**Java基础**

1. Vector，ArrayList和LinkedList区别在什么地方？

答：ArrayList是通过数组实现的，可以通过下标索引直接查找到指定位置的元素，因此查找效率高，但每次插入或删除元素，就要大量地移动元素，插入删除元素的效率低。

LinkedList是一个双向链表，查找效率不如ArrayList高，但是插入和删除元素的效率。

ArrayList有自动扩容机制，当数组中添加元素之后，会自动检查添加元素后的个数是否超过了当前数组的大小，如果超过了，会自动将数组进行扩容1.5倍，并将原数组中的元素拷贝到新数组中。Vector和ArrayList都是通过数组实现的，区别在于，Vector实现了同步机制，是线程安全的，而ArrayList不是。

1. Abstract类的特性
2. 抽象类只能作为其他类的基类。
3. 抽象类允许包含抽象成员，抽象类中可以有非抽象方法。
4. 抽象类不能同时又是final的，因为final类不能被继承，而抽象类一定要被继承。
5. 如果非抽象类继承一个抽象类，其必须通过覆盖来实现所有继承而来的抽象成员。如果抽象类继承一个抽象类，其不必覆盖抽象成员。
6. 有抽象方法的一定是抽象类，但抽象类中不一定有抽象方法。

abstract class Vehicle{

public int wheels;

public float weight;

public Vehicle(int w, float g){

wheels = w;

weight = g;

}

}

class Car extends Vehicle{

int passengers;

public Car(int w, float g, int p){

super(w,g);

//wheels = w;

//weight = g;

passenger = p;

}

}

1. 接口的特性
2. 接口中的成员常量和方法都必须是public类型的，确保外部使用者能访问它们。
3. 接口中的方法都是abstract方法，没有构造方法，只有静态变量
4. 接口中的成员都是final的，为所有实现类共享。
5. 抽象类与接口的其中一个区别是：抽象类中可以定义一些子类的公共方法，子类只需要增加新的功能，不需要重复写已经存在的方法；而接口中只是对方法的申明和常量的定义。

4、String，StringBuffer，StringBuilder的区别

答：（1）可变与不可变

　　String类中使用字符数组保存字符串，如下就是，因为有“final”修饰符，所以可以知道string对象是不可变的。

　　　　private final char value[];

　　StringBuilder与StringBuffer都继承自AbstractStringBuilder类，在AbstractStringBuilder中也是使用字符数组保存字符串，如下就是，可知这两种对象都是可变的。

　　　　char[] value;

（2）是否多线程安全

　　String中的对象是不可变的，也就可以理解为常量，显然线程安全。

　　StringBuffer对方法加了同步锁或者对调用的方法加了同步锁，所以是线程安全的。

　　StringBuilder并没有对方法进行加同步锁，所以是非线程安全的。

5、为什么Java程序比C系程序执行效率低？

答：因为C程序是直接运行在机器上的，其语言的设计也是更贴近机器语言的，可以直接操作内存。而Java程序编译后产生的.class文件是运行在JVM上的，由JVM解译之后再在机器上运行，中间的JVM需要耗费比较大的内存空间，因此Java的执行效率不如C高。

6、HashMap和HashSet的区别

答：HashSet实现Set接口，HashMap实现Map接口。对象存储在HashSet之前要先确保对象重写equals()和hashCode()方法，这样才能比较对象的值是否相等，以确保set中没有存储相等的对象。如果没有重写这两个方法，将会使用这个方法的默认实现。HashSet存储对象，HashMap存储键值对。HashMap使用键对象来计算hashcode的值，HashSet使用对象来计算hashcode的值，两个对象是否相同是用equals()方法来判断的。

7、HashMap的工作原理

答：HashMap是基于hashing原理，当我们给put()方法传递键值对时，我们先对键对象调用hashCode()方法，返回的hashCode用于找到bucket位置来存储Entry对象，bucket中存储的是键对象和值对象。如果两个键对象的hashcode相同，那么它们的bucket位置也相同，会发生碰撞，但是因为HashMap是用链表存储对象的，因此碰撞的键值对会存储在同样的bucket下的链表中。在调用get()方法时，会调用keys.equals()方法去找到链表中正确的节点，进而找到要找的值对象。为避免碰撞的发生，提高HashMap的性能，应该多用final类型的对象作为键，比如String，Integer，因为它们都是不可变的，且重写了equals()和hashCode()方法，可减少产生相同的hashcode的概率。

8、如果HashMap的大小超过了负载因子定义的容量，怎么办？

答：HashMap和ArrayList一样，默认的负载因子是0.75，当75%的bucket被填满时，将会创建原来大小的两倍的bucket数组，来重新调整map大小，并将原来的对象放入新的bucket数组中

9、如何对ArrayList中的元素进行排序？

答：如果其中存储的是一般的元素，那么直接调用Collections接口的sort()方法即可。如果是自定义的对象，通过自定义排序类实现Comparator接口，并在该类中重写compare()方法，然后将自定义排序类对象以参数传入Collections.sort()方法中，如：

Collections.sort(studentList, new SortByAge());

class SortByAge implements Comparator {  
public int compare(Object o1, Object o2) {  
Student s1 = (Student) o1;  
Student s2 = (Student) o2;  
if (s1.getAge() > s2.getAge())  
 return 1;  
   return 0;

}

10、多线程的几个状态？

答： 新建状态(New)：   
        当用new操作符创建一个线程时， 例如new Thread(r)，线程还没有开始运行，此时线程处在新建状态。 当一个线程处于新生状态时，程序还没有开始运行线程中的代码  
  
     就绪状态(Runnable)

        一个新创建的线程并不自动开始运行，要执行线程，必须调用线程的start()方法。当线程对象调用start()方法即启动了线程，start()方法创建线程运行的系统资源，并调度线程运行run()方法。当start()方法返回后，线程就处于就绪状态。

        处于就绪状态的线程并不一定立即运行run()方法，线程还必须同其他线程竞争CPU时间，只有获得CPU时间才可以运行线程。因为在单CPU的计算机系统中，不可能同时运行多个线程，一个时刻仅有一个线程处于运行状态。因此此时可能有多个线程处于就绪状态。对多个处于就绪状态的线程是由[Java](http://lib.csdn.net/base/javase" \t "http://blog.csdn.net/peter_teng/article/details/_blank" \o "Java SE知识库)运行时系统的线程调度程序(thread scheduler)来调度的。  
  
    运行状态(Running)

        当线程获得CPU时间后，它才进入运行状态，真正开始执行run()方法.

阻塞状态(Blocked)

        线程运行过程中，可能由于各种原因进入阻塞状态:  
        1>线程通过调用sleep方法进入睡眠状态；  
        2>线程调用一个在I/O上被阻塞的操作，即该操作在输入输出操作完成之前不会返回到它的调用者；  
        3>线程试图得到一个锁，而该锁正被其他线程持有；  
        4>线程在等待某个触发条件；  
        ......             
  
        所谓阻塞状态是正在运行的线程没有运行结束，暂时让出CPU，这时其他处于就绪状态的线程就可以获得CPU时间，进入运行状态。  
  
     死亡状态(Dead)

        有两个原因会导致线程死亡：  
        1) run方法正常退出而自然死亡，  
        2) 一个未捕获的异常终止了run方法而使线程猝死。  
        为了确定线程在当前是否存活着（就是要么是可运行的，要么是被阻塞了），需要使用isAlive方法。如果是可运行或被阻塞，这个方法返回true； 如果线程仍旧是new状态且不是可运行的， 或者线程死亡了，则返回false.

11、sleep和wait的区别

答：sleep指线程被调用时，占着CPU不工作，此时，系统的CPU部分资源被占用，其他线程无法进入，会增加时间限制。Sleep会自动唤醒，但需要其他占用锁的线程执行完毕。  
wait指线程处于进入等待状态，此时线程不占用任何资源，不增加时间限制，必须手动唤醒。

12、Sychronized和Lock的不同

答：如果一个代码块被synchronized修饰了，当一个线程获取了对应的锁，并执行该代码块时，其他线程便只能一直等待，等待获取锁的线程释放锁，而这里获取锁的线程释放锁只会有两种情况：

　　1）获取锁的线程执行完了该代码块，然后线程释放对锁的占有；

　　2）线程执行发生异常，此时JVM会让线程自动释放锁。

　　那么如果这个获取锁的线程由于要等待IO或者其他原因（比如调用sleep方法）被阻塞了，但是又没有释放锁，其他线程便只能干巴巴地等待，影响程序执行效率。

　　因此就需要有一种机制可以不让等待的线程一直无期限地等待下去（比如只等待一定的时间或者能够响应中断），通过Lock就可以办到。

　　再举个例子：当有多个线程读写文件时，读操作和写操作会发生冲突现象，写操作和写操作会发生冲突现象，但是读操作和读操作不会发生冲突现象。

　　但是采用synchronized关键字来实现同步的话，就会导致一个问题：

　　如果多个线程都只是进行读操作，所以当一个线程在进行读操作时，其他线程只能等待无法进行读操作。

　　因此就需要一种机制来使得多个线程都只是进行读操作时，线程之间不会发生冲突，通过Lock就可以办到。

　　另外，通过Lock可以知道线程有没有成功获取到锁。这个是synchronized无法办到的。

　　总结一下，也就是说Lock提供了比synchronized更多的功能。但是要注意以下几点：

　　1）Lock不是Java语言内置的，synchronized是Java语言的关键字，因此是内置特性。Lock是一个类，通过这个类可以实现同步访问；

　　2）Lock和synchronized有一点非常大的不同，采用synchronized不需要用户去手动释放锁，当synchronized方法或者synchronized代码块执行完之后，系统会自动让线程释放对锁的占用；而Lock则必须要用户去手动释放锁，如果没有主动释放锁，就有可能导致出现死锁现象。

13、堆和栈的区别  
（1） 栈空间由操作系统分配和释放，堆由程序分配和释放。  
（2）栈空间需要提前申请大小，申请完后大小不可变，速度较快；堆空间可动态申请，速度较慢。  
（3） 数据结构不同，栈是先进后出的线性数据结构，堆的结构是树。

14、垃圾回收器有哪几种？

答：Java有四种类型的垃圾回收器：

串行垃圾回收器（Serial Garbage Collector）

并行垃圾回收器（Parallel Garbage Collector）

并发标记扫描垃圾回收器（CMS Garbage Collector）

G1垃圾回收器（G1 Garbage Collector）

（1）串行垃圾回收器

串行垃圾回收器通过持有应用程序所有的线程进行工作。它为单线程环境设计，只使用一个单独的线程进行垃圾回收，通过冻结所有应用程序线程进行工作，所以可能不适合服务器环境。它最适合的是简单的命令行程序。

通过JVM参数-XX:+UseSerialGC可以使用串行垃圾回收器。

（2）并行垃圾回收器

并行垃圾回收器也叫做 throughput collector 。它是JVM的默认垃圾回收器。与串行垃圾回收器不同，它使用多线程进行垃圾回收。相似的是，当执行垃圾回收的时候，它也会冻结所有的应用程序线程。

（3）并发标记扫描垃圾回收器

并发标记垃圾回收使用多线程扫描堆内存，标记需要清理的实例并且清理被标记过的实例。并发标记垃圾回收器只会在下面两种情况持有应用程序所有线程。

当标记的引用对象在tenured区域；

在进行垃圾回收的时候，堆内存的数据被并发的改变。

相比并行垃圾回收器，并发标记扫描垃圾回收器使用更多的CPU来确保程序的吞吐量。如果我们可以为了更好的程序性能分配更多的CPU，那么并发标记扫描垃圾回收器是更好的选择。

通过JVM参数 XX:+USeParNewGC 打开并发标记扫描垃圾回收器。

（4）G1垃圾回收器

G1垃圾回收器适用于堆内存很大的情况，他将堆内存分割成不同的区域，并且并发的对其进行垃圾回收。G1也可以在回收内存之后对剩余的堆内存空间进行压缩。并发扫描标记垃圾回收器在STW情况下压缩内存。G1垃圾回收会优先选择第一块垃圾最多的区域

15、介绍几种主流的垃圾回收机制？

答：跟踪回收  
　　跟踪回收的方式独立于程序，定期运行来检查垃圾，需要较长时间的中断。  
　　标记清除  
　　标记清除的方式需要对程序的对象进行两次扫描，第一次从根（Root）开始扫描，被根引用了的对象标记为不是垃圾，不是垃圾的对象引用的对象同样标记为不是垃圾，以此递归。所有不是垃圾的对象的引用都扫描完了之后。就进行第二次扫描，第一次扫描中没有得到标记的对象就是垃圾了，对此进行回收。  
　　复制收集  
　　复制收集的方式只需要对对象进行一次扫描。准备一个「新的空间」，从根开始，对对象进行扫，如果存在对这个对象的引用，就把它复制到「新空间中」。一次扫描结束之后，所有存在于「新空间」的对象就是所有的非垃圾对象。

16、为什么需要线程池

目前的大多数网络服务器，包括Web服务器、Email服务器以及[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "http://blog.csdn.net/suifengpiao_2011/article/details/_blank)服务器等都具有一个共同点，就是单位时间内必须处理数目巨大的连接请求，但处理时间却相对较短。

传统多线程方案中我们采用的服务器模型则是一旦接受到请求之后，即时创建，即时销毁。尽管与创建进程相比，创建线程的时间已经大大的缩短，但是如果提交给线程的任务是执行时间较短，而且执行次数极其频繁，那么服务器将处于不停的创建线程，销毁线程的状态。

我们将传统方案中的线程执行过程分为三个过程：T1、T2、T3。

T1：线程创建时间

T2：线程执行时间，包括线程的同步等时间

T3：线程销毁时间

那么我们可以看出，线程本身的开销所占的比例为(T1+T3) / (T1+T2+T3)。如果线程执行的时间很短的话，这比开销可能占到20%-50%左右。如果任务执行时间很频繁的话，这笔开销将是不可忽略的。

除此之外，线程池能够减少创建的线程个数。通常线程池所允许的并发线程是有上界的，如果同时需要并发的线程数超过上界，那么一部分线程将会等待。

因此线程池的出现正是着眼于减少线程池本身带来的开销。线程池采用预创建的技术，在应用程序启动之后，将立即创建一定数量的线程(N1)，放入空闲队列中。这些线程都是处于阻塞（Suspended）状态，不消耗CPU，但占用较小的内存空间。当任务到来后，缓冲池选择一个空闲线程，把任务传入此线程中运行。当N1个线程都在处理任务后，缓冲池自动创建一定数量的新线程，用于处理更多的任务。在任务执行完毕后线程也不退出，而是继续保持在池中等待下一次的任务。当系统比较空闲时，大部分线程都一直处于暂停状态，线程池自动销毁一部分线程，回收系统资源。

基于这种预创建技术，线程池将线程创建和销毁本身所带来的开销分摊到了各个具体的任务上，执行次数越多，每个任务所分担到的线程本身开销则越小，不过我们另外可能需要考虑进去线程之间同步所带来的开销。

1. 如何创建多线程池？

答：（1）固定大小的线程池：  
  import java.util.concurrent.Executors;  
  import java.util.concurrent.ExecutorService;  
   
  public class JavaThreadPool {  
    public static void main(String[] args) {  
        ExecutorService pool = Executors.newFixedThreadPool(3); //指定线程池的大小  
        Thread t1 = new MyThread();  
        Thread t2 = new MyThread();  
        Thread t3 = new MyThread();  
       
        pool.execute(t1);  
        pool.execute(t2);  
        pool.execute(t3);  
   
        pool.shutdown();        // 关闭线程池  
    }  
 }

ExecutorService的功能没有想像中的那么好，而且最多只是提供一个线程的容器而已，

（2）用java.lang.ThreadGroup，ThreadGroup有很多优势，最重要的一点就是它可以对线程进行遍历，知道哪些线程已经运行完毕，还有哪些线程在运行。如下：

class MyThread extends Thread {  
  boolean stopped;  
   
  MyThread(ThreadGroup tg, String name) {  
    super(tg, name);  
    stopped = false;  
  }  
   
  public void run() {  
    ……  
  }  
   
  synchronized void myStop() {  
    stopped = true;  
  }  
 }  
   
 public class Main {  
  public static void main(String args[]) throws Exception {  
    ThreadGroup tg = new ThreadGroup("My Group");  
   
    MyThread thrd1 = new MyThread(tg, "MyThread \_1");  
    MyThread thrd2 = new MyThread(tg, "MyThread \_2");  
    MyThread thrd3 = new MyThread(tg, "MyThread \_3");  
   
    thrd1.start();  
    thrd2.start();  
    thrd3.start();  
   
    Thread.sleep(1000);  
   
    System.out.println(tg.activeCount() + " threads in thread group.");  
   
    Thread thrds[] = new Thread[tg.activeCount()];  
    tg.enumerate(thrds); //将tg线程组中活动的线程复制到thrds数组中  
    for (Thread t : thrds)  
      System.out.println(t.getName());  
    thrd1.myStop(); //设置线程中断标志位为true  
    Thread.sleep(1000);  
     System.out.println(tg.activeCount() + " threads in tg.");  
    tg.interrupt(); //interrupt使线程从阻塞状态恢复到运行状态（修改标志位为false）  
  }  
 }

ThreadGroup比ExecutorService多以下几个优势

a.ThreadGroup可以遍历线程，知道那些线程已经运行完毕，那些还在运行

b.可以通过ThreadGroup.activeCount知道有多少线程从而可以控制插入的线程数

17、Spring的两个主要机制

答：（1）IOC：将所有的组件拉取到外部容器中，由这个容器进行统一的管理和装配，以达

到降低组件之间耦合的目的。

1. AOP：将分布于各个系统中的服务提取为一个可重用模块，即切面，然后将该模块以注解的形式加入到需要影响的系统中去，不需要在每个系统中都重复定义。便于减少系统的重复代码，降低模块之间的耦合度，并有利于未来的可操作性和可维护性。AOP的作用在于分离系统中的各种关注点，将核心关注点和横切关注点分离开来。AOP 主要是利用代理模式的技术来实现的
2. 拦截器与过滤器的区别

答：（1）拦截器是基于[Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "http://blog.csdn.net/chenleixing/article/details/_blank)的反射机制的，而过滤器是基于函数回调。  
　　（2）拦截器不依赖与servlet容器，过滤器依赖与servlet容器。  
　　（3）拦截器只能对action请求起作用，而过滤器则可以对几乎所有的请求起作用。  
　　（4）拦截器可以访问action上下文、值栈里的对象，而过滤器不能访问。  
　　（5）在action的生命周期中，拦截器可以多次被调用，而过滤器只能在容器初始化时被调用一次。

　　（6）拦截器是spring框架下的组件，可以获取IOC容器中的各个bean，而过滤器就不行，在拦截器里注入一个service，可以调用业务逻辑。

1. 对Spring MVC的理解

答：Model：模型应该包含由视图显示的数据。在J2EE Web应用中，数据通常应该由普通的Javabean组成。一旦一个控制器选择了视图，模型就要包含视图相应的数据。模型本身不应该进一步的访问数据，也不应该和业务对象相联系。

View：视图负责显示出模型包含的信息，视图不必了解控制器或是底层业务对象的具体实现。

Controller：控制器就好像MVC里的中枢神经，它也许会需要一些助手来帮助它比如解析视图，解析参数等.控制器可以访问到业务对象或者是它的代理是很重要的，比如Struts里的Action。

19、Spring MVC的解析过程：

 1）首先是用户发送请求，比如在浏览器中输入http://localhost:8080/SpringTest/login

 2）用户的请求就发送到了DispatcherServlet也就是前端控制器，这个东西是一个Servlet的实现，用来干什么呢？用来接收用户的请求，然后响应结果，相当于一个转发器，它的配置如下

1. 经过DispatcherServlet过滤后URL后，比如上述的URL是http://localhost:8080/SpringTest/login，那么这时候就进入到了HandleMapping也就是映射处理器。这是用来干什么的呢？就是用来根据URL来匹配我们的处理器（或者就做控制器），比如我们请求的路径最后结尾的字母是login，那么让那个控制器来为我们服务呢？这时候就靠HandleMapping来处理了。
2. 通过映射处理器就找到了我们需要处理用户请求的控制器也就是Handler，那么此时还需要HandlerAdapter（处理器适配器）来，作用就是要按照特定规则（HandlerAdapter要求的规则）去执行Handler。
3. 经过适配器调用控制器的方法后，就会想前端控制器返回一个ModelView，包含我们需要信息的pojo对象和视图信息。
4. 前端控制器开始解析视图信息，通过视图解析器（View Resolver），进行视图解析，根据逻辑视图名解析成真正的视图（view）。
5. 解析完视图信息后，就把modelview中的model信息传递到我们的view界面上，整个流程就处理完毕。

20、session与cookie的区别

答：它们都是用来存储用户信息的。Cookie存放在客户端（浏览器上），session存放在服务端。

20．Error与Exception的区别

答：Error类和Exception类都继承自Throwable类。

二者的不同之处：

Exception：

1．可以是可被控制(checked) 或不可控制的(unchecked)。

2．表示一个由程序员导致的错误。

3．应该在应用程序级被处理。

Error：

1．总是不可控制的(unchecked)。

2．经常用来用于表示系统错误或底层资源的错误。

3．如何可能的话，应该在系统级被捕捉。

Java 中定义了两类异常：

1）checked和unchecked。checked Exception就是在写代码的时候，[IDE](https://www.baidu.com/s?wd=IDE&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzPym4mWf3uHcLnANbrjFB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3Erjbvn1f4ns" \t "_blank)(比如Eclipse)会要求你写try catch的那种Exception，比如IOException。这种Exception是Java的设计者要求你的程序去处理的。这种异常一般不会影响程序的主体，容易手动诊断修复，所以Java要求你在catch下面写出处理的代码，以保证程序遇到此类exception之后还可以正常运行。

2）unchecked这一类就是你在代码处理了checked exception之后，你在运行时候依然会遇到的exception，所以又叫做RunTimeException，比如NullPointerException, IndexOutOfBoundsException。此类exception相较于前面那种更容易影响程序运行，从设计者角度不提倡从程序中catch出来并处理，当然你也可以这么做。

**计算机网络**

1. 一、ICMP协议是什么  
   1. ICMP协议全称Internet Control Message Protocol，即Internet控制报文协议。  
   2. 是TCP/IP协议的一个子族，网络层。  
   3. 用于在计算机、路由器之间传递控制消息，如网络通不通、主机是否可达、路由是否可用等等。  
   4. ping命令即基于ICMP协议。  
   5. 路由跟踪的Tracert等命令也基于ICMP  
     
   二、DNS协议是什么  
   1. DNS全称Domain Name System，即域名系统，一个用于映射域名和IP地址的分布式系统。  
   2. DNS请求使用UDP协议，但本身属于应用层协议。  
     
   四、TCP和UDP的区别  
   1. TCP是传输控制协议，面向字节流，提供拥塞控制功能；UDP是用户数据报协议，面向报文，尽最大努力交付数据；  
   2. TCP是面向连接的协议，UDP是面向非连接的协议。  
   3. TCP是传输可靠型协议，UDP是传输不可靠型协议。  
   4. TCP的速度较慢，UDP的速度较快。  
     
   五、TCP拥塞控制逻辑  
   1. 主要由拥塞窗口cwnd实现，具体逻辑主要包括慢启动、快速重传和快速恢复等。  
   2. 慢启动：窗口一开始很小（一般为1）但是很快地（指数级）增长上来，直到达到慢启动门限。  
   3. 快速重传：TCP利用3个相同的ACK来判定数据包丢失，开始快去重传  
   4. 快速启动：重传后的慢启动  
     
   六、TCP为什么建立连接时三次握手，断开连接要四次握手？  
   答：建立连接时，客户端先向服务端发送一个含有SYN（建立连接请求）的报文，服务端收到报文后向客户端发送一个含有ACK（应答）和SYN的报文，客户端收到后再向服务端发送一个ACK报文。

释放连接时，客户端先向服务端发送一个含有FIN（断开连接请求）的报文，服务端收到报文后向客户端发送一个ACK报文，待服务端数据全部发送完毕后，继续向客户端发送一个FIN报文，客户端收到后再返回一个ACK报文。

TCP是全双工的，每一个方向都必须单独进行开关，所以需要四次握手。而建立连接时发起者A的两个方向是默认打开的，B可以省去一个通知A打开的请求，所以只需要三次握手。

**数据库**

一、（1）查询名为李四的同学的考试科目（c\_name）和考试成绩（grade）。科目和成绩都存储在score表中。但是socre表中只有学生的学号stu\_id，没有学生的姓名。学生的姓名name都存在student表中，student表中的id与score表中的student\_id是对应的。

SELECT c\_name , grade FROM score where stu\_id=(select id from student where name='李四');

（2）查询所有学生的信息和考试信息

SELECT student.id, name, sex, birth, department, address, c\_name, grade

FROM student, score

WHERE student.id=score.stu\_id; 用内连接查询

（3）查询每个学生的学号、姓名和总成绩。

Select student.id , name , sum(grade) form score, student where student.id=score.stu\_id group by student.id;

（4）计算每个考试科目的平均成绩

SELECT c\_name, AVG(grade) FROM score GROUP BY c\_name;

（5）查询计算机成绩低于95的学生的id,name,sex信息。

SELECT student.id , name , sex from student

where id in(select stu\_id from score where c\_name='计算机' and grade<95 )； 请注意，应为子查询的结果不止一条，所以外面必须用in

（6）查询同时参加计算机和英语考试的学生信息

Select \* from student where id in(select stu\_id from score where c\_name=’计算机’ and c\_name=’英语’);

（7）从student表和score表中查询出学生的学号，然后合并查询结果

Select id from student

UNION

select stu\_id from score;

（8）查询姓张或者姓王的同学的姓名、院系、考试科目和成绩。

SELECT student.id, name,sex,birth,department, address, c\_name,grade

FROM student, score

WHERE

(name LIKE '张%' OR name LIKE '王%')

AND

student.id=score.stu\_id ;

二、Redis数据库的基本原理

答：Redis是REmote DIctionary Server（远程字典服务器）的缩写，它使用字典结构存储数据，Redis字典中的键值除了可以是字符串外，还可以是其他数据类型，比如：字符串、散列、列表、集合、有序集合。

Redis数据库中的所有数据都存储在内存中，由于内存的读写速度远快于硬盘，性能明显优势。Redis也提供了对持久化的支持，即可以将内存中的数据异步写入到硬盘中，同时不影响继续提供服务。

Redis可以用作缓存、队列系统等。

Redis可以为每个键设置生存时间（TTL），生存时间到期后键会自动被删除，这一功能配合出色的性能让Redis可以作为缓存系统来使用，成为了缓存系统Memcached的有力竞争者。作为缓存系统，Redis可以限定数据占用的最大内存空间，在数据达到空间限制后可以按照一定的规则自动淘汰不需要的键。

除此之外，Redis的列表类型键可以实现队列，并且支持阻塞式读取，可以很容易实现一个高性能的优先队列。

三、Hbase数据库的基本原理

答：HBase是一个构建在HDFS上的分布式列存储系统，典型的key/value型数据库，HBase将数据按照表、行和列进行存储。HBase本身从功能上可以分为三块：Zookeeper、Master和RegionServer群。

**RS定位过程：**

首先，Client通过访问ZK来请求目标数据的地址。过程是一个三层索引结构，从ZK获取-ROOT-表信息，再从-ROOT-表中获取.META.表信息，最后从.META.表中查到RS地址后缓存。Client端获取到RS地址后，然后直接向该地址发送数据请求。

【这里有几个问题：

1. 为什么不直接把.META.表信息直接保存到ZK中？主要是为了保存的数据量考虑，ZK中不宜保存大量数据，而.META.表主要是保存Region和RS的映射信息，region的数量没有具体约，如果保存在ZK中，ZK的压力会很大。所以，通过一个-ROOT-表来转存到RS中是一个比较理想的方案，相比直接保存在ZK中，也就多了一层-ROOT-表的查询，对性能来说影响不大。  
    b.每次访问都需要走ZK –> -ROOT- —> .META.的流程么？当然不需要，Client端有缓存，第一次查询到相应region所在RS后，这个信息将被缓存到Client端，以后每次访问都直接从缓存中获取RS地址即可。当访问的region在RS上发生了改变，通过缓存的地址访问会出现异常，Client需要重新走一遍上面的流程来获取新的RS地址。总体来说，region的变动只会在极少数情况下发生，一般变动不会很大，所以在整个集群访问过程中，影响可以忽略。】

**RS的数据操作过程：**

Client写入 -> 存入MemStore（数据缓存），一直到MemStore满 -> Flush成一个StoreFile，直至增长到一定阈值 -> 触发Compact合并操作 -> 多个StoreFile合并成一个StoreFile，同时进行版本合并和数据删除 -> 当StoreFiles Compact后，逐步形成越来越大的StoreFile -> 单个StoreFile大小超过一定阈值后，触发Split操作，把当前Region Split成2个Region，Parent Region会下线，新Split出的2个孩子Region会被HMaster分配到相应的RegionServer上，使得原先1个Region的压力得以分流到2个Region上。

向指定RS的对应region进行数据写入，HBase的数据先写入Hlog，后写数据，以防止当Region崩溃时数据丢失。

向指定RS的对应region进行数据写入，HBase的数据先写入Hlog，后写数据，以防止当Region崩溃时数据丢失。

二．关系型数据库和非关系型数据库有什么区别？

答：关系型[数据库](http://lib.csdn.net/base/mysql)，是指采用了关系模型来组织数据的数据库。

简单来说，关系模型指的就是二维表格模型，而一个关系型数据库就是由二维表及其之间的联系所组成的一个数据组织。

非关系型数据库，以键值对存储，且结构不固定，每一个元组可以有不一样的字段，每个元组可以根据需要增加一些自己的键值对。

非关系型数据库的优势：  
1）性能：NOSQL是基于键值对的，可以想象成表中的主键和值的对应关系，而且不需要经过SQL层的解析，所以性能非常高。key-value数据库的主要特点是具有极高的并发读写性能，可以在海量的数据中快速的查询数据。  
2）可扩展性  
同样也是因为基于键值对，数据之间没有耦合性，所以非常容易水平扩展。

不会局限于固定的结构，可以减少一些时间和空间的开销。

关系型数据库的优势：

1）复杂查询  
可以用SQL语句方便的在一个表以及多个表之间做非常复杂的数据查询。  
2）事务支持  
使得对于安全性能很高的数据访问要求得以实现。

**智力题**

1、登一个N级的台阶，每次可以上一级或两次，有多少种走法

当n=1，f(n)=1

当n=2，f(n)=2

当n=3，分第一步是踏2级还是1级，如果第一步踏1级，剩下2级，f(2)，如果第一步踏2级，剩下1级，f(1)。f(3) = f(2)+f(1)

……

以次类推，f(n) = f(n-1)+f(n-2)

2、怎样让一个产生1~5随机整数的函数扩大为产生1~9的随机整数

3、在一个数组中有两个元素只出现了一次，其他元素都出现了两次，如何找出那两个只出现了一次的元素？

答：假设这两个元素为a,b，那么将所有的元素异或之后，得到的结果是a^b，因为a!=b，所以结果一定不为0，结果化为2进制之后不为0的位，即a与b不相同的位，比如第k位。那么我们可以根据这个原则，将所有元素第k位是否相同将数组划分为两组，一组第k位为0，另一组为1，分别将这两组中的所有元素求异或，两组中分别得到的结果即为a,b。（求第k位是否为0，将该数与2^(k-1)相与即可）。