## **常见问题汇总**

1. **Java**
2. **Java SE / Java EE**
3. **Java基础知识**
4. final 关键字的作用。
5. java 中==和 equals 和 hashCode 的区别。
6. int、char、long 各占多少字节数。
7. int 与 integer 的区别。
8. String、StringBuffer、StringBuilder 区别。
9. 什么是内部类？内部类的作用。
10. 泛型中 extends 和 super 的区别？
11. final，finally，finalize 的区别。
12. 序列化 Serializable 和 Parcelable 的区别。
13. String 转换成 Integer 的方式及原理。
14. **面向对象编程**
15. 抽象类和普通类的区别，接口和抽象类的区别。
16. 父类的静态方法能否被子类重写？静态属性和静态方法是否可以被继承？
17. **AOP**
18. **多线程**

线程池有哪些类型？

concurrentHashMap分段锁原理，java8和java7实现的区别。

Java锁有了解吗？Synchronized和ReentrantLock区别？说说如何ReentrantLock如何实现超时锁的等待？

进程和线程的区别。

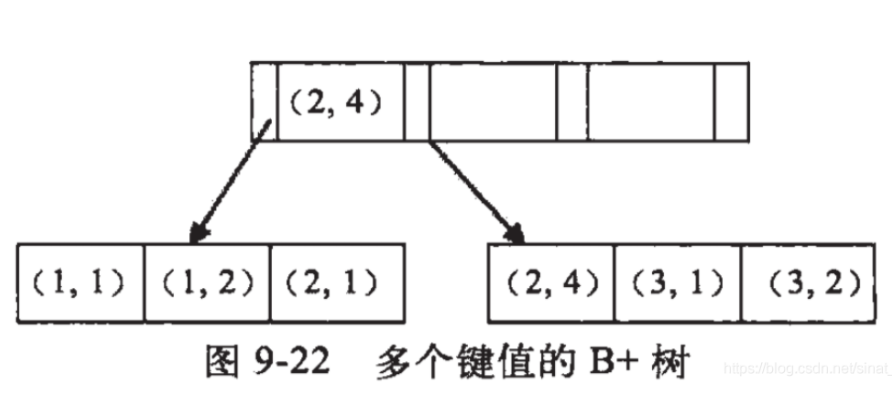
1. **IO**
2. **Java Web**
3. **Web编程原理**
4. **Servlet**
5. **MVC模型**
6. **Web**
7. **Web 基础**
8. **脚本语言**
9. **HTML**
10. **JavaScript**
11. **jQuery**
12. **主流框架**
13. **Vue**
14. **Angular**
15. **数据库**
16. **关系型数据库**
17. **MySQL**
18. MySQL索引的原理是什么。

B+树，本质上它是将瘦长的平衡二叉树，改造成一个矮扁的平衡多叉树从而减少IO查询次数，平均查询复杂度O(logN),另外与B树不同的是，除了叶子结点外，其他结点均不存储数据，另外叶子结点之间，用[链表](https://www.nowcoder.com/jump/super-jump/word?word=%E9%93%BE%E8%A1%A8" \t "https://www.nowcoder.com/discuss/_blank)方式相连以加速查询。当然视面试官的反应，可以再补充B+树的插入和删除操作。【牛客】

1. MySQL建立索引时，应该注意什么？

①对区分度高的列作索引，对于那种只有两三个值的字段做索引没有太多意义。②建立联合索引时要满足最左匹配原理。只对需要作为查询条件的列做索引，过度索引会拖慢插入速度。【牛客】

最左匹配原理：是针对联合索引的。索引的底层是一个B+数，联合索引自然也是一个B+数，只不过联合索引的键值数量是多个。由于构建一颗B+数只能根据一个值来构建，因此数据库依据联合索引的最左字段来构建B+树。假如我们创建一个联合索引(a,b)，如下图：



从上图我们可以发现，a的值是有序的1,1,2,2,3,3，而b的值是无序的，所以当查询条件为a = 1时，索引是生效的，而查询条件仅为b = 2时索引是无效的，因为a的有序性和b的无序性决定了这一特点。但当a的值为定值时，我们发现，b的值是有序的，如上图，当a = 1时，b是有序的1,2；当a = 2时，b为1,4。因此当查询条件为a = 1 and b = 2时，索引是生效的，由于feature优化器的存在，当查询条件为b = 2 and a = 1时，索引同样生效，因为优化器会自动优化条件顺序。但是例如查询条件为a > 1 and b = 2时，a的字段可以匹配上索引，但是b不可以，因为当a是一个范围是，b是无序的。

假设建立联合索引(a,b,c):

1、全值匹配查询时

①select \* from table\_name where a = ‘1’ and b = ‘2’ and c = ‘3’

②select \* from table\_name where b = ‘2’ and a = ‘1’ and c = ‘3’

③select \* from table\_name where c = ‘3’ and a =‘1’ and b = ‘2’

因为优化器的存在，索引均生效

2、匹配左边的列时

①select \* from table\_name where a = ‘1’

②select \* from table\_name where a = ‘1’ and b = ‘2’

③select \* from table\_name where c = a =‘1’ and b = ‘2’ and ‘3’

符合最左原理，索引均生效

④select \* from table\_name where a = ‘1’ and c = ‘3’

不连续，a的索引生效，c的索引不生效。

3、 匹配列前缀

①select \* from table\_name where a like ‘As%’ //前缀都是排序好的，走索引

②select \* from table\_name where a like ‘%As’ //全表查询

③select \* from table\_name where a like ‘As%’ //全表查询

4、匹配范围值

①select \* from table\_name where a > 1 and a < 3 //索引生效

②select \* from table\_name where a > 1 and a < 3 and b > 3 //a索引生效，b索引无效。

③select \* from table\_name where a =1 and b > 3 //a,b索引均生效。

5、排序

①select \* from table\_name order by a,b,c //索引生效

②select \* from table\_name order by b,a,c //调到顺序，违反最左匹配原理，索引不生效。

③select \* from table\_name where a = 1 order by b,c //索引生效

1. Mysql用过对吧？说说Mysql索引建立策略？假如我给你三个字段, 性别, 年龄和身份证号, 哪个字段更适合做索引？
2. **Oracle**
3. **非关系型数据库**
4. **Redis**
5. Redis为什么这么快？
6. **Neo4j**
7. **数据库优化**
8. MySQL的优化。
9. **数据结构与算法**
10. **数据结构基础**
11. B+树相比于哈希索引有什么优点？
12. hashMap原理,put和resize过程。
13. B-树和B+树区别，数据库索引原理，组合索引怎么使用？阐述最左匹配的原理。
14. hashMap原理。
15. **数据操作**
16. **常见算法实现**
17. **框架**
18. **Spring**
19. Spring生命周期，几种scope区别，aop实现有哪几种实现，接口代理和类代理会有什么区别？
20. BeanFactory 和 ApplicationContext 有什么区别？
21. Spring Bean 的生命周期。
22. Spring IOC 如何实现。
23. Spring AOP 实现原理。
24. Spring MVC 运行流程。
25. Spring 事务底层原理。
26. Spring 的单例实现原理。
27. **Spring MVC + MyBatis / Hibernate**
28. Mybatis 的 ${} #{}的区别是什么？
29. Spring MVC中，@RequestMapping的实现原理？
30. Spring MVC 启动流程。
31. **Spring Boot**
32. springboot的启动原理（说了main函数，类加载器，主类的注解等）。
33. **Spring Cloud**
34. **微服务**
35. **设计模式**
36. **常见设计模式**
37. **常见设计模式代码实现**
38. **JVM**
39. **JVM原理**
40. G1 和 CMS 的区别。
41. 类加载过程（类的生命周期），解释各个阶段的意义。
42. 你们垃圾回收器用的什么？ G1有哪些特点？G1如何实现可预测的停顿时间？漏标问题如何解决的？介绍下三色标记？说说STAB[算法](https://www.nowcoder.com/jump/super-jump/word?word=%E7%AE%97%E6%B3%95" \t "https://www.nowcoder.com/discuss/_blank)？
43. 哪些情况下的对象会被垃圾回收机制处理掉？
44. 对象不可达，一定会被垃圾收集器回收么？
45. Java中实现多态的机制是什么？
46. **JVM优化**
47. **计算机网络**
48. **通讯原理及主流通讯协议**
49. **通讯代码实现**
50. **Socket**
51. **WebSocket**
52. **MQ**
53. **操作系统**
54. **Windows**
55. **Linux**
56. **项目经验**