMIT 事务确认

2、直接用 SET 来改变 MySQL 的自动提交模式:

SET AUTOCOMMIT=0 禁止自动提交

SET AUTOCOMMIT=1 开启自动提交

**4．数据库索引**

数据库索引，是数据库管理系统中一个排序的数据结构，以协助快速查询、更新数据库表中数据。索引的实现通常使用B\_TREE。B\_TREE索引加速了数据访问，因为存储引擎不会再去扫描整张表得到需要的数据；相反，它从根节点开始，根节点保存了子节点的指针，存储引擎会根据指针快速寻找数据。

## 5．索引怎么建立？原理是什么？

在表格上面创建某个一个唯一的索引。唯一的索引意味着两个行不能拥有相同的索引值。

CREATE UNIQUE INDEX 索引名称

ON 表名称 (列名称)

“列名称”规定你需要索引的列

原理：索引问题就是一个查找问题。

数据库索引，是数据库管理系统中一个排序的数据结构，以协助快速查询、更新数据库表中数据。索引的实现通常使用B树及其变种B+树。  
在数据之外，数据库系统还维护着满足特定查找算法的数据结构，这些数据结构以某种方式引用（指向）数据，这样就可以在这些数据结构上实现高级查找算法。这种数据结构，就是索引。

**6.在关系型数据库中，索引（index）存在的意义是什么？**

（1）通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。

（2）可以大大加快数据的检索速度，这也是创建索引的最主要的原因。

（3）可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。

（4）在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。

（5）通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。

**7.BTree索引和Hash索引的优缺点各是什么？**

（1）BTree（多路搜索树，并不是二叉的）是一种常见的数据结构。使用BTree结构可以显著减少定位记录时所经历的中间过程，从而加快存取速度。这个数据结构一般用于数据库的索引，综合效率较高。

不适合：单列索引的列不能包含null的记录，复合索引的各个列不能包含同时为null的记录，否则会全表扫描；

不适合键值较少的列（重复数据较多的列）；前导模糊查询不能利用索引(like '%XX'或者like '%XX%')

（2）Hash散列索引

Hash散列索引是根据HASH算法来构建的索引。虽然 Hash 索引效率高，但是 Hash 索引本身由于其特殊性也带来了很多限制和弊端，主要有以下这些。

适合：精确查找非常快（包括= <> 和in），其检索效率非常高，索引的检索可以一次定位，不像BTree 索引需要从根节点到枝节点，所以 Hash 索引的查询效率要远高于 B-Tree 索引。

不适合：

1、不适合模糊查询和范围查询（包括like，>，<，between……and等），由于 Hash 索引比较的是进行 Hash 运算之后的 Hash 值，所以它只能用于等值的过滤，不能用于基于范围的过滤，因为经过相应的 Hash 算法处理之后的 Hash 值的大小关系，并不能保证和Hash运算前完全一样；

2、不适合排序，数据库无法利用索引的数据来提升排序性能，同样是因为Hash值的大小不确定；

复合索引不能利用部分索引字段查询，Hash 索引在计算 Hash 值的时候是组合索引键合并后再一起计算 Hash 值，而不是单独计算 Hash 值，所以通过组合索引的前面一个或几个索引键进行查询的时候，Hash 索引也无法被利用。

同样不适合键值较少的列（重复值较多的列）；

**8.redis 和 mysql 的区别**

readis 是内存数据库， 数据保存在内存中， 速度快。

mysql 是关系型数据库，持久化存储，存放在磁盘里面，功能强大。检索的话， 会涉及到一定的 IO， 数据访问也就慢。

**9.你用的mysql是哪个引擎，各引擎之间有什么区别？**

**主要 MyISAM 与 InnoDB 两个引擎，其主要区别如下：**

InnoDB 支持事务，MyISAM 不支持，这一点是非常之重要。事务是一种高级的处理方式，如在一些列增删改中只要哪个出错还可以回滚还原，而 MyISAM就不可以了；

MyISAM 适合查询以及插入为主的应用，InnoDB 适合频繁修改以及涉及到安全性较高的应用；

InnoDB 支持外键，MyISAM 不支持；

MyISAM 是默认引擎，InnoDB 需要指定；

InnoDB 不支持 FULLTEXT 类型的索引；

InnoDB 中不保存表的行数，如 select count(*\*) from table 时，InnoDB；需要扫描一遍整个表来计算有多少行，但是 MyISAM 只要简单的读出保存好的行数即可。注意的是，当 count(\**)语句包含 where 条件时 MyISAM 也需要扫描整个表；

对于自增长的字段，InnoDB 中必须包含只有该字段的索引，但是在 MyISAM表中可以和其他字段一起建立联合索引；清空整个表时，InnoDB 是一行一行的删除，效率非常慢。MyISAM 则会重建表；

InnoDB 支持行锁（某些情况下还是锁整表，如 update table set a=1 where user like '%lee%'

**10.Redis和MongoDB的优缺点**

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

**10.数据库优化方案**

优化索引、SQL 语句、分析慢查询

设计表的时候严格根据数据库的设计范式来设计数据库

使用缓存，把经常访问到的数据而且不需要经常变化的数据放在缓存中，能节约磁盘IO

优化硬件；采用SSD，使用磁盘队列技术(RAID0,RAID1,RDID5)等；

采用MySQL 内部自带的表分区技术，把数据分层不同的文件，能够提高磁盘的读取效率

垂直分表；把一些不经常读的数据放在一张表里，节约磁盘I/O

主从分离读写；采用主从复制把数据库的读操作和写入操作分离开来

分库分表分机器（数据量特别大），主要的的原理就是数据路由

选择合适的表引擎，参数上的优化

进行架构级别的缓存，静态化和分布式

不采用全文索引

采用更快的存储方式，例如NoSql存储经常访问的数据

**11.怎样解决海量数据的存储和访问造成系统设计瓶颈的问题？**

- 水平切分数据库： 可以降低单台机器的负载， 同时最大限度的降低了宕机造成的损失； 分库降低了单点机器的负载； 分表， 提高了数据操作的效率，

- 负载均衡策略： 可以降低单台机器的访问负载， 降低宕机的可能性；

- 集群方案：解决了数据库宕机带来的单点数据库不能访问的问题；

- 读写分离策略：最大限度了提高了应用中读取数据的速度和并发量；

**12.怎样解决数据库高并发的问题？**

**解决数据库高并发：**

**分表分库**

**数据库索引**

**redis缓存数据库**

**读写分离**

**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。**

**12.数据库负载均衡**

负载均衡集群是由一组相互独立的计算机系统构成，通过常规网络或专用网络进行连接， 由路由器衔接在一起， 各节点相互协作、 共同负载、 均衡压力， 对客户端来说， 整个群集可以视为一台具有超高性能的独立服务器。

**1、 实现原理**

实现数据库的负载均衡技术，首先要有一个可以控制连接数据库的控制端。 在这里， 它截断了数据库和程序的直接连接， 由所有的程序来访问这个中间层， 然后再由中间层来访问数据库。 这样， 我们就可以具体控制访问某个数据库了，然后还可以根据数据库的当前负载采取有效的均衡策略， 来调整每次连接到哪个数据库。

**2、 实现多据库数据同步**

对于负载均衡，最重要的就是所有服务器的数据都是实时同步的。这是一个集群所必需的， 因为， 如果数不据实时、 不同步， 那么用户从一台服务器读出的数据， 就有别于从另一台服务器读出的数据， 这是不能允许的。 所以必须实现数据库的数据同步。 这样， 在查询的时候就可以有多个资源， 实现均衡。比较常用的方法是 Moebius for SQL Server 集群， Moebius for SQL Server 集群采用将核心程序驻留在每个机器的数据库中的办法， 这个核心程序称为 Moebius for SQL Server 中间件， 主要作用是监测数据库内数据的变化并将变化的数据同步到其他数据库中。 数据同步完成后客户端才会得到响应， 同步过程是并发完成的，所以同步到多个数据库和同步到一个数据库的时间基本相等； 另外同步的过程是在事务的环境下完成的， 保证了多份数据在任何时刻数据的一致性。 正因为 Moebius 中间件宿主在数据库中的创新， 让中间件不但能知道数据的变化， 而且知道引起数据变化的 SQL 语句，根据 SQL 语句的类型智能的采取不同的数据同步的策略以保证数据同步成本的最小化。数据条数很少， 数据内容也不大， 则直接同步数据数据条数很少， 但是里面包含大数据类型， 比如文本， 二进制数

据等， 则先对数据进行压缩然后再同步， 从而减少网络带宽的占用和传输所用的时间。数据条数很多，此时中间件会拿到造成数据变化的 SQL 语句， 然后对 SQL 语句进行解析， 分析其执行计划和执行成本， 并选择是同步数据还是同步 SQL 语句到其他的数据库中。此种情况应用在对表结构进行调整或者批量更改数据的时候非常有用。

**3、 优缺点优点：**

**\*\*优点：\*\***

- (1) 扩展性强： 当系统要更高数据库处理速度时， 只要简单地增加数据库服务器就 可以得到扩展。

- (2) 可维护性： 当某节点发生故障时， 系统会自动检测故障并转移故障节点的应用， 保证数据库的持续工作。

- (3) 安全性： 因为数据会同步的多台服务器上， 可以实现数据集的冗余， 通过多份数据来保证安全性。 另外它成功地将数据库放到了内网之中， 更好地保护了数据库的安全性。

- (4) 易用性： 对应用来说完全透明， 集群暴露出来的就是一个IP

**\*\*缺点：\*\***

- (1) 不能够按照 Web 服务器的处理能力分配负载。

- (2) 负载均衡器(控制端)故障， 会导致整个数据库系统瘫痪。

**13.数据库备份怎么做，服务器挂了怎么办？**

**备份数据库**

shell> mysqldump -h host -u root -p dbname >dbname\_backup.sql

**恢复数据库**

shell> mysqladmin -h myhost -u root -p create dbname

shell> mysqldump -h host -u root -p dbname < dbname\_backup.sql

**14.如何把整个数据库导出来，再导入指定数据库中**

**导出：**

mysqldump [-h主机] -u用户名 -p 数据库名 > 导出的数据库名.sql

导入指定的数据库中:

**第一种方法：**

mysqldump [-h主机] -u用户名 -p 数据库名 < 导出的数据库名.sql

**第二种方法：**

先创建好数据库,因为导出的文件里没有创建数据库的语句,如果数据库已经建好,则不用再创建。

create database example charset=utf8;（数据库名可以不一样）

**切换数据库：**

use example;

**导入指定sql文件：**

mysql>source /path/example.sql;

**15.数据库怎么优化查询效率？**

1、储存引擎选择：如果数据表需要事务处理，应该考虑使用InnoDB，因为它完全符合ACID特性。如果不需要事务处理，使用默认存储引擎MyISAM是比较明智的

2、分表分库，主从。

3、对查询进行优化，要尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引

4、应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描

5、应尽量避免在 where 子句中使用 != 或 <> 操作符，否则将引擎放弃使用索引而进行全表扫描

6、应尽量避免在 where 子句中使用 or 来连接条件，如果一个字段有索引，一个字段没有索引，将导致引擎放弃使用索引而进行全表扫描

7、Update 语句，如果只更改1、2个字段，不要Update全部字段，否则频繁调用会引起明显的性能消耗，同时带来大量日志

8、对于多张大数据量（这里几百条就算大了）的表JOIN，要先分页再JOIN，否则逻辑读会很高，性能很差。

## （新增）如果每天有5T的用户数据量需要存入mysql中，怎么优化数据库？

1、尽量使数据库一次性写入Data File   
 2、减少数据库的checkpoint 操作   
 3、程序上尽量缓冲数据，进行批量式插入与提交   
 4、减少系统的IO冲突

## （新增）如何查找MySQL中查询慢的SQL语句

1，slow\_query\_log

这个参数设置为ON，可以捕获执行时间超过一定数值的SQL语句。

2，long\_query\_time

当SQL语句执行时间超过此数值时，就会被记录到日志中，建议设置为1或者更短。