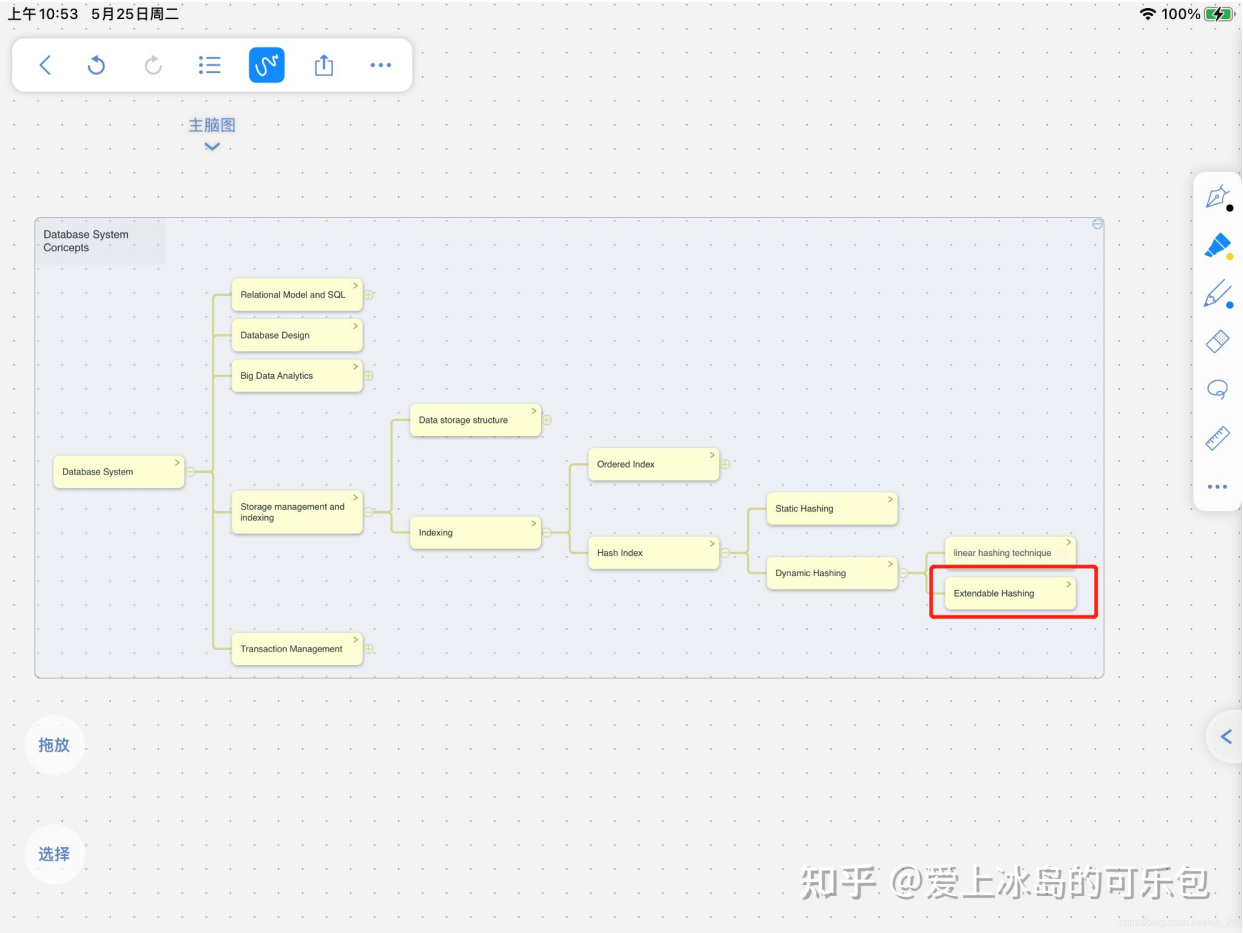
数据库——可拓展哈希（Extendable Hashing）

干什么用的

首先明确这是一种【存数据】的方法。比如有100个文件，有方法的找肯定比一个一个找要快。聪明的前辈们想出很多方法，有二分法，B-Tree，Hash等等。这些方法也被叫做“索引”（Index）。下图是可拓展哈希在数据库知识模块里的位置。（图是我期末考试自己总结的，不认同请友善指出。）



怎么用

从一个栗子入手。作为学校IT部门的打工人，领导要求我把7个部门的信息存起来，也就是7条记录（record）。

dept_name	h(*dept_name*)
Biology	0010 1101 1111 1011 0010 1100 0011 0000
Comp. Sci.	1111 0001 0010 0100 1001 0011 0110 1101
Elec. Eng.	0100 0011 1010 1100 1100 0110 1101 1111
Finance	1010 0011 1010 0000 1100 0110 1001 1111
History	1100 0111 1110 1101 1011 1111 0011 1010
Music	0011 0101 1010 0110 1100 1001 1110 1011
Physics	1001 1000 0011 1111 1001 1100 0000 0001

Figure 24.9 Hash function for *dept_name*.

第一步：得到哈希值（hash value）。

如右栏所示。（什么是哈希不是这篇的重点，跳过）。通过哈希函数，我们会得到一个二进制数。之后我们会重点对这个二进制数进行操作。

dept_name	h(*dept_name*)
Biology	0010 1101 1111 1011 0010 1100 0011 0000
Comp. Sci.	1111 0001 0010 0100 1001 0011 0110 1101
Elec. Eng.	0100 0011 1010 1100 1100 0110 1101 1111
Finance	1010 0011 1010 0000 1100 0110 1001 1111
History	1100 0111 1110 1101 1011 1111 0011 1010
Music	0011 0101 1010 0110 1100 1001 1110 1011
Physics	1001 1000 0011 1111 1001 1100 0000 0001

Figure 24.9 Hash function for *dept_name*.

知乎 @爱上冰岛的可乐包
https://blog.csdn.net/feb_24

第二步：先对可拓展哈希有个大概的概念。

主要明确三个概念：全局深度（global depth），局部深度（local depth）以及桶（bucket）。全局深度和局部深度我们之后会提。桶就是存记录（records）的地方，我可以往桶里放一条，两条，三条。但要规定

好最多能装几条。这个例子里我们规定最多能装2条。在图里你能看到bucket 1只有两行。下图这个就是可拓展哈希的初始状态。

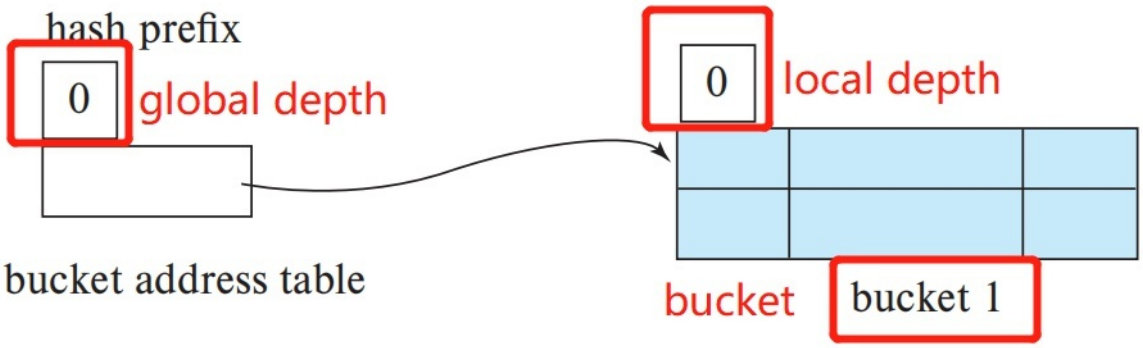


Figure 24.10 Initial extendable hash structure.

第三步：插入数据。

插入Comp.Sci和Finance这两条。现在还不涉及全局深度和局部深度的改变。两条新记录（record）就直接往里放就行。书上没有这个步骤的图，大家看下我临时编辑的图。

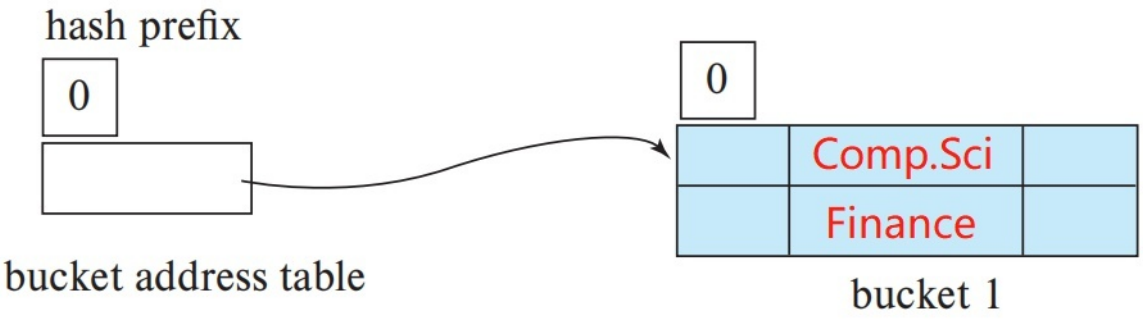


Figure 24.10 Initial extendable hash structure.

插入Music。这时一个桶的条数大于它的上限（2条），所以要开始分裂。如何分裂是根据哈希值（hash value）决定的。取哈希值第一位。哈希值是二进制数，第一位只有两种可能，0和1。Comp.Sci是1，Finance是1，Music是0。很简单的把是1的放在一个桶（bucket）里，是0的放在另一个桶（bucket）里。由于我们用了哈希值的第一位，所以全局深度（global depth）变成1。这里做一下全局深度和局部深度的

区分。两个都表示用了几位哈希值。全局深度是所有记录要用到几位哈希值，局部深度是该桶里的记录用到几位哈希值。产生这种区别的原因是，有的哈希值用不到全局深度那么多位。我们会在后面的例子看到，现在这么说还比较抽象。

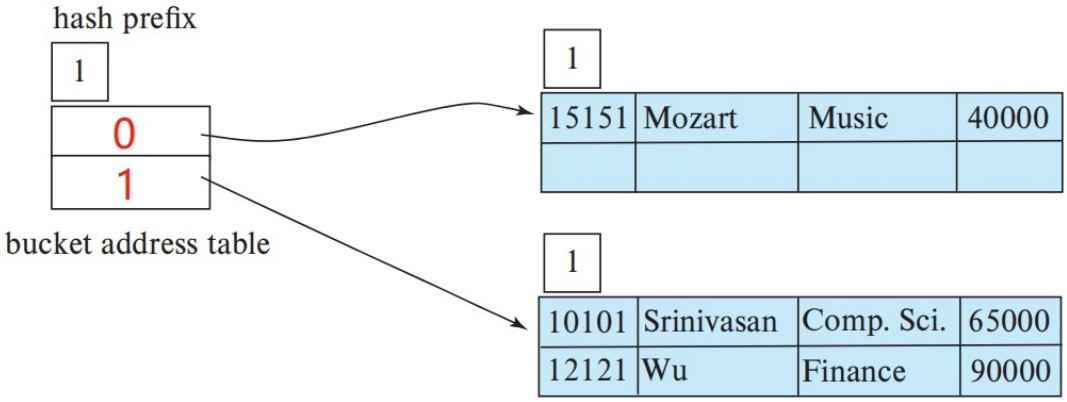


Figure 24.11 Hash structure after three insertions.

插入Physics。Physics的插入会让第二个桶产生溢出，所以要再次分裂。一位哈希值就不够用了，需取哈希值前两位，相应的全局深度变成2。第二个桶分裂成桶二和桶三。这时你会发现前两个指针（00和01）都指向了第一个桶。书上没具体说明原因。我给出的解释是：这样做节省内存。当一条新纪录（record）且哈希值开头为（0x，x可为0或者1），都可以被存入第一个桶而不导致溢出。所以桶一没有分裂的必要。只要我设置局部深度（local depth）为1，也就值只读哈希值第一位，就可以兼容所有第一位哈希值为0的记录，而不用考虑全局深度所说的前两位哈希值。个人认为这是可拓展哈希的精髓。

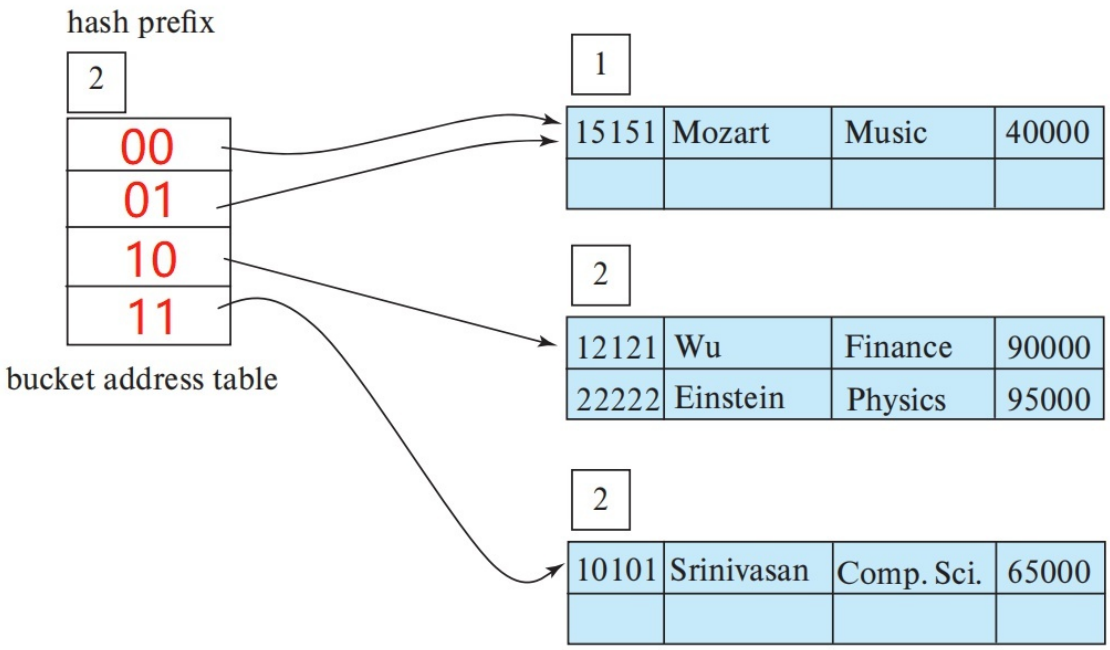


Figure 24.12 Hash structure after four insertions.

第四步：插入的数据无法用分裂解决。

可拓展哈希允许插入哈希值相同的记录。所以当插入三条哈希值一样的记录，一个桶就一定放不下（假设桶的容纳上限是两条）。就像图中所示，这种情况被叫做overflow，而解决方法是用指针指向overflow bucket，也就是人为增加桶。这种方式看上去不美，也暴露了可拓展哈希的局限性，但一个方法在实际应用中确实无法确保永远的统一性，总是会需要“补丁”。实际应用中，可拓展哈希也不是最普遍的方法，更多则是B-Tree。

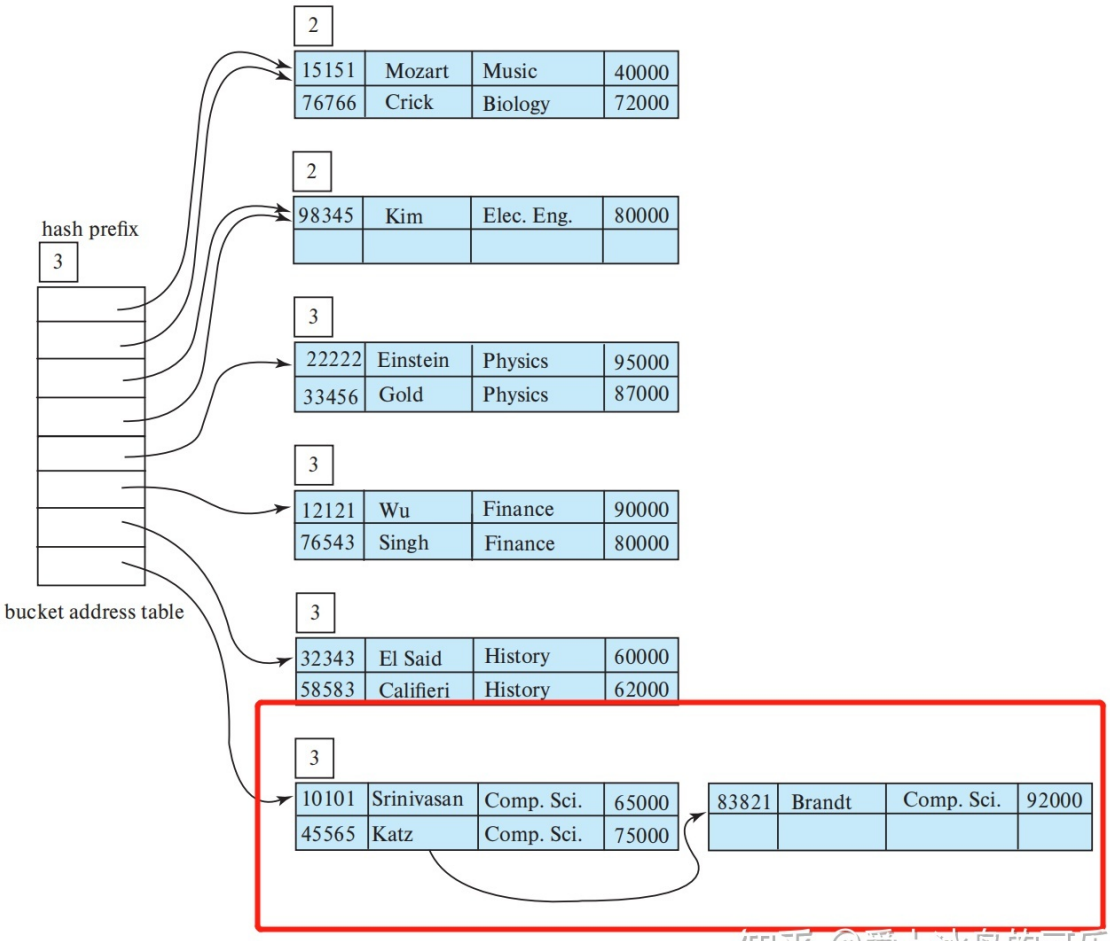


Figure 24.16 Extendable hash structure for the *instructor* file.