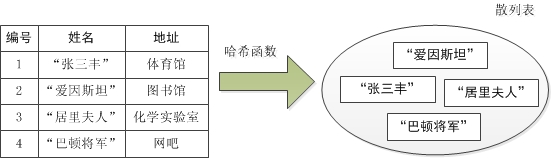
散列表是怎么进行查找的？

**散列过程**

整个散列过程其实就是两步。

1. 在存储的时候，通过散列函数计算记录的散列地址，并按此散列地址存储该记录。

就像张三丰我们就让他在体育馆，那如果是“爱因斯坦”我们让他在图书馆，如果是“居里夫人”，那就让她在化学实验室。如果是“巴顿将军”，这个打即时战略游戏的高手，我们可以让他到网吧。总之，不管什么记录，我们都需要用同一个散列函数计算出地址再存储。



2. 当査找记录时，我们通过同样的散列函数计算记录的散列地址，按此散列地址访问该记录。说起来很简单，在哪存的，上哪去找，由于存取用的是同一个散列函 数，因此结果当然也是相同的。

* 所以说，散列技术既是一种存储方法，也是一种查找方法。然而它与线性表、 树、图等结构不同的是，前面几种结构，数据元素之间都存在某种逻辑关系，可以用连线图示表示出来，而散列技术的记录之间不存在什么逻辑关系，它只与关键字有关联。因此，散列主要是面向査找的存储结构。

**散列表的优势与劣势**

散列技术最适合的求解问题是査找与给定值相等的记录。对于査找来说，简化了比较过程，效率就会大大提高。但万事有利就有弊，散列技术不具备很多常规数据结构的能力。

比如那种同样的关键字，它能对应很多记录的情况，却不适合用散列技术。一个班级几十个学生，他们的性别有男有女，你用关键字“男”去査找，对应的有许多学生的记录，这显然是不合适的。这个时候可以用班级学生的学号或者身份证号来散列存储，此时一个号码唯一对应一个学生才比较合适。

同样散列表也不适合范围查找，比如査找一个班级18-22岁的同学，在散列表中没法进行。想获得表中记录的排序也不可能，像最大值、最小值等结果也都无法从散列表中计算出来。

* 我们说了这么多，散列函数应该如何设计？这个我们需要重点来讲解，总之设计一个简单、均匀、存储利用率高的散列函数是散列技术中最关键的问题。重复一遍，设计一个合适的散列函数最重要！

**哈希冲突**

另一个问题是冲突。在理想的情况下，每一个关键字，通过散列函数计算出来的地址都是不一样的，可现实中，这只是一个理想。

我们时常会碰到两个关键字key1 ≠ key2，但是却有f (key1) = f (key2)，这种现象我们称为冲突（collision)，并把key1和 key2称为这个散列函数的同义词（synonym）。出现了冲突当然非常糟糕，那将造成数据査找错误。尽管我们可以通过精心设计的散列函数让冲突尽可能的少，但是不能完全避免。于是如何处理冲突就成了一个很重要的课题，这在我们后面也需要详细讲解。

* 处理哈希冲突是个大坑……需要深入了解的东西很多。这里简单预告下，通常有两类方法处理冲突：开放定址(Open Addressing)法和拉链(Chaining)法。有兴趣可以跳到后面章节先了解下。