维普资讯 http://www.cqvip.com

OBMS 面向对象 IDKE 数据库 存储

计算机科学1992 Vol.19 No.2

OBMS/IDKE中对象存储/存取子系统的若干关键问题*)

49-55 TP311.13

王 珊 任永杰 唐元昌

(中国人民大学数据与知识工程研究所，北京100872)

摘 要

The storage/access subsystem is one of the kernal module in OBMS/IDKE. It should support the object-oriented data model PUC, store and manage massive, shared and permanent object effectively and efficiently. In this paper, we discuss some important issues and main ideas in designing and implementing the storage management, buffer management and access path management of OBMS/ IDKE.

一个数据库系统称为是面向对象的数据库系统首先应满足以下两个条件：

(1) 支持一个面向对象数据模型内核。

(2) 支持传统数据库所有数据库成分。

当然为了与面向对象数据模型内核的语义一致，必须作适当的语义扩充和修改。

从文[1]图1中可以看到OBMS/IDKE是一个面向对象数据库系统。其体系结构包括了传统的DBMS几乎所有的模块，只是在OO环境下有了新的内容，有些则是传统的DBMS没有的。OBMS/IDKE核心可以大致分为三部分：

Data Model, VLDB 87, pp83-96

[13] edited by Stonebracker, M. etc., The POSTGRES PAPERS, 25 June 1987

[14] Weiser S.P. etc., OZ+:An Object Oriented Database System, Object-Oriented Concepts, Databases, and Applications, pp309-337

[15] Kim, W., Features of the ORION Object-Oriented Database System, Object-Oriented Concepts, Databases and Applications, pp251-282

[16] Bretl, R., The GemStone Data Management System, Object-Oriented Concepts, Database, and Applications, pp283-308

[17] 王珊、任永杰、唐元昌，OBMS/IDKE对象存贮/存取系统的若干关键问题，本期《计算机科学》

[18] 唐元昌、王珊，OBMS/IDKE，模型及其无缝的面向对象的编程语言，本期《计算机科学》

[19] 萨师煊，王珊，数据库系统概论(第二版)，高教出版社

[20] X3/SPARC/DBSSG/OODBTG FINAL REPORT 12-SEPT-1991

*)国家自然科学基金资助项目

维普资讯 http://www.cqvip.com

（1）界面部分。包括：面向End-User的交互式查询界面——Object Browser；面向对象的数据库编程语言ODPL及语言处理器；面向系统管理员的系统生成模块。

（2）存贮/存取子系统。包括：对象存贮管理；缓冲区管理；并发控制／恢复；存取路径管理；事务管理。

（3）对象管理子系统。包括：类型管理器、方法（函数）管理器；版本管理器。

作为OBMS/IDKE中核心模块之一的存贮／存取子系统，其主要任务是管理和存储大量的共享、持久对象，而不仅仅是处理挥发性对象，这也是一个面向对象数据库系统的区别或者说其复杂性所在。限于篇幅，本文仅对其中若干关键问题作一讨论。

一、对象数据的分段存贮策略

对象的存贮由两部分组成。第一部分是数据库类的结构信息和语义关联，包括数据项构成、函数动作、和其他类的语义关联等。因为同一类的所有对象具有相同的结构和行为，因此这些信息只需存贮一次。我们把它存放在该类的数据字典中。函数体的源代码和目标代码则分别存于系统的源代码库和目标代码库中，有关这部分的存贮策略和技术请参见[2]。第二部分是该类对象的实例数据，它们被一一存放在对象库中。

OBMS/IDKE支持一个语义严格的OO模型——PUC模型[1]。在该模型中对象具有下列主要特征：

1.穷尽地描述一个对象的是它的抽象格。抽象格中超类、子类之间有紧密的语义关联。

2.一个对象的组成成分（域）可以是一个简单类型(type)、集合、结构、甚至是一个对象。允许嵌套集合、嵌套结构、嵌套对象。对象的嵌套层次不限。因此对象域的长度、对象的长度均是变长的而且常常是大对象。

OBMS/IDKE中对象的存贮不能简单地沿用传统DBMS中的技术，而采用了分段存贮的策略。对于一个抽象格中的对象，其信息项的存贮是按其抽象类分段存放的。具体地讲，设一个对象有n个抽象类，那么这个对象的所有数据信息将按这些抽象类分割成n个片段，每个段由直接定义在该抽象类中的那些信息项（不包括从其超类继承来的信息项）组成。这n个分段分别存放在n个类所对应的数据文件中。每一片段是该类文件中的一个记录，因此这个对象所有信息由n个记录构成。

对于合成对象，OBMS/IDKE也采用把主体对象与成员对象分开存贮的策略。主体对象中相应其成员对象的位置存放成员对象的OID，而成员对象则存放在该类对应的数据文件中。

对象的分段存贮策略具有以下优点：

1.查询效率高，尤其是对那些"从某个类中找出满足条件的所有对象"这样的查询。

2.能很好地支持对象变迁。如某一对象在某一时刻变为其子类中的一个实例或退化为其超类中的一个实例，此时只需增减某些片段，只对某一类的数据文件进行修改而不会影响其他抽象类的存贮。

3.方便大对象的处理。一个大对象在按抽象类分割后就变成了一些较小的片段，这些片段具有相对的独立性，可以单独调入内存处理。

4.能为不同版本的对象提供共享机制。如果一个对象的不同版本在某些数据片段上没有差异，则这二个版本在这些数据片段上可以共享同一份存贮。

对于结构(STRUCT)的存贮，我们在物理上直接支持其域嵌套，STRUCT不进行单独存贮而是把它存在所属对象中。对STRUCT嵌套层数作了限制(最多嵌套两层)，对于超过两层的STRUCT嵌套可以要求用户使用数据库类来定义。这样，OBMS/IDKE一方面在物理上支持一定层次的域嵌套，另一方面通过数据库类又能在逻辑上定义并支持无穷多级的域嵌套。

维普资讯 http://www.cqvip.com

二、对象的内部标识及其存贮方法

PUC模型支持对象标识符的概念。对象标识分为内部标识和外部标识两种。对象内部标识符(OID)是一个全系统范围内唯一标识一个对象的数据项。由OBMS系统分配。一个对象一经建立，OBMS就赋于它一个OID直到该对象被彻底删除。用户不能对OID进行修改。在PUC模型中一个对象可以看作是其抽象格中任何一个类上的对象，即"映象对象"。如何标识"映象对象"呢？一个对象的不同映象对象既相互独立，又具有紧密的语义联系。如何在概念上和物理存贮上反映这一点呢？为此，我们引入了伪OID的概念，记为OID′，它的构成是：

OID	抽象类

其中，抽象类为该对象进行抽象时所用的类。这样，一个OID′就唯一标识了一个对象在某一抽象类上的映象对象。而一个对象的不同映象对象又通过它们OID′中相同的OID反映了它们是同一个对象的不同抽象层次。

在使用时，当我们在某个类上引用一个对象时使用的是该对象在此类上的映象对象的OID′。

在存贮上，当我们存贮一个对象时，每个存贮片段中所存放的OID′不是该对象在此类上映象对象的OID′，而是它在精确类上映象对象的OID′。也就是说，一个对象的不同存贮片段中，OID′是相同的。因此，OID′和OID一样也是一根主线，将分散在不同存贮文件中的数据片段按其语义联系在一起。

OID和OID′的概念和存贮方法使OBMS具有下列特色：

1.既容易维护对象在抽象格上的语义一致性，又方便地提供了对象抽象的表示、存取和存贮方法。

2.高效而灵活地支持对象的存取。对某一对象的查询入口可以是它的任何一个抽象类。根据查询给出的OID′得到了OID和查询入口的抽象类。根据抽象类从数据字典中得到抽象格中所有抽象类对应的数据文件，依次在诸数据文件中根据OID取出相应的数据片段，拼在一起就得到了该对象完整的信息。

3.OBMS存贮OID′的方法有力地从物理上支持了PUC模型中深层数据隐蔽的概念。如果用户给出了一个OID′并想在它所对应的"映象对象"上进行操纵，例如显示或修改。OBMS将按类似上述的方法取出此"映象对象"。由于从存贮片段中存放的OID′可以得到该对象的精确类，根据精确类和查询给出的抽象类，OBMS就可方便地判断出该数据操纵是否破坏了深层数据隐蔽，若破坏了，拒绝执行．从而保证了深层数据隐蔽的实现。

4.OID和OID′的概念，使PUC模型可支持三个级别的对象相等概念。

强相等（基于OID′的相等）。反映了客观世界中同一实体的同一种抽象方式。

弱相等(基于OID的相等)。反映了客观世界中的同一个实体，尽管抽象方式不同。

值相等(基于值的相等)。反映了客观世界中不同实体之间的值间关系，和传统的关系数据库中值相等的概念是一样的。

三、对象的物理存贮结构和管理

由于对象结构复杂，语义关联丰富及大对象的存在，对象的物理存贮结构和管理方法将极大影响对象存取的效率。OBMS中每一个数据库类对应一个对象（数据）文件，而文件的物理组织采用Hash方法。一个数据库类的所有永久对象存于一个Hash文件中，文件的每一个记录对应一个对象。

OBMS之所以采用Hash方法存贮对象，是因为在诸多物理文件组织中Hash文件能直接、自然地支持对象标识符的索引。给定一个OID，不需借助其他辅助索引就可实现对象的快速查找。为了适应对象长度变化大，尤其是大对象的存贮和管理，我们对Hash文件的物理组织作了相应的改进。其逻辑结构

维普资讯 http://www.cqvip.com

如图1所示。

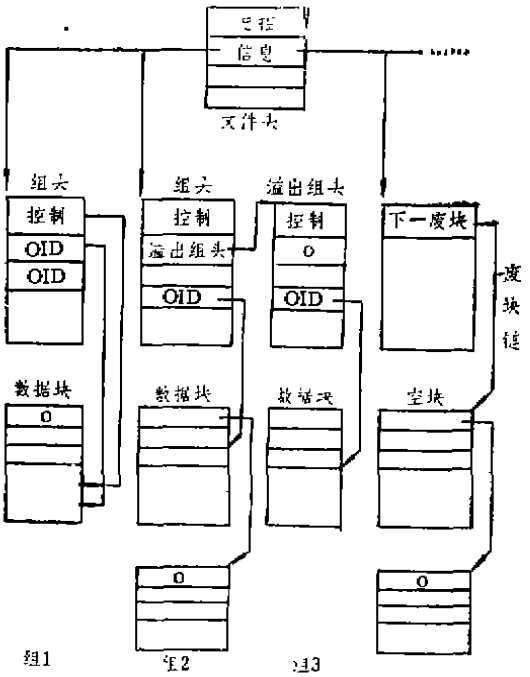

图1 对象数据文件的逻辑结构

每个文件由一个文件头和多个组组成，每个组又由一个或多个组头及一些数据块组成。

文件头记录总控信息，如Hash模数、Hash类型、索引信息、块大小等等。其块大小取值可为2^nK(n=0，1，2，3，4)，标明该文件每一块的大小。Hash模数确定整个文件的组数。每组的组头中记录本组控制信息和组内对象数据情况。每个对象其OID′、对象长度、所在块号、块内偏址等信息均存在组头内。对象数据（不包括OID′）则存到数据块内。所有对象的OID′在组头内按其(块号、块内偏址）排序存放。这样只需存取组头就可以获得(OID，抽象类)，对某些查询不必存取数据块便可完成，提高了效率。同时由组头就可算出每一数据块的剩余空间，对插入操作十分有利。如果一个组内对象过多，以至一个组头块不够存放组头信息时，则放入溢出组头块中。同一组的所有溢出组头用单链相连。

数据块用来存放对象数据。对象记录的物理结构不仅支持变长域、嵌套结构、变长对象的存贮，而且对于对象大小几乎没有限制。对于长度超过一个数据块的大对象，则在Hash文件相应组中从一个新的数据块开始用多个数据块存放它，这些块用链连接起来。

为了满足OBMS/IDKE作为一个通用的面向对象数据库系统核心的设计目标，适应众多的应用领域需要，在存贮结构上我们允许系统管理员根据实际应用中对象的大小、个数灵活设定系统参数。例如，文件块大小不再是全库统一的，可以由1K～16K变化。每个文件的分组个数可以用Hash模数的选择、Hash函数类型的调整来改变，使得对象的分布均匀又能使对象在某些属性值上聚集。

四、缓冲区管理向上透明的对象存取界面

在数据库管理系统中设置数据库缓冲区时，精心设计其优良的管理方法，对于提高系统效率十分重要。由于传统数据库系统中记录大小、文件块尺寸变化有很大限制，所以缓冲区管理相对简单，关键在于缓冲区页块淘汰策略，即使用什么样的页块淘汰算法可以使得物理I/O次数较小。然而在面向对象环境下，由于对象大小相差很大，文件块大小也不划一及大对象跨块存放等情况，对缓冲区管理提出了新的要求。一些现有系统对缓冲区管理都作了一些特殊处理，如在EXODUS系统中，缓冲区有页块和Chunk两种，前者针对小对象页块淘汰，后者专用于处理大对象。当遇到大对象时把其中连续页块调入chunk的缓冲区中。这种方法的缺点是：由于处理不统一，不灵活，导致缓冲区管理算法非常复杂，有时还要处理chunk的连接与分割等问题。此外，难以预料究竟需调入多少连续块。这就可能导致chunk大小选择不当，调入页块数量不合适，使得时间、空间开销均会增大。有的系统使用了硬币

维普资讯 http://www.cqvip.com

孔似的七个大小不同的缓冲区，其中一个专门处理超长对象。这种方法的缺陷是缓冲区大小限制了对象大小。一旦对象大小超过缓冲区大小，系统就终止了，并且当只查询对象的某一部分时导致整个对象的调入，这是不合算的。在OBMS/IDKE中，我们利用C++运算符重载，实现了一个统一的、对上层提供完全透明的对象存取界面的缓冲区管理。

OBMS支持大对象的存贮，不可避免地会出现一次只能调入大对象的一部分进入缓冲区的情况，这样页和页切换的概念是必不可少的。为了使页和页切换对上层程序透明，我们引入一个新的缓冲区指针类型CLASS buf-pointer，应用了C++运算符重载机制。具体的设计思想是：定义新的类型CLASS buf-pointer，其数据成员和成员函数均是public的。数据成员有：实际操作指针char *bp、缓冲区块号blkno、事务标识TID等。在其上重新定义了+(buf-pointer,int)、-(buf-pointer, int)、+=(int)、-=(int)、++( )、--( )等运算符和bufcpy()、bufmove()、bufcmp()、freep()等操作。所有这些操作的语义和正常指针操作的语义类似，只是在其内部必要时执行一定的页切换和FIX、UNFIX操作。相应地，凡是对缓冲区操作的指针全定义为buf-pointer类型，而不再是char*或void*，给上层系统程序提供的操作界面(图2)get-obj( )、first-obj( )等返回的也是buf-pointer类型。

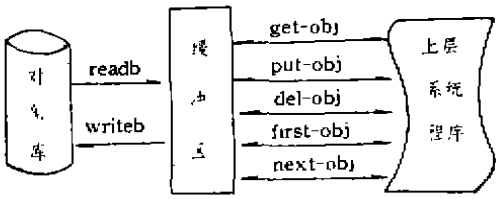

图2 缓冲区界面

作为示例，我们看一个缓冲区操作过程，以考察其怎样实现透明的对象存取。

假如由get-obj(oid, tfd)取得一对象Ox在缓冲区的指针buf-pointer p，现在要对Ox的第n个域进行操作。该域始址相对于p的偏址为int y。此时只需使用buf-pointer q=p+y，即取得Ox的第n个域始址指针（在缓冲区里），并能直接对其进行操作。由于"+"运算符重载，该加法执行时先检查q.bp+y是否超过页界面，若没越过，则直接返回buf-pointer q(p.bp+y,p.blkno, p.TID)；若越界，则计算q应在块的块号、块内偏址，然后将相应块调入缓冲区，返回当前的q值。缓冲区块的FIX标志由上层程序显式调用freep( )释放。有些操作内部(如p+=y)有时也执行必要的freep( )操作。

由此我们可以看到，对象大小、对象存贮是否越过页框对于上层的操作界面是完全透明的，对象的处理也是统一的。在上层程序看来，调用get-obj( )后相应的对象数据片段"似乎"已全部进入缓冲区连续空间，可以在此空间内直接执行操作。此外，缓冲区管理程序每次可以只调入对象要进行操作的那部分进入缓冲区，从而使得对象大小不受缓冲区大小的限制，也避免了在仅对大对象一小部分进行操作（如修改）时将整个大对象全部调入缓冲区带来的时空浪费。

OBMS/IDKE中上层程序只需调用get-obj( )、put-obj、…、next-obj( )(图2)等过程就能实现所有的对象存取。上述过程(del-obj( )除外）返回相应对象在缓冲区的位置指针，上层程序就可直接对缓冲区中的对象进行操作了。

我们采用伙伴系统（Buddy System)管理缓冲区空间。缓冲区块的淘汰按LRU方式进行。由于OBMS/IDKE中各数据文件块的大小可以不同，因此对LRU算法进行了相应的修改。这里就不细述了。

五、小结

我们在研究面向对象数据库系统的存贮/存取子系统时，分析了传统的非关系数据库（网状、层次数据库）和关系数据库的存贮结构和存取方法，非关系数据库系统中，

维普资讯 http://www.cqvip.com

存贮数据记录的同时要存贮记录之间联系的信息，用存取路径来表示记录之间的联系使得沿着物理存取路径（常用的如指针链）来存取记录的效率大大提高，但是带来了数据库操纵语言DML导航的弊端，给用户造成了负担。关系数据库反其道而行之，把存取路径完全向用户隐蔽起来，同时在存贮一级把数据记录和存取路径(常用的如B^+树索引)完全分开。这样，方便了用户，但给系统的查询优化带来了负担。面对PUC模型复杂的数据结构、众多的语义关联，完全沿用非关系数据库的设计方法，不仅会使查询语言的导航更加复杂，而且无法支持动态模式定义和更新(就如非关系数据库那样，模式一经建立就不能修改)。而完全沿用关系数据库的存贮/存取方法又将更加降低查询和操纵的效率。我们吸取了两者的优点，扬长避短，在用户界面上保持关系查询语言高度非过程化的特点，用户使用ODPL语言查询，操纵对象不必进行物理导航。ODPL继承了SQL语言的风格。在存贮结构、存取路径方面，OBMS/IDKE对原有的技术方法进行了扩充和改进，作了重新设计。总结起来主要有以下两个方面：

1.对PUC模型中复杂的数据结构和语义关联，采用了直接存贮和间接存贮相结合的方法。对于嵌套集合，结构及嵌套结构采用直接存贮的方法。我们在对象记录结构中每一记录头、域头及嵌套子域头都有一个标志项用来记录该域长度。对于ISA关联、ISP关联和数量关联等采用间接表示的方法，即不直接存贮相关联的对象，也不用物理指针把它们直接连接起来，而是存贮对象的OID′。在ISA关联中，抽象格的各个抽象类的数据片段中存贮的是对象在其精确类上的OID′，在ISP关联中，主体对象的合成域上存贮的是成员对象的OID′，在数据关联(1:m)中，在1端的对象中存贮m端相应对象的OID′集合。这样，在查询实现中通过间接存贮的OID′实现逻辑（内部完成）导航。加上已知OID′后用Hash方法能快速存取到相应对象，保持了非关系系统高效的存取效率。又因为OID′毕竟不是物理路径从而当动态模式更新时容易修改、维护，使动态模式更新得以实现。

2.对于存取路径，OBMS/IDKE采用了分离(显式)和综合(隐式)两类路径。本系统数据文件的Hash结构已隐含了对OID的Hash索引，把数据和存取路径的存贮综合在文件的同一组内。这类索引只支持==(identical to)和!=(not identical to)两种搜索操作符。我们称这类索引为标识索引（identity index)。此外我们吸收了关系数据库中显式地建立次索引的机制，对对象中某些域建立次索引。这是和对象数据相分离的存取路径，用以支持区间搜索的=、!=、<、<=、>、>=等操作符。这类索引称为等量索引（equality index)，其结构类似B^+树。当然由于对象域变长及可能的超长域，对B^+树结构和维护算法作了特殊处理，详细讨论见[3]。这样，OBMS的存取系统就可以较好地支持ODPL非过程化的查询语言及其优化的实现。

本文仅对OBMS/IDKE的存贮/存取子系统在设计和实现过程中遇到的若干关键问题进行了讨论。不失一般性，这些问题是研究和开发一个面向对象数据库系统中必然会遇到的共同问题。我们的研究和实验工作尚在继续，许多设计和算法需待实验结果验证其优劣。例如缓冲区管理的算法，它能实现缓冲区管理的目标，理论上分析效率也较好，但究竟如何，缺乏实验结果的分析和比较。有些缓冲区参数的确定也难以得到论证，如缓冲区的大小，缓冲区块的大小等等。此外还可以把readb( )，writeb( )过程写成后台进程等等均需进一步考虑。

主要参考文献

[1] 唐元昌、王珊：OBMS/IDKE：模型及其无缝的面向对象编程语言，本期《计算机科学》

维普资讯 http://www.cqvip.com

人工智能 人类智能

计算机科学1992 Vol.19 №.2

人类智能与人工智能

55-59 刘庆波 （150006哈尔滨工业大学计算机系） TP18

摘 要

本文从五个方面对人类智能和人工智能进行了比较，讨论了当前国际上有关人工智能可能性的争论，并对AI研究的发展提出了笔者的见解。

随着历史的发展，时代的进步，现代科学技术使许多古代神话变成了现实，人体的许多器官的功能都可以由机器来模拟了，摄象机模拟了眼，声纳模拟了耳，声音合成器模拟了喉，机械手则可以取代人手完成规定的动作。但作为万物之灵的人类有一个不易由机器来模拟的器官，这就是脑。

智慧和思维是人类所特有的。Simon的物理符号系统假设认为，输入，输出，存储，复制，结构化和条件转换是智能系统的充分必要条件，任何一种信息加工系统，只要具有上述六种功能，就被认为具有了智能。而大脑的功能是非常复杂的，它不仅具有信息加工功能（如计算、推理、归纳、演绎），而且具有非信息加工功能（如灵感、直觉、语言）。其作为信息加工系统的功能可以由计算机来模拟，现有的许多人工智能系统已经证明了这一点，但其非信息加工功能是难以被现有的作为信息加工系统的计算机来模拟的，目前人工智能研究的低谷状态就是一个例证。人类智能是在同大自然的斗争中经过千百万年的进化获得的一种求生能力，人类

〔2〕王宗斌、王珊、武志文：OBMS/IDKE动态模式管理，本期《计算机科学》

〔3〕任永杰：OBMS/IDKE存贮/存取系统设计，1991.5．中国人民大学硕士论文

〔4〕萨师煊等：面向新的应用领域的数据库技术，计算机科学 1989.2

〔5〕王 珊：面向对象的数据库系统，计算机世界，1991.1.10

〔6〕唐元昌：对象库管理系统OBMS/IDKE的模型及其对象操纵，中国人民大学硕士论文 1991.5

〔7〕武志文：OBMS/IDKE系统的动态模式定义和模式更新，中国人民大学硕士论文 1991.5

〔8〕Michael J. Carey, David J. Dewitt, Joel E. Richardson and Eugene J. Shekita, Storage Management of Objects in Exodus. Object-Oriented Concepts, Databases, and Applications, pp.341-369, ACM press, 1989

〔9〕Fred Mellender, Steve Riegel, Andrew Strew, Optimizing Smalltalk Message Performance. Object-Oriented Concepts, Databases, and Applications. pp.423-450, ACM press, 1989

〔10〕Effelsberg, W., Haerder, Principles of Database Buffer Management, ACM Transactions on Database Systems, Vol.9, №4, 1984, pp.560-595

〔11〕Wang Shan, Liu Yi, An Object-Oriented Model for Geographic Information Systems. Conference Proceeding of Information Technology, Advancement Productivity and International Cooperation, the 3th Pan Pacific Computer Conference, Beijing, 1989.pp.552-558

TP59