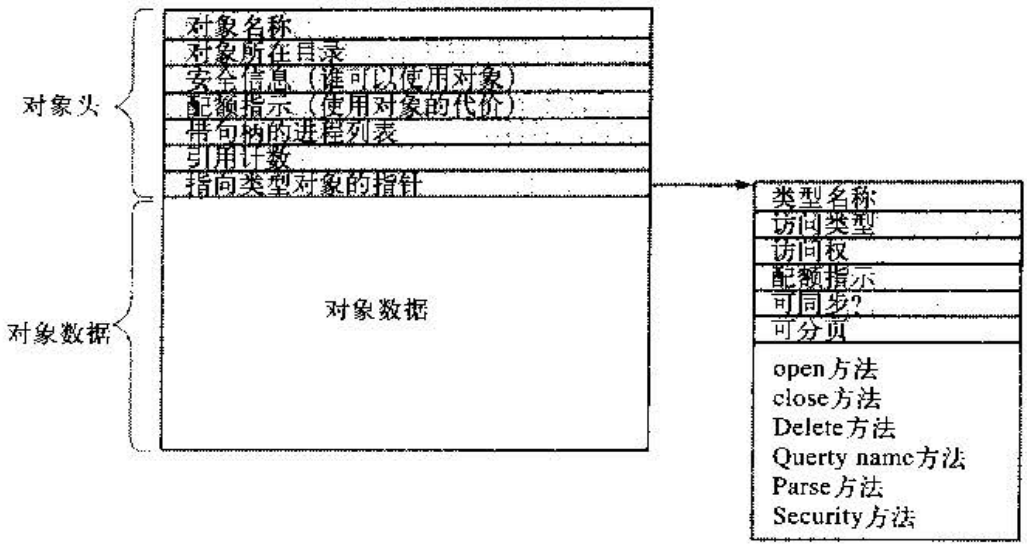

图11-17 对象管理器管理的执行体对象的结构

每个对象标头包含一个配额字段，这是用于对进程访问一个对象的配额征收。配额是用来保持用户使用较多的系统资源。对无分页核心内存（这需要分配物理内存和内核虚拟地址）和分页的核心内存（使用了内核虚拟地址）有不同的限制。当内存类型的累积费用达到了配额限制，由于资源不足而导致给该进程的分配失败。内存管理器也正在使用配额来控制工作集的大小和线程管理器，以限制CPU的使用率。

物理内存和内核虚拟地址都是宝贵的资源。当一个对象不再需要，应该取消并回收它的内存和地址。但是，如果一个仍在被使用的对象收到新的请求，则内存可以被分配给另一个对象，然而数据结构有可能被损坏。在Windows执行体中可以很容易发生这样的问题，因为它是高度多线程的，并实施了许多异步操作（例如，在完成特定数据结构之上的操作之前，就返回这些数据结构传递给函数的调用者）。

为了避免由于竞争条件而过早地释放对象，对象管理器实现了一个引用计数机制，以及引用指针的概念。需要一个参考指针来访问一个对象，即便是在该物体有可能正要被删除时。根据每一个特定对象类型有关的协议里面，只有在某些时候一个对象才可以被另一个线程删除。在其他时间使用的锁，数据结构之间的依赖关系，甚至是没有其他线程有一个对象的指针，这些都能够充分保护一个对象，使其避免被过早删除。

1. 句柄

用户态提到内核态对象不能使用指针，因为它们很难验证。相反内核态对象必须使用一些其他方式命名，使用户代码可以引用它们。Windows使用句柄来引用内核态对象。句柄是不透明值（opaque value），该不透明值是被对象管理器转换到具体的应用，以表示一个对象的内核态数据结构。图11-18表示了用来把句柄转换成对象的指针的句柄表的数据结构。句柄表增加额外的间接层来扩展。每个进程都有自己的表，包括该系统的进程，其中包含那些只含有内核线程与用户态进程不相关的进程。

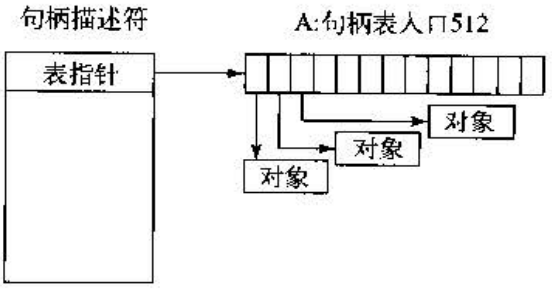

图11-18 使用一个单独页达到512个句柄的最小表的句柄表数据结构

图11-19显示，句柄表最大支持两个额外的间接层。这使得在内核态中执行代码能够方便地使用句柄，而不是引用指针。内核句柄都是经过特殊编码的，从而它们能够与用户态的句柄区分开。内核句柄都保存在系统进程的句柄表里，而且不能以用户态存取。就像大部分内核虚拟地址空间被所有进程共享，系统句柄表由所有的内核成分共享，无论当前的用户态进程是什么。

用户可以通过Win32调用的CreateSemaphore或OpenSemaphore来创建新的对象或打开一个已经存在的对象。这些都是对程序库的调用，并且最后会转向适当的系统调用。任何成功创建或打开对象的