1. HashMap和HashTable的区别(前四条必答)：
   1. 两者父类不同。

HashMap继承自AbstractMap类，HashTable继承自Dictionary类。

不过他们都同时实现了map、Cloneable(可复制)、Serializable(可序列化)三个接口。

* 1. 对外提供的接口不同。

HashTable比HashMap多了elements()和contains()的方法。

elements()继承自HashTbale的父类Dictionary，用于返回此HashTable中value的枚举。

contains()判断该HashTable是否包含传入的value，作用与containsValue()一致。

* 1. 对null的支持不同

HashTable不允许key和value为空。

HashMap允许key和value都为null，但key只能有一个为null，为了保证key的唯一性；可以有多个key值对应的value为空。

* 1. 安全性不同

HashMap是线程不安全的，但是速度比HashTable快。

HashTable是线程安全的，每个方法上都加上synchronized关键字。

* 1. 两者的初始容量大小和每次扩容容量大小不同。
  2. 计算hash值的算法不同

1. 哪些集合类是线程安全的？

Vector(比ArrayList多了线程安全)

HashTable(比HashMap多了线程安全)

ConcurrentHashMap(一种高效但是线程安全的集合)

Stack(栈，线程安全，继承Vector)

1. 并行和并发的区别：

并发：指的是多个事情在同一时间间隔内同时发生，即轮流处理多个任务的能力。

并行：指的是多个事情在同一时刻同时发生，即同时处理多个任务的能力。

1. sleep()和wait()有什么区别？
   1. sleep方法是Thread类里面的静态方法,wait 是Object类中的方法
   2. sleep可以在非同步块中执行，wait只能在同步块中执行，否则就会报IllegalMonitorStateException
   3. sleep方法在进行线程阻塞时，不释放锁,wait方法在线程阻塞时，释放锁，在没有被其他线程唤醒时一直阻塞
2. 请讲一下分布式、(集群)、SOA、微服务之间的区别
   1. 分布式：不同模块部署在不同服务器上，用于解决网站高并发带来的问题
   2. 集群：多台服务器部署相同应用构成一个集群，通过负载均衡设备共同对外提供服务
   3. SOA：业务系统分解为多个组件，让每个组件都独立提供离散、自治、可复用的服务能力，通过服务组合和编排来实现上层的业务流程。用于简化维护、降低整体风险、伸缩灵活。
   4. 微服务：架构设计概念，各服务间隔离(分布式也是隔离)，自治(分布式依赖整体组合)其他特性(单一职责，边界，一部同行，独立部署)是分布式概念的跟严格执行SOA到微服务架构的演讲过程。

作用：个服务可以独立应用，组合服务也可以系统应用。

1. Session分布式方案
   1. 直接将数据存储在客户端口的cookie中
   2. Session复制(一种服务器集群session管理机制)
   3. 使用nginx的反向代理和负载均衡，基于nginx的ip-hash策略进行session绑定
   4. 基于redis存储session
2. 分布式锁的场景以及实现方案
   1. 场景：解决在分布式的、集群部署情况下，对象在多台机器的内存中保存时，保证服务器上的变量的一致性。
   2. 实现方案
      1. 数据库实现：
         1. 数据库建一张表，字段方法名并且作为唯一性，当一个方法执行时插入，则相当于获得锁，其他线程将无法访问，方法执行完则释放锁。
         2. 使用select \* from user u where username = '' for update 来对记录加上排他锁。操作完成后使用commit命令释放锁。
      2. Redis实现：
         1. 使用setnx 在key不存在时设置值，实现加锁，其他线程无法在创建该key的锁
         2. 设置成功的，就等于拿到了锁，执行业务
         3. 执行业务之后，使用del删除锁
         4. 为了防止执行业务过程中出现异常，第3步没有执行，第1步执行完毕时， 使用expire 设置过期时间，防止出现死锁等现象，让多个请求等待）
      3. Zookeeper实现：
         1. 首先zookeeper中我们可以创建一个持久化节点
         2. 然后再在该持久化节点下创建自己的临时顺序节点
         3. 获取所有的1中的持久化节点下的所有子节点，并排序
         4. 判读自己创建的节点是否最小值（第一位）
         5. 如果是，则获取得到锁，执行自己的业务逻辑，最后删除这个临时节点
         6. 如果不是最小值，则需要监听自己创建节点前一位节点的数据变化，并阻塞
         7. 如果不是最小值，则需要监听自己创建节点前一位节点的数据变化，并阻塞
3. 如何保证接口的幂等性(网址大概看看)

幂等性：保证多次调用对系统的影响是一致的，即多次调用之后结果是一致的。声明为幂等的服务会认为外部调用失败是常态，并且失败之后必然会有重试。

* 1. 使用token机制
  2. 使用数据库的乐观锁

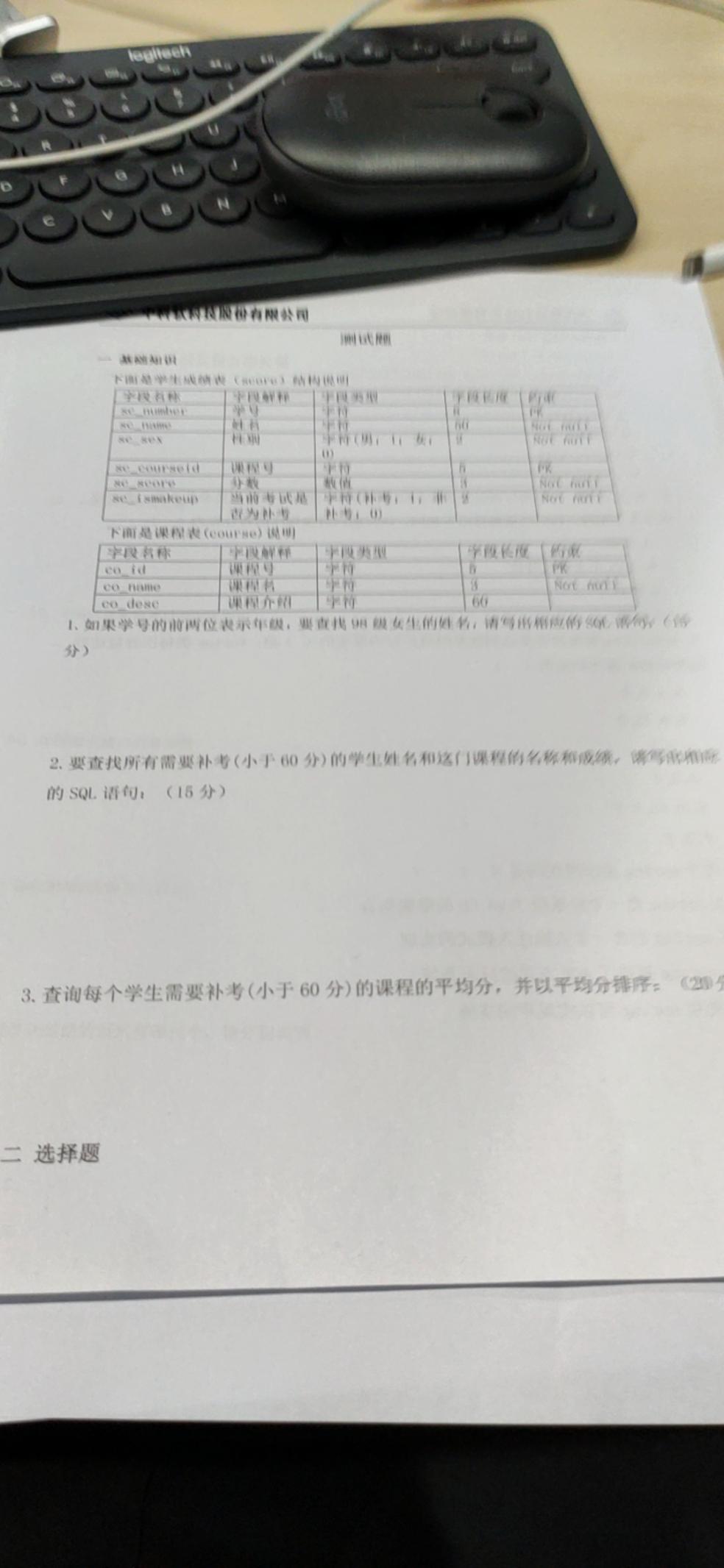
<https://segmentfault.com/a/1190000020172463>

1. Limit20000很慢怎么解决
   1. 前端加缓存，或者其他方式，减少落到库的查询操作，例如某些系统中数据在搜索引擎中有备份的，可以用 es 等进行搜索
   2. 使用延迟关联，即先通用 limit 得到需要数据的索引字段，然后再通过原表和索引字段关联获得需要数据(下面的语句是例子解释不用写上去)

select a.\* from a,(select id from table\_1 where is\_deleted='N' limit 100000,20) b where a.id = b.id

1. 表设计遵循的范式有哪几个，请分别说明。
   1. 1NF：列不可再分
   2. 2NF：基于1NF，行可唯一区分，依赖于主键
   3. 3NF：基于1NF、2NF，非主键属性不可依赖其他表的非主键属性，外键约束

测试题



1. select sc\_name from score where sc\_number like '98%' and sc\_sex = 0;

2. select s.sc\_name,c.co\_name from score s

join course c

on s.sc\_courseid = c.co\_id

where sc\_ismakeup = 1 and sc\_score < 60

3. select avg(sc\_score) from score s

where sc\_ismakeup = 1 and sc\_score < 60