OODB存储管理子系统的设计*)

李伟华　　胡守仁（国防科技大学计算机系）

摘　要

GOODbase是我们正在SUN工作站上开发的一个通用的面向对象数据库系统，本文简单介绍了该系统的总体结构及其存储管理子系统的逻辑结构，详细描述了它的内存管理的实现方案，包括对象存储器的划分、对象数据结构的建立、对象的引用、内外存对象的交换等方面的设计.GOODbase存储管理系统的设计充分考虑了对内外存对象的快速访问。

面向对象数据库（*OODB*）是程序设计语言、数据库技术、软件工程、人工智能以及非传统数据库应用等方面研究的共同产物，它从八十年代中期一出现就显示了强大的活力，并迅速形成了世界范围的研究热潮。总的来看，OODB是一个程序设计语言与数据库的集成系统，它能很好地支持上述各个方面的应用，但主要的特征仍然体现在数据库功能方面。绝大多数OODB都可以不太严格地划分为两个主要子系统，即存储管理子系统以及一个运行于其上的解释器，存储管理子系统是诸如永久性、并发控制、恢复、一致性和查询等数据库功能的最基本的实现支持。

如同传统数据库一样，在OODB中，所有的数据对象都存放在辅助存储器上，处理时必须送入计算机主存储器中。为此要在主存中设立若干管理表格，开辟若干个系统缓冲区，作为数据库系统的所有系统模块与辅助存储器之间的虚拟存储器接口，在这个接口上，可以直接访问数据对象。存储管理系统就是负责这些管理表格的维护、缓冲区的管理、外存对象的存放、对象在主存与外存之间的移动，新对象的创建、权限的设置、索引的建立、恢复、并发控制等等，并提供高层模块所要访问的对象。

由于面向对象数据模型的思想与传统的关系、层次、网络模型的思想有实质性的区别，所以在面向对象数据库系统中，要引用不同的对象表示与存储结构，以及更为复杂的引用机制，从而对存储管理系统提出了很高的要求。

中经通讯网连接而成的硬软结合的分布式使用的RDBMS体系结构。Ada-DDBMS是我们正在研制的VAX、VMS、DECNET运程网上的Ada相关的分布式关系型数据库管理系统。这种作法对Uniface而言是增加了分布式的数据管理能力；对Share Base及Ada-DDBMS而言，则是增强了用户界面的功能。为Uniface做一个适应于ShareBase的驱动器，或做一个适应于Ada-DDBMS的驱动器将是不难办到的，我们正在为此努力创造条件。最后衷心感谢陈兴明，邢晶先生为我们提供了资料！

参考文献

1. Introduction to Uniface, Uniface B.V. Amsterdam 89.4
2. 徐泽同，涂健，共享库概述，计算机科学 90.6期
3. 徐泽同，Ada相关的分布式数据库管理系统 Ada-DDBMS，计算机科学 90.1期

*) 国家自然科学基金与863高技术学助研究项目

GOODbase数据库是我们正在研究开发的通用的面向对象知识处理系统（GOOKPS）的一个重要部分，下面我们简要描述GOODbase的总体结构与存储管理子系统的逻辑结构，并详细论述GOODbase内存管理子系统的具体实现，最后说明我们今后的研究工作。

一、GOODbase概览

1. 总体结构

目前，我们已经着手在SUN工作站上建立GOODbase原型系统，其体系结构包括八个子系统，如图1所示。

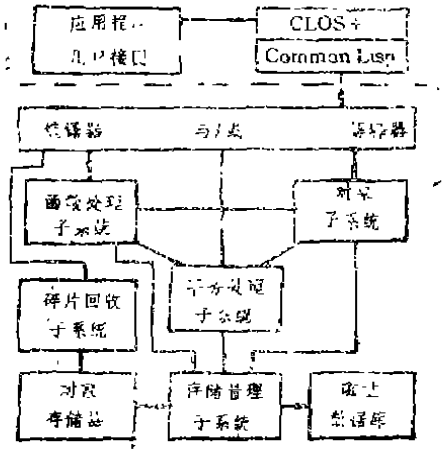

图1 GOODbase的体系结构（CLOS+表示CLOS加上查询语言）

其中存储管理子系统是原型系统实现中较为重要的部分，它至少应该包括a)主存对象存储器的访问管理程序，负责查找磁盘上特定的对象并进行相应的操作；b)数据库磁盘管理程序，提供对磁盘对象的访问，管理磁盘页面，在主存与磁盘之间移动对象，如果需要的话，还完成对象的磁盘格式与主存格式的转换。另外，存储管理子系统还为面向对象系统中的各种重要表格提供了具体实现或支持。

对象存储器设于主存中，存储各种临时对象或缓冲对象，可根据需要在其中划分若干缓冲池及管理表格，由存储管理子系统进行管理。碎片回收子系统可以看作存储管理系统的一部分，强调它是为了说明在面向对象系统中，绝大多数对象是临时对象，只有很少一部分作为永久性对象存回磁盘数据库中，因此存储器碎片的回收对系统性能的影响很大，应该仔细权衡其中的得益与开销，最好采用均匀回收算法。磁盘数据库存放永久对象。

2. 存储管理系统的逻辑结构

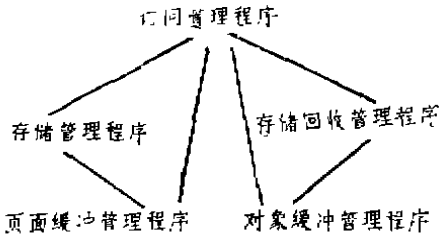

图2 GOODbase的存储管理系统

存储管理程序以磁盘格式操纵对象，负责对象的磁盘格式与内存格式之间的转换，完成对象在主存与外存之间的移动。访问管理程序控制对象在对象缓冲池与页面缓冲池之间的移动，以及对对象的创建、删除等操作。页面缓冲管理程序与传统数据库系统的缓冲管理程序一样，管理页面池并实现一个页面替代算法，页面缓冲池用作小对象的后备区并作为大对象的缓冲区。对象缓冲管理程序完成三个功能：一是管理对象缓冲池；二是维持内存驻留对象表(ROT)，用于登记对象缓冲池的对象；三是管理对象描述符驻留区。只有一个物理上的对象缓冲池，多个应用程序可以同时访问缓冲池中的对象，通过创建新对象和向GOODbase发送对象请求，应用程序能将对象汇集在对象缓冲池中。存储回收管理程序负责删除对象时的存储回收，以及在内存用完时将老对象存回磁盘上，腾出内存空间。

对象缓冲池中对象的缓冲管理要比页面的缓冲管理复杂得多，这是因为对象大小的差异性。对新检索到的对象的安置并不是一件容易的事，因为必须找到一块至少具有与对象一样大的空闲存储块。当不同大小的对象在存储器中排入排出时，缓冲池的碎片问题就变得越来越严重了，可能会不时地要求对对象缓冲池进行昂贵的压缩，这样，我们

需要使用碎片回收技术来回收非活动对象占有的存储空间。

二、GOODbase的内存管理

下面我们主要讨论对内存管理的低层实现的设计与支持。

1. 对象存储器的划分

在主存中，由数据库管理系统进行管理的部分称作对象存储器，根据面向对象的特点，我们将对象存储器划分为五个主要部分，其中对象表驻留区与页面缓冲区的大小是固定的，对象描述符基区与对象缓冲基区也是固定的，但它们使用公共的扩展区。

对象表驻留区存放一个大对象表，该表

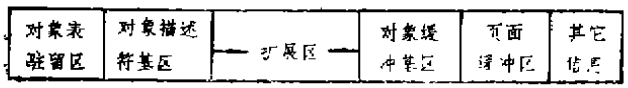

图3 对象存储器的划分

登记了存放在对象缓冲区中的所有对象，内存中的每一个对象都有一个驻留对象描述符（ROD），存放关于该对象的信息，所有的ROD都存放在对象描述符区中。对象缓冲区存放实际对象，由于对象的大小不同，这一部分的管理相对复杂。页面缓冲区存放从磁盘调入内存的页面，能支持对象的预取。

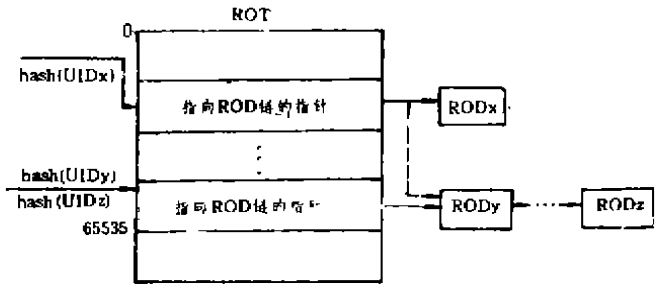

图4 ROT的结构

2. 内存驻留对象表的设计

GOODbase采用了内存驻留对象表（ROT）的概念，ROT表中登记了所有存放在内存中的对象对任何对象的访问首先都要通过ROT才能进行。ROT表大致（但不唯一）确定了内存中所能存放的最大对象数目。GOODbase的ROT中设立65536个表项，由于这是一个很大的表，并且是在启动系统时预先分配内存，不能回收表项，所以在实际设计时，对ROT表项作了最大限度的压缩，其结构如图4所示。

ROT本身并不支持hash冲突的消解，冲突的解决在ROD中进行，这里就可能产生一个问题，当RODx被回收，对象y又利用RODx释放的存储单元建立RODy后，如果又有对对象x的引用，那么就得查遍以RODy开始的整个ROD链才能知道对象x此时并不在内存中，其各时间开销毫无疑问是没有意义的。幸好，实际情况并非如此，我们的模拟实验表明，是否在ROT中解决hash冲突，对系统性能没有影响，这是因为ROT表项所指向的ROD链的长度绝大多数为1，绝少达到3，并且在对象存储器的ROD区采用了特殊的组织方式，空闲的ROD单元加在自由ROD链的最后面，要隔很长时间才能用到所释放的RODx单元，如果在此之前重新引用对象x，RODx完全不必重建，而只须从自由ROD链中取下相应的ROD单元并置其忙闲位为1即可。这种实现方式是GOODbase的一个特色，也是提高系统性能的有效手段。

通过系统中唯一标识符（UID）去访问一个对象的请求被直接送到访问管理程序，它调用对象管理程序首先搜索ROT，由于ROT可能变得很大，所以使用一个hash表根据UID来加速相联检索。hash表的键字是UID，其值是一个指针，指向与该UID相联系的对象的描述符。

3. 驻留对象描述符

当应用程序发消息请求访问一个对象时，GOODbase返回一个指针，指向对象缓冲池中该对象的描述符，而不是指向对象本身的指针，这也是LOOM中所使用的方法。ROD是ROT到实际对象之间的一个结构固定的中间数据结构，能支持ROT表项hash冲

突的消解，当创建一个新对象或者从外存调入一个对象时，首先要建立该对象的ROD。ROD采用图5所示的结构。

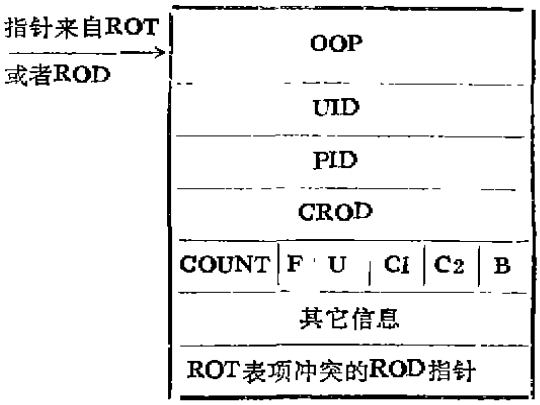

图5 ROD的结构

ROD包含指向实际对象的指针OOP，通过它可以访问该对象的内容；UID域包含该对象的UID；PID域包含该对象的磁盘物理地址；类的ROD域CROD是指向该对象所属类的ROD的指针；F指示是否有ROT表项的冲突；U用于指示该对象是否被修改过；COUNT是对象的引用计数；C1指示该对象是否类对象；C2指明该对象是否组合对象；B是该ROD结构的使用位。

我们引入ROD，是因为需要满足定位内存对象的两个有点相冲突目标之间的一个折衷，一方面，我们想在用户通过UID请求对象时返回一个实际对象（实际上是一个指向对象本身的指针）。另一方面，我们需要保留在主存中排出任何对象的能力，当这种要求出现时，例如，当内存中老对象太多时。但是，没有直接的硬件支持，就没有一个容易的方法去获取对排出对象的直接引用并采取适当的操作。

当一个对象没有任何活跃事务引用或者对象缓冲池已满时，就可能排出该对象，然后，将ROD中指向该对象的指针改为NIL，要是没有其它指向它的指针的话，ROD本身也被回收。

在内存中，我们设置了专门的ROD驻留区，ROD驻留区划分为许多与ROD的结构大小相同的单元，自由的ROD单元存放在一个自由ROD链表中。只有在ROD驻留基区中的自由ROD单元都用完之后才使用扩展区。

4. 主存对象的存储结构

由于面向对象系统中的对象分为类对象与实例对象，前者比后者包含有更丰富的信息，所以在GOODbase中，我们为类对象与实例对象设置了基本相似但又不尽相同的存储结构，图6给出了实例对象的存储结构。

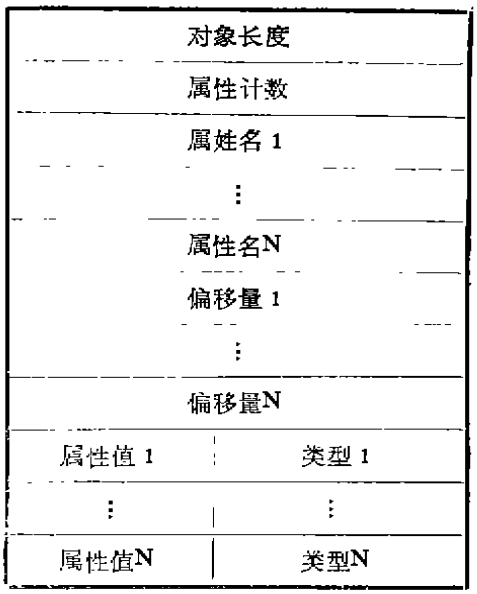

图6 实例对象的存储结构

类对象除具有上述信息外，还应该包括指向类的实例对象集合及类的直接子类集合的指针，这两个集合的设置是为了支持数据库查询的处理。另外，磁盘对象的表示与内存对象的表示也有不同之处，两者之间存在一定的转换关系，由存储管理程序确定对象的PID，这要通过一个对所有数据库对象的UID-PID表进行hash（该表不同于ROT），限于篇幅，这里不再说明。

我们已经用C语言在SUN工作站上实现本文所给出的面向对象数据库存储管理思想的主要部分，即系统内存的管理，目前正在进行的以及今后的工作将包括内外存对象的转换、磁盘对象的访问及预取、数据库事务处理、高层的数据库功能的实现等。

WHYMX存储系统的设计方法

毛兆余　王能斌　董玮　徐庆云　杨传钧

（机电部第11研究所，北京）

摘　要

The concept and design of the simplest common interface(SCI) are firstly presented in the paper. Then the construction of the storage system of WHYMX is simply introduced. Consequently, an approach is described for extending the storage system. several problems with the storage system expansion are also analysed.

目前，许多新的应用都要求数据库系统提供扩展功能，以便用户随时可以定义自己的数据类型及其操作。但是，传统的数据库系统无论在数据模型，还是在存储结构及其存取方法等方面，都不能有效地提供这种支持。尽管有些数据库系统允许用户增加一些自己定义的数据类型及其操作，但是这种扩展是通过查询语言或嵌入到应用程序中来实现的，况且这些数据库系统的基础DBMS在基本存储方面并没有支持这种扩展，从而这种扩展方法在通用性上受到限制，而且还给查询及查询优化带来了困难，影响系统性能。因此，数据库管理系统怎样才能有效地支持用户的扩展呢？即允许用户在数据库系统上添加适当的存储结构和存取算法，以适应于自己扩展的数据类型及其操作。这已成为当今数据库领域研究的重点之一。

面向对象数据库管理系统的"扩展"之一是：存

参考文献

[1] M. Atkinson et al, "The Object-Oriented Database System Manifesto", Proceedings of the 1st DOOD, Japan, Dec. 1989。中译文见《计算机科学》1990№.3

[2] J. Banerjee et al, "Data Model Issues for Object-Oriented Applications", ACM TODS, Jan. 1987

[3] Goldberg and D. Roboson, "Smalltalk-80: the language and its implementation", Addison-Wesley, 1983

[4] S. F. Hudson and R. King, "Object-Oriented Database Support for Software Environments", Proceedings of the ACM SIGMOD Conference, May 1987

[5] T. Kaehler, "Virtual Memory on a Narrow Machine for an Object-Orented Language", OOPSLA'86

[6] W. Kim, N. Ballou, H. T. Chou, D. Woelk, J. Banerjee, "Integrating an Object-Oriented Programming System with a Database System", OOPSLA'88, Sep 1988

[7] D. Maier et al, "Development of an Object-Oriented DBMS", Proceedings of the OOPSLA'86, Sep. 1986

[8] 田中克己，"オブジェト指向データベースシステム-その背景と概念", bit, Vol.20, No.6, 1988.

[9] 李伟华、王志英、慈云桂，"面向对象数据库研究"，计算机工程与科学，1990年第3期

[10] 胡守仁、慈云桂、王志英、李伟华、吴建清，"通用的面向对象知识处理系统GOOK-PS"，第九届全国数据库会议论文集，1990.9，上海