面经 | 记 21 年 3 月一次 java 面试经历

作者：z875479694h
原文链接：https://ld246.com/article/1625681336953
来源网站：链滴

1.笔试题

Integer intger = new Integer(100)和String str = new String（”ABC”）分别会创建多少个对象。

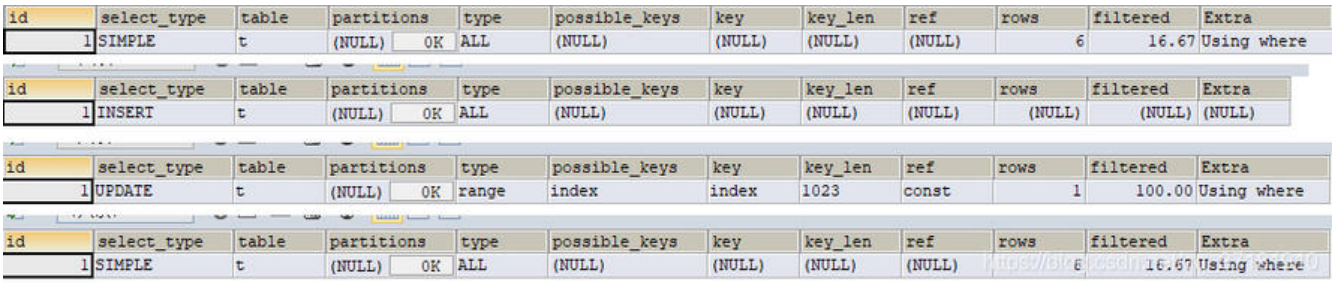
解析

```
// 一个
Integer intger = new Intger(100)
// 是2个，"abc"创建在常量池，new String()创建在堆
String str = new String("abc");
// 1个，"abc"创建在常量池
String str = "abc";
// 1个，"abc"
String str1 ="abc"; String str2 = "abc";
// 3个，"ab" , "c" ,"abc"
String str = "ab" + "c";
```

数据表t有三个字段title，name，age建立复合索引tab_index(“name”,”age”)，下面哪些语句能用上索引？（）

A. select * from t where age=18
B. insert into t values('title','name',18)
C. update t set title = 'a' where name = 'A'
D. select * from t where name like "%t%" and age=18

解析



id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	t	(NULL)	ALL	(NULL)	(NULL)	(NULL)	(NULL)	6	16.67	Using where

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	INSERT	t	(NULL)	ALL	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	UPDATE	t	(NULL)	range	index	index	1023	const	1	100.00	Using where

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	t	(NULL)	ALL	(NULL)	(NULL)	(NULL)	(NULL)	6	16.67	Using where

（wps简述题）为什么TCP4次挥手时客户端需要等待2MSL?

解析

MSL（Maximum Segment Lifetime），TCP允许不同的实现可以设置不同的MSL值。

第一，保证客户端发送的最后一个ACK报文能够到达服务器，因为这个ACK报文可能丢失，站在服务的角度看来，我已经发送了FIN+ACK报文请求断开了，客户端还没有给我回应，应该是我发送的请断开报文它没有收到，于是服务器又会重新发送一次，而客户端就能在这个2MSL时间段内收到这个传的报文，接着给出回应报文，并且会重启2MSL计时器。

第二，防止类似与“三次握手”中提到了的“已经失效的连接请求报文段”出现在本连接中。客户端送完最后一个确认报文后，在这个2MSL时间中，就可以使本连接持续的时间内所产生的所有报文段从网络中消失。这样新的连接中不会出现旧连接的请求报文。

[来源]CSDN

（wps简述题）线程与进程有多少种状态以及他们之间的的区别?

解析

进程：

创建状态(new)、就绪状态(ready) 、运行状态(running)、阻塞状态(waiting) 、结束状态(terminated)

线程：

初始(new)、运行(runnable)、阻塞(blocked)、等待(waiting)、超时等待(timed_waiting)、终止(terminated)

两者区别：

做个简单的比喻，进程=火车，线程=车厢。线程在进程下行进（单纯的车厢无法运行），一个进程以包含多个线程（一辆火车可以有多个车厢），不同进程间数据很难共享（一辆火车上的乘客很难换另外一辆火车，比如站点换乘），同一进程下不同线程间数据很易共享（A车厢换到B车厢很容易）。

进程要比线程消耗更多的计算机资源（采用多列火车相比多个车厢更耗资源），进程间不会相互影响一个线程挂掉将导致整个进程挂掉（一列火车不会影响到另外一列火车，但是如果一列火车上中间的节车厢着火了，将影响到所有车厢），进程可以拓展到多机，进程最多适合多核（不同火车可以开在

个轨道上，同一火车的车厢不能在行进的不同的轨道上）。

进程使用的内存地址可以上锁，即一个线程使用某些共享内存时，其他线程必须等它结束，才能使用一块内存。（比如火车上的洗手间） - "互斥锁"，进程使用的内存地址可以限定使用量（比如火车上餐厅，最多只允许多少人进入，如果满了需要在门口等，等有人出来了才能进去） - “信号量”。

[来源]知乎

逻辑题:有100个人，从1排到100分别在背后随机贴红黑两种颜色的卡片，然后100号能看到前面的99个人的颜色卡，以此类推，2号能看1号的颜色，每个人都能看见自己前面人的颜色。从后往前开始回到色，猜出自己的卡片颜色正确的一分，出一个策略，写出你的策略能到的保底分数以及写出思路

解析

3人一组，100号看99和98号的颜色，如果98号跟99号一样就红色，不一样就黑色，99号根据看到的色与100的答案说出自己的颜色，98号根据100号和99号的答案推断出自己的颜色以此类推，多出来一个盲猜颜色，总共33组。每组能确保2个人颜色一定正确，33*2=保底66分。

算法题:珠宝大盗偷珠宝，有一排珠宝展柜，每个格子的珠宝的价值同，连续两个格子变空就报警，请问珠宝大盗怎么偷到最大价值的珠。

解析

```
// 动态规划
  public static int getMaxValue(int[] arr){
        int[] data = new int[arr.length];
        data[0] = arr[0];
        data[1] = Math.max(arr[0], arr[1]);
        for (int i = 2; i < arr.length; i++) {
            int val = arr[i];
            data[i] = Math.max(data[i - 1], data[i - 2] + val);
        }
        return data[data.length - 1];
    }
  // 非动态规划
  public static int getMaxValue2(int[] arr) {
        int sum1 = 0, sum2 = 0;
        for(int i = 0; i < arr.length; i++){
            if(i % 2==0){
                sum1 = Math.max(sum2, sum1 + arr[i]);
            }else{
                sum2 = Math.max(sum1, sum2 + arr[i]);
            }
        }
```

```
    return Math.max(sum1,sum2);
}
```

2.面试题

Redis的数据类型有哪几种

解析

9种，5种基础类型分别是String，Hash，List，Set，Sorted Set(ZSet)。后面新增的3种高级数据类分别是BitMap（位图只有0和1两种情况），HyperLogLog（海量数据的计算，适用于统计数据），GO（记录经纬度及计算距离）。Redis5.0以后新增了Stream（消息队列）。

你说一下你用Zset做排行榜是为什么

我的回答

利用了有序集合的有序性，因为有序集合以score大小的不同为集合中的成员进行从小到大的排序。需要将评论数作为score，获取集合的倒序指定的个数。完成排行。

Java的集合有什么实现类

解析

Collection接口：

List：

ArrayList（动态数组实现）、LinkedList（双向链表实现）、Vector(线程安全但是古老没人用了)、Stck（线程安全继承的Vector）、CopyOnWriteArrayList（线程安全，只能保证最终结果的一致性，内存）

Queue：

PriorityQueue（通过数组来维护一个堆结构）、ArrayDeque（双端队列，循环数组实现）

Set：

HashSet（HashMap实现，无序，HashMap的封装）、LinkedHashSet（HashMap实现，有序，证进出顺序）、TreeSet（红黑树，有序，自然排序）

非Collection接口：

Map:

HashMap（无序）、LinkedHashMap（有序，按进入的顺序排序）、TreeMap（红黑树，有序，然排序）、Hashtable（线程安全，直接锁整张表性能差）、ConcurrentHashMap（线程安全，锁点。性能高）

（小编吐槽面试官搁这背API呢）

你说一下HashMap的扩容机制

解析

空参数的构造函数：实例化的HashMap默认内部数组是null，即没有实例化。第一次调用put方法时则会开始第一次初始化扩容，长度为16。

有参构造函数：用于指定容量。会根据指定的正整数找到不小于指定容量的2的幂数，将这个数设置值给阈值（threshold）。第一次调用put方法时，会将阈值赋值给容量，然后让 。（因此并不是我手动指定了容量就一定不会触发扩容，超过阈值后一样会扩容！！）

如果不是第一次扩容，则容量变为原来的2倍，阈值也变为原来的2倍。（容量和阈值都变为原来的2时，负载因子还是不变）

此外还有几个细节需要注意：首次put时，先会触发扩容（算是初始化），然后存入数据，然后判断否需要扩容；不是首次put，则不再初始化，直接存入数据，然后判断是否需要扩容；

[来源]知乎

说一下你对CopyOnWriteArrayList的线程安全性的理解

解析

CopyOnWriteArrayList 类的所有可变操作（add，set等等）都是通过创建底层数组的新副本来实现。当 List 要被修改时，将对原有数据进行一次拷贝并把新数据写入到副本中。写完之后，再将修改完副本替换成原来的数据，所以CopyOnWriteArrayList不保证即时数据一致性，只能保证最终结果的据一致性。且因为每次改变都会复制一份新副本因此需要大量内存且性能较差，合适读多写少且不追强一致性的场景。

ArrayList的扩容机制，后面追问扩容的时候数据怎么复制到新的数组

解析

扩容机制参考这篇：https://snailclimb.gitee.io/javaguide/#/docs/java/collection/ArrayList%E6BA%90%E7%A0%81+%E6%89%A9%E5%AE%B9%E6%9C%BA%E5%88%B6%E5%88%86%E6%

E%90

复制新数组的方法是调用的是：System.arraycopy(elementData,index, elementData, index + 1,size - index);

而 System.arraycopy调用的是C本地库（具体自己去了解，不深入了，超纲了）

请说一下数据库的1NF、2NF、3NF和BNCF

解析

1NF:列不可再分

2NF：非主键列和主键列之间，是完全依赖于主键，还是依赖于主键的一部分（只依赖某个主键）

3NF：非主键列之间，不存在依赖，只直接依赖主键。非主键属性之间无依赖关系

BCNF：主键列之间，不存在依赖。主键属性之间无依赖关系

MySQL的四种事务隔离级别

解析

1.原子性（Atomicity）：事务是最小的执行单位，不允许分割。事务的原子性确保动作要么全部完，要么完全不起作用；

2.一致性（Consistency）：执行事务前后，数据保持一致，例如转账业务中，无论事务是否成功，账者和收款人的总额应该是不变的；

3.隔离性（Isolation）：并发访问数据库时，一个用户的事务不被其他事务所干扰，各并发事务之间据库是独立的；

4.持久性（Durabilily）：一个事务被提交之后。它对数据库中数据的改变是持久的，即使数据库发故障也不应该对其有任何影响。

主键索引跟主键数据是存放在一起的吗?

解析

InnoDB 引擎中，其数据文件本身就是索引文件。树的叶节点 data 域保存了完整的数据记录。

MyISAM 引擎中，索引文件和数据文件是分离的，B+Tree 叶节点的 data 域存放的是数据记录的地

。

数据和索引分别存在B+树的什么节点?

解析

叶子节点的key域是数据表的主键而data 域保存了完整的数据记录。

非叶子节点只存储索引并不存储行数据，只为了能存储更多索引键，从而降低B+树的高度，进而减少O次数。

如果没有设置主键索引，mysql用什么做主键

解析

如果定义了主键，那么InnoDB会使用主键作为聚簇索引

如果没有定义主键，那么会使用第一非空的唯一索引（NOT NULL and UNIQUE INDEX）作为聚簇引。

如果既没有主键也找不到合适的非空索引，那么InnoDB会自动生成一个不可见的名为ROW_ID的列为GEN_CLUST_INDEX的聚簇索引，该列是一个6字节的自增数值，随着插入而自增。

缺少主键的表，InnoDB会内置一列用于聚簇索引来组织数据。而没有建立主键的话就没法通过主键进行索引，查询的时候都是全表扫描，小数据量没问题，大数据量就会出现性能问题。

TCP与UDP有什么不同?

解析

UDP 在传送数据之前不需要先建立连接，远地主机在收到 UDP 报文后，不需要给出任何确认。虽然DP 不提供可靠交付，但在某些情况下 UDP 确是一种最有效的工作方式（一般用于即时通信）

TCP 提供面向连接的服务。在传送数据之前必须先建立连接，数据传送结束后要释放连接。TCP 不提广播或多播服务。由于 TCP 要提供可靠的，面向连接的传输服务（TCP的可靠体现在TCP在传递数据前，会有三次握手来建立连接，而且在数据传递时，有确认、窗口、重传、拥塞控制机制，在数据传后，还会断开连接用来节约系统资源），这一难以避免增加了许多开销，如确认，流量控制，计时器及连接管理等。这不仅使协议数据单元的首部增大很多，还要占用许多处理机资源。

TCP的握手与挥手的示意图

解析

自己搜吧懒得贴了

说一下你对Spring的IOC/DI的理解

解析

IOC（Inverse of Control:控制反转）是一种设计思想，就是 将原本在程序中手动创建对象的控制权交由Spring框架来管理。

而DI（依赖注入）组件之间依赖关系由容器在运行期决定，即由容器动态的将某个依赖关系注入到组之中。

DI的概念诞生比IOC晚，是2004年大神Martin Fowler提出的。IOC与DI它们是同一个概念的不同角描述。

为什么要用IOC

解析

将对象之间的相互依赖关系交给 IoC 容器来管理，并由 IoC 容器完成对象的注入。这样可以很大程上简化应用的开发，把应用从复杂的依赖关系中解放出来。

IoC 容器就像是一个工厂一样，当我们需要创建一个对象的时候，只需要配置好配置文件/注解即可，全不用考虑对象是如何被创建出来的。

在实际项目中一个 Service 类可能有几百甚至上千个类作为它的底层，假如我们需要实例化这个 Servie，你可能要每次都要搞清这个 Service 所有底层类的构造函数，这可能会把人逼疯。

如果利用 IoC 的话，你只需要配置好，然后在需要的地方引用就行了，这大大增加了项目的可维护性降低了开发难度。

Elasticsearch 搜索为什么那么快

解析

ES 是基于 Lucene 的全文检索引擎，它会对数据进行分词后保存索引，擅长管理大量的索引数据，对于 MySQL 来说不擅长经常更新数据及关联查询。

正常来说我们一般使用id查询到具体对象的方式称为使用正排索引，其实也能理解为一种散列表。本是通过 key 来查找 value。比如通过 id=4 便能很快查询到 name=张红尘，age=20 这条数据。

ES 中采用的是一种名叫倒排索引的数据结构。通过分词器将所有非关键词标识的内容进行分词，并分词作为key，而li（类似于id）作为value反向存储，每当搜索关键词时只需要取得改关键词的li即可得所有数据。
