 最近研究了Hibernate中的一些问题，发现除了缓存机制，还有些问题也值得我们深思，在hibernate严格限定Java包装类和工具类与相应数据库底层数据类型的映射的时候，各位是否想过，为什么要这么映射，也许你会说这个是hibernate3.0的规范而已，但是当初为什么指定这样的规范呢，也许各位没有去深究，暂时我们抛开这个问题不谈，这里只说明一件事，那就是要积极的思考为什么？ 学习完了各种数据库，学习了各种SQL查询语句后，你对数据库了解多少，其底层是怎么实现的？昨天和疯哥就一个IO问题讨论许久，回家后突然对数据库的实现原理茅塞顿开，下面我就把我这两天的一点总结和大家一起分享分享，希望大家不要见笑，共同交流，共同提高： 1.首先我们要明白数据库到底是怎么存储的： 在Oracle中，我们的数据存储在表中，我们的表存储在表空间中，而表空间直观的看就是几个数据.dbf文件，在惊讶Oracle和众多数据库的查询速度之快的时候，我们自己读取文件的时侯就算运用了貌似很牛逼的各种缓存机制来实现快速的从文件中得到文件中的数据，如果遍历数据文件，那么绝对会出现全数据文件遍历的问题，那么我们联系实际来寻求一种更加好的方式。 2.生活的实例： 在我们的生活中，长沙这么大的城市我们想找一间房子或者酒吧，那么就必须得到地址，否则就会漫无目的的到处乱撞，更合理的是遍历整个城市的所有地点，我的天，这样的效率该何其低下。得到地址和数据文件联系起来的关系

貌似不大，但是其中必有牵连，要不然牛叉的数据库哪里来的有理论支撑？ 3.解决方式： 既然我们可以与地址挂钩，你可能已经想到了将文件逐条读出，然后放入内存，然后根据地址去找，这样做的话固然可以，但是对于突然地断电，或者内存容量不够的突发状况的时候，就应该寻求更加稳妥和更加节约昂贵资源的方法，即：在文件中按照地址来进行查询和修改 4：关于文件偏移量： 文件不比内存，有相应的地址，但是我们可以采取另外一种类似于内存地址的方式，那就是文件偏移量: 在百度上面有人做了这样的解答：文件偏移：文件开始的第一个字节偏移量为0之后每经过一个字节偏移量就加1。对，我们就是要将文件按照一定的格式和顺序存储，我们以后去读取的时候只需要在相应的“ 地址簿 ”中获取文件偏移量，然后直接去操作IO进行读取或者修改，就达到了快速的对数据文件进行读取和修改的问题。 那么我们用Java的IO包 中的RandomAccessFile类来模拟一下这样的场景： 首先介绍一下这个类，这个类称之为随机读取文件类，或者是随机访问文件类，适应随机读取文件，但是一般将它用于处理定长数据，何谓定长数据呢？则是每条记录按照相应的格式并且拥有相同的字节长度，这样处理起来才能更加的方便，IO才知道到底从哪里开始读取数据和获取数据，同样的在网络上的断点续传，多线程下载就是这样的理论的经典实现，但是现在我们不拘泥与理论，上代码： public class Employee { private String name . private int age . public static final int len = 8 .//String 型的定长，相当于数据库中的类型长度 public Employee(String name , int age ){ if(name.getBytes().length lt. len){ name = "\u0000" .//空白 ， 占8b } this.name = name. }

this.age = age . } public String getName() { return name. } public int getAge() { return age. } } 我现在创建几个对象进行存储： public class TestRandomAccessFile { public static void main(String[] args) throws Exception { Employee e1 = new Employee("张三" , 23). Employee e2 = new Employee("zhangsan" , 35). Employee e3 = new Employee("li张三" , 67). RandomAccessFile rf = new RandomAccessFile(new File("C:/employee.txt") , "rw"). rf.write(e1.getName().getBytes()). rf.writeInt(e1.getAge()). rf.write(e2.getName().getBytes()). rf.writeInt(e2.getAge()). rf.write(e3.getName().getBytes()). rf.writeInt(e3.getAge()). rf.close(). RandomAccessFile raf = new RandomAccessFile(new File("C:/employee.txt") , "r"). raf.skipBytes(12). byte[] buf = new byte[8]. int i = raf.read(buf). System.out.println("i= " i). System.out.println(new String(buf).trim() raf.readInt()). raf.seek(24). raf.read(buf). System.out.println("i= " i). System.out.println(new String(buf).trim() raf.readInt()). } } 结果如下： i= 8 zhangsan35 i= 8 li张三67 在对数据进行指定位置的读取的时候，其他的修改和快速检索的实现和类似于数据库创建索引的实现的问题其实就是将所在的文件读取偏移量存储与一个B-tree中进行快捷所有的功能。如果你想做的更加完善，那么请参考MS-SQL中的经典理论著作，数据库系统原理。到此我相信大家对我文章开始的那个问题，即为什么严格限定类型映射和数据库字段定长的问题，有了一个比较清晰的答案。这里我就不废话了，希望各位学的愉快。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com