1. 集合
2. HashMap初始大小是多少?怎么扩容的？

Jdk1.7默认是16,key的个数超过容量长度\*加载因子

的数量，会扩容为原来的2倍。Jdk1.8在扩容的时候采用

expectedSize / 0.75 + 1 公式，为了提高效率可以采用Maps.newHashMapWithExpectedSize().

1. Arrays.sort 是怎么实现排序的?

实现Comparator接口，或用lambda表达式,，实现倒序排序。

快速排序算法，先选择一个基准，从指针end往前移,遇到大于end值，进行交换，然后从指针start往后移动，遇到小于start值，进行交换，

递归完成排序。

public void sort(int[] a,int low,int high){

int start = low;

int end = high;

int key = a[low];

while(end>start){

//从后往前比较

while(end>start&&a[end]>=key)

//如果没有比关键值小的，比较下一个，直到有比关键值小的交换位置，然后又从前往后比较

end--;

if(a[end]<key){

int temp = a[end];

a[end] = a[start];

a[start] = temp;

}

//从前往后比较 while(end>start&&a[start]<=key)//如果没有比关键值大的，比较下一个，直到有比关键值大的交换位置

start++;

if(a[start]>key){

int temp = a[start];

a[start] = a[end];

a[end] = temp;

} //此时第一次循环比较结束，关键值的位置已经确定了。左边的值都比关键值小，右边的值都比关键值大，但是两边的顺序还有可能是不一样的，进行下面的递归调用

}

//递归

if(start>low) sort(a,low,start-1);//左边序列。第一个索引位置到关键值索引-1

if(end<high) sort(a,end+1,high);//右边序列。从关键值索引+1到最后一个

}

[制代码](javascript:void(0);)

1. HashMap实现原理？

数组+链表,HahMap 静态内部类Entry属性有key,value,next,hash 4个属性; 记录存储个数size,,阈值为初始容量\*负载因子，负载因子；

key通过hash函数确定hash值，hash值与容量 &运算确定位置，通过遍历链表查找value;对于自定义对象作为key,需要重写equals和hashcode方法

1. ConcurrentHashMap 比HashTable 性能优的地方?

HashTable对整个集合加锁，为了防止ConcurrentModificationException抛出，ConcurrentHashMap 使用volatile 关键字，直接修改内存的值，注销直接的缓存，禁止程序执行结果的重排序，写操作由于读操作，采取锁分段技术。

1. 为什么要重写hashcode方法？

equals继承Object,对于值对象，比较的是两个对象的值，对于引用对象比较的是两个对象的地址。如果你重写了equals 方法，而没有重新hashcode 方法，那么会出现明明两个对象是相同的，却因它的hash值不同，让这两个对象也不一样。

1. 多线程

线程池工作队列饱和，有哪些策略？

1. Abort策略，队列满了之后，新提交任务时，抛出异常。
2. CallerRuns策略，调节机制，既不抛弃任务也不抛出异常，在exectord的线程中去执行。
3. Discard策略，新提交的任务被抛弃。

4．DiscardOldest策略，抛弃对头的任务，提交新的任务。

三、Spring

动态代理实现的两种方式，以及他们的区别？  
SpirngAOP利用动态代理实现日志打印,事务管理，通过JDK和CGLIB两种方式实现，JDK需要被代理类实现一个接口，代理类实现同样的接口，CGLIB是代理类继承被代理类，作为代理对象，实现AOP的功能。

Spring如何实现事务管理?

Spring事务管理分为编程式事务管理和基于AOP技术实现的声明式事务管理，声明式事务管理只能作用在方法上，

切点精确到哪个类，切面精确到哪个方法,创建代理类,调用方法调用的是代理对象的方法,编程式事务管理能作用在代码块上。

Spring可以设置事务的传播行为，隔离级别，事务的只读，以减轻

事务对数据库的压力，遇到异常时，发生事务回滚。

1. linux命令

统计一个文件下，一个字符串出现的次数?

在命令 模式下输入 :%s/objStr//gn

1. redis

redis持久化的两种方式?

RDB和AOF,RDB 定时将操作日志写入硬盘，AOF是实时的将操作日志写入磁盘。

1. hibernate

hibernate的find和load方法的区别

find会先在一级缓存中查询，如果查不到再去数据库去查询，load可以取到一级缓存和二级缓存的数据。

1. 数据库

索引的实现原理

分为Clustered Index(主键索引) 和Secondary Index（辅助索引）,基于B+Tree的数据结构，Secondary Index 的索引树存的是Clustered Index的索引值，为了减小update,insert,delete操作的复杂度。

主键查询----直接在Clustered B+Tree上查询。

辅助索引查询---先在Secondary B+Tree上查询到主键，然后在ClusteredB + Tree上查询。

Insert---在Clustered B+Tree上插入数据，然后在SecondaryB +Tree上插入主键。

DELETE---在Clustered B+Tree上删除数据，在所有SecondaryB+Tree上删除主键。

UPDATE非主键列---在ClusteredB+Tree更新数据。

UPDATE主键列---在ClusteredB+Tree假删除原来的记录，插入新的记录,Secondary B+Tree删除原来的数据，插入新的数据。

UPDATE辅助索引的键值---在ClusteredB+Tree上更新数据，在每一个SecondaryB+Tree删除原来的原来的主键，新增原来的主键。

1. JVM调优

8种基本数据类型，byte取值范围-128-127，char的取值范围为0 ~ 216-1

堆中存什么?栈中存什么?

栈中存的信息都是跟当前线程相关信息，比如基本数据类型和堆中对象的引用。堆中负责存储对象信息。

为什么不把基本类型放堆中？

基本类型占用内容大小固定，不会出现动态增长的情况。

Java的参数传递是传值?还是传引用?

对象是传引用，基本数据类型是值传递。

Java对象大小计算?

栈中引用占用4个字节，Object空对象占用8个字节，一个空对象至少12个字节，一般java保存的时候都是8的整数倍。

垃圾回收算法?

a1.引用计数 增加一个引用，增加一个计数，删除一个引用，减少一个计数，最致命的无法处理循环引用的问题。

a2.标记-清除 从根节点开始标记所有被引用的对象，遍历整个堆，把未标记的对象清除。

a3.复制 垃圾回收遍历正在使用区域，把使用的对象复制到另一个区域，在这个区域进行处理。

a4.标记-整理 从根节点开始标记所有被引用的对象，清除未标记的对象，并且将存活对象压缩到堆中一块。

b1.增量收集 应用进行的同时进行垃圾回收。

b2.分代收集 对象不同生命周期的垃圾回收算法，把对象分为年轻代，年老代，持久代。

c1.串行收集 单线程处理所有垃圾回收

c2.并行收集 多线程处理垃圾回收，效率高。

c3.并发收集 GC线程和应用线程大部分时间是并发执行。

分代收集：堆内存分为年轻代，年老代，持久代，年轻代分为一个Eden区，两个Survivor区，如果它们被放满，则会被放到年老代，持久代用于存放一些静态文件，通过设置-XX:MaxPermSize进行设置。并行收集器-XX:+UseParallelOldGC，串行收集器-XX:+UseSerialGC打开。并发收集器-XX:+UseConcMarkSweepGC打开，如果收集期间，堆内存并沾满，会出现Concurrent Mode Failure情况会暂停，进行垃圾回收。

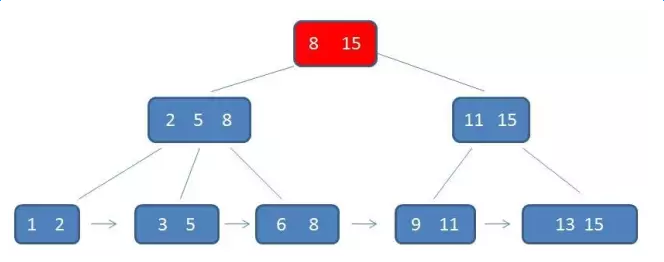
索引失效的情况？

1. 类型的隐式转换
2. 索引字段允许为空
3. 使用in,or,exists
4. like 索引字段前面加 %
5. 索引字段使用函数
6. 算法

折半查找

public static int searchByHalf(int key,int[] array){  
 int pos = -1;  
 int start = 0,end = array.length - 1;  
 int mid = (start + end) / 2;  
 while(start <= end){  
 if(array[mid] == key){  
 pos = mid;  
 break;  
 }else if(array[mid] > key){  
 end = mid - 1;  
 }else {  
 start = mid + 1;  
 }  
 mid = (start + end) / 2;  
  
 }  
 return pos;  
}

B+Tree



1. 单节点可以存储更多的元素，使得查询磁盘IO次数更少。
2. 所有查询都走到叶子节点，查询性能稳定。
3. 所有叶子节点形成有序列表，便于范围查询。
4. jdk

java 8 新特性