MIDS/BUAA 存储管理子系统的设计*

车敦仁 周立柱 麦中凡

(清华大学计算机系,北京 100084) (北京航空航天大学计算机系,北京 100083)

摘要 存储管理(特别是辅存管理)是任何一个数据库系统的物理实现层,持久性若不最终落实到辅存上,就是一句空话.本文首先讨论了将程序设计语言和持久对象进行集成的3种途径;然后介绍了MIDS存储管理子系统SMS(Storage Management Subsystem)的设计:(1)SMS的体系结构,(2)MIDS对象的存储结构,(3)对象标识的设计,(4)簇聚与索引,(5)持久性的实现,(6)动态地址转换.

关键词 存储管理,对象,对象标识,簇聚,索引,持久性.

在文献[1—3]中,我们分别对MIDS/BUAA多媒体智能数据库系统的总体设计、数据模型及其特征、模式管理器和对象管理器等进行了系统介绍.存储管理子系统SMS在MIDS/BUAA中处在最底层,是整个系统的基础.对于任何一个数据库系统,SMS设计与实现的好与坏直接影响着系统的性能、系统安全与恢复.

对于MIDS/BUAA而言,SMS的设计又与整个系统的总体结构和数据模型(包括模式和持久性策略)息息相关.因此,下面先就这几个方面进行简单回顾.

MIDS/BUAA由8大功能模块组成,与SMS直接接口的是系统的核心模块:对象管理子系统OMS(Object Management Subsystem).由于库操作总是被封装到事务中,在事务管理子系统TMS(Transaction Management Subsystem)的统一调度之下来完成的,因此更具体地讲,SMS是直接服务于OMS中的TMS子模块的.

每个MIDS对象具有4个元素:(1)唯一性标识:UID,(2)结构特性(一组属性)表征对象的状态,(3)行为特性(一组方法)反映对象具有的行为,(4)对象约束用于限定对象的合法状态.

在持久性的实现上,MIDS定义了3条持久性规则[1],即名规则、传递规则和挥发规则,这样MIDS就能较好地符合P. M. Atkinson定义的持久性正交性和数据类型完全性要求.

MIDS/BUAA的库模式应能反映它的持久性策略.MIDS库模式由下列部分组成[1]:

(1)一个由IKO(Is a Kind Of)关联起来的DBclasses层次结构(简称为IKO层次);

* 本文1993—04—16收到,1993—12—09定稿

作者**车敦仁**,1964年生,讲师,主要研究领域为软件与知识工程,数据库,面向对象,多媒体技术.**周立柱**,1947年生,教授,主要研究领域为计算机软件,数据库.**麦中凡**,1935年生,教授,主要研究领域为软件工程,程序设计语言,数据库.

本文通讯联系人:车敦仁,北京100084,清华大学计算机系软件组

(2)有多少个 DBclass,就有多少个相应的由 IPO(Is Part Of)关联起来的 constituent class/type 层次结构(简称为 IPO 层次);

(3)一个全局持久性名字表 GNT(Global persistent Name Table).

1　程序设计语言和持久对象的集成(方法和比较)

大多数的实际应用,尤其是比较复杂的应用,既要求通用(general－purpose)的程序设计支持,又要求对大量持久共享数据的数据库事务支持.程序设计语言和数据库系统分处在两极,不管是谁都无法单独满足这一对要求.把这二者协同使用的传统作法是把 SQL 嵌入到 PL 中,引起的两个严重弊端是,(1)语义断层和阻抗失配,(2)程序员必须掌握 SQL 和 PL 两个语言.因此便提出了持久性程序设计语言 PPL 和数据库程序设计语言 DBPL 的想法,无论是 PPL,还是 DBPL,都需要探讨将程序设计语言和持久对象进行无缝集成的有效途径.其实,共同基于 OO 范型的 PPL(记为 OOPPL)和 DBPL(记为 OODBPL)又完全可以在 OO 范型的天下走向统一[4].

下面我们介绍并比较 3 种集成程序设计语言和持久对象的方法.

1.1　对象翻译(object translation)法

对象翻译法——数据库对象以磁盘格式(disk－based format)存储,调入内存后被翻译成驻内存格式(in－memory format),写回库中时再翻译成磁盘格式.该方法一般要求双缓冲区,即页面缓冲区和对象缓冲区.ORION[5]就采用此种方法.该方法的优点是,数据库对象(被翻译后)和易变的 PL 对象对于运行环境表现出完全相同的格式.

对象翻译法存在的问题是:

(1)维护数据库对象和它们的内存副本之间的关系,具有较高的复杂性;

(2)对象拷贝和变换引来较高的性能开销,实践表明这是 CPU 时间开销的瓶颈[6];

(3)维护双缓冲区需要更多的内存空间.

1.2　基于磁盘(disk－based)方法

基于磁盘的方法——对象按磁盘格式存储在库中,被调入内存访问时仍使用这一格式,而程序设计语言中的对象(简称 PL 对象),既可以是驻内存格式,也可以是驻磁盘格式,但这两种格式必须隔离.

有不少支持复杂对象的系统,如 Exodus、ODE、O_2 和 Trellis/Owl[6],直接采用面向磁盘的(数据)结构,使用面向磁盘的指针作为对象标识符,并直接将这些指针用于导航.由于基于磁盘的方法和对象翻译法都是建立在两级存储结构上(two－level storage architecture),因而基于磁盘的方法也存在着上述同样的问题.

1.3　一体化管理方法

一体化管理方法——是一种基于虚拟存储空间(singe－level store)的方法.任何对象无论是持久的或易变的、驻磁盘的或驻内存的,均采用统一的对象格式,读入和写出时不进行格式变换,并且对象引用(reference)也采用虚存地址.其实,一体化的单一级存储思想早在 1972 年就提出来了[6],但由于受到下列指责而未受到更多关注:数据库规模严格受限于虚存空间的大小;对恢复的支持不足;对缓冲区的管理低效;页表尺寸太大,严重浪费内存.

近年来由于硬件的飞速发展和 OS 功能朝着 DB 方向的不断完善,加之采用合理的实

现方法,这些指责都已不再成为严重问题.因此,属于单一级存储策略的新一代系统,如Bubba、Objectstore等,开始涌现出来并受到关注.

单一级存储结构的优点可以概括成下列4点:

(1)对象追踪更快.由于使用虚存地址作为对象引用,可得到OS虚存硬件之充分支持.

(2)没有格式转换的一体化对象管理,节省了时间,提高了性能.

(3)最少量的字节拷贝.由于各事务可以直接访问页缓冲区中的对象,既节省空间(单缓冲区),又节省时间.

(4)磁盘I/O减少了.由于仅使用单缓冲区可将节省出的空间进一步用于提高缓冲命中率.

2 MIDS对象的存储管理

2.1 MIDS存储管理策略概述

(1)整个数据库被映射到一个可被多事务共享的逻辑对象库LOB空间中,LOB是在SunOS虚存之上实现的又一层虚存,LOB的大小不受SunOS虚存空间大小的限制.

(2)LOB分页后其页码是一个一维线性空间,我们就用LOB中的页码作为持久对象的UID.未入过库的对象没有UID,但有OID(Object IDentity).

(3)持久对象和易变对象的存储格式基本相同,唯一的差别在于对易变对象按Sun C++中的方式引用.对持久对象的访问需要通过从UID到reference的"动态地址"转换.P++预处理器在对应用进行编译时,就把"动态地址转换"的功能插入其中.

(4)小对象(小于页尺寸)和简单对象(无嵌套)可以被应用的目标码直接访问,它们没有索引.

(5)复合的大对象的indexing和clustering隐含在它们的结构中,它们占有两种页,indexing page和data page.

2.2 存储管理子系统SMS的设计

2.2.1 SMS的体系结构

图1是SMS的体系结构示意图,它很象OS实现虚存的结构图,SMS的设计也正是借鉴了虚存的实现思想.物理对象库(在图中表示为DB—file文件)保存在Sun OS辅存设备上,每个数据库事务(有自己的Workspace)从逻辑上看到的是一个"无限"大的逻辑对象库LOB,但实际访问和操作的却是Sun OS虚存中的一个可供所有事务共享的页缓冲区G.buf.从LOB到G.buf和从G.buf到LOB间的映射是由SMS利用逻辑对象库的页表LPT来完成的.

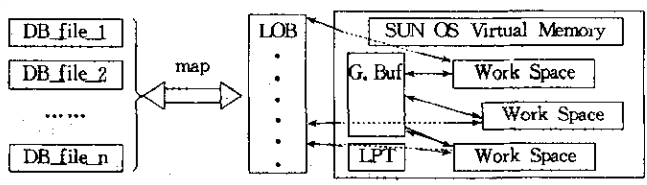

图1 存储管理子系统SMS

2.2.2 对象的存储结构

在 MIDS 中,持久对象和易变对象的存储结构略有差异:对持久对象通过 UID 引用,对易变对象通过 Sun C++中的 reference 引用.因此,在 MIDS 中,对象格式转换的全部意义也就是从 UID 到 reference("动态地址变换")和从 reference 到 UID 的变换.

易变对象的存储格式与 Sun C++环境中的对象格式完全一样.

持久对象的存储格式除了与易变对象在 reference 上的区别外,又分下列两种情况:

(1)简单对象和小对象(尺寸小于一页)无索引;

(2)复合大对象(有嵌套并超过一页)的存储需要两种页面,索引页(index page)和数据页(data page),见图 2.

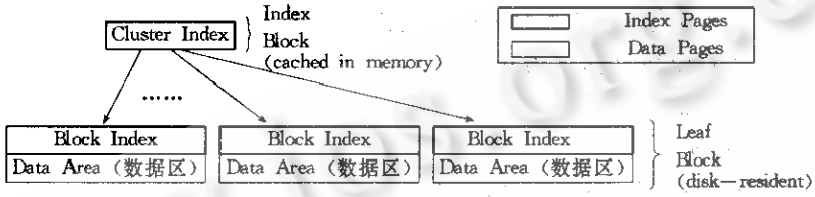

图2 复合大对象及其索引

2.2.3　对象标识

标识(Identity)是一个对象区别于它对象的一种性质,并且是对象的一种内在的抽象性质,对象的状态可以不断变化,但标识是始终如一的.通用程序设计语言和数据库语言各自都对标识进行过几乎独立的探索,但对于标识的支持都不尽理想.标识应同时具有两个独立性[7]:数据独立性和(物理)位置独立性.分析表明支持标识最有力的是使用代理者[7].所谓代理者是由系统产生的、全局唯一的标识符,它具有完全的数据独立性和位置独立性.

基于代理者的标识在理论上是完美的,在实现的性能上又是低下的.因为基于代理者的标识的常规实现方法都是使用一个全局的大对象表.大对象表的维护和查找需要较多的空间和时间.因此许多实际的系统都不采用基于代理者的标识,如 O_2 使用面向磁盘的指针[6],Objectstore 利用 C++中的指针[8].

标识的实现一般与存储结构是有关的.MIDS 的对象标识是逻辑对象库空间中对象所占页簇(page cluster)中的第 1 页的编号.

图 1 中逻辑页表 LPT 就象是页式虚存管理中的页表一样,可起到对象表的作用.除此之外,我们还需为 G.buf 建立相应的页表 GPT,GPT 就好比页式虚存管理中的"快表",所有的驻内存对象均可从 GPT 查出,因此快表 GPT 可以起到驻存对象表的作用.

由于 LOB 本身就是一个虚拟的、概念上的东西,不管在辅存还是在内存,MIDS 对象标识均具有完全的(物理)位置独立性和数据独立性,这显然符合基于代理者的标识要求.

2.2.4　簇聚(clustering)与索引(indexing)

簇聚只对复合对象才有意义,只对于物理(辅存)数据库才有意义.简单对象本来就占有一片连续的存储单元不需要簇聚.

系统为每一个 DBclass 的外延(即该类的全部实例)自动维护一个 B^+_tree 索引,外延的索引相对于每个对象而言是"外索引".

对于复合对象,特别是 Set 型或 List 型复合对象,为了在检索时尽快找到各个成分对象,需要建立对象内部的索引(称为"内索引").

为了减少索引维护的开销,规定只有复合大对象才有资格建内索引.

2.2.5 持久性的实现

如果对象的存储最终不落实到辅存设备上,持久性便是一句空话.欲要持久的对象,必须先通过对象定义子语言 ODL 在创建模式时将之声明为 DBClass(间接或直接)的派生类;或者使它们成为其它持久对象的成分对象(传递规则);或者使之约束于某一持久性名上(名规则).

对持久对象的操作必须封装在事务中进行,在一个事务过程中,无论是持久对象,还是易变对象,对它们的修改都在事务 Workspace 中进行(直接在 G.buf 中进行有哪些问题?).对持久对象的修改被记录在"持久对象修改表"中.在事务提交之前先处理持久对象修改表,并将逻辑上的持久对象拷贝到 G.buf 中.

2.3 SMS 的实现问题

2.3.1 动态地址转换

逻辑页表 LPT 的每个表项的内容如下

有效位	空闲位	合法位	页类型	页簇页数(CPC)	物理地址	内存页号

其中"有效位"表示该逻辑页号是否代表一个有效的 UID(即是否是一个页簇的首页);"空闲位"表示该页当前是否空闲;"合法位"表示该页当前是否在 G.buf 中;"页类型"表示该页是否索引页/数据页;"页簇页数"表示该页所在页簇共占有多少个(逻辑上连续的)页面;"物理地址"指示该页对应的磁盘地址(文件名和相对页号);"内存页号"指示该页在 G.buf 中的页号.

动态地址转换的过程如下:

(1)把 UID 当作逻辑页号查 LPT 页表,找到相应的页表项;

(2)若有效位为 0,则报错并返回;

(3)若合法位为 1,则转(7);

(4)按页簇页数 CPC 在 G.buf 中申请 CPC 个页面(可不连续);

(5)将物理库中该页簇(是连续的)内容拷贝到 G.buf 中相应的页簇中;

(6)修改该页簇在 LPT 页表中各相应表项,然后转(8);

(7)从该页表项查得 G.buf 中的内存页号;

(8)用内存页号作为返回值,返回.

2.3.2 其它问题

1) G.buf 中页面替换采用 LRU 算法,并且每次被淘汰的不是一个 G.buf 页面.而是相关的一簇,即一个对象的全部页面.

2) List 型复合对象按可插入数组(insertable array) 去实现,有序树(ordered tree)结构也是一种较好的选择,O_2 和 Exodus 均采用了有序树结构去实现 List[9].

3) B^+树是表示 Set 型复合对象的较好方法.因为对集合的一般要求是:高效的成员测试和高效的元素浏览.

4) SMS 与 TMS 之间的消息接口,见图 3.其中 get_obj 也意味着对动态地址转换的请求;而其余 3 个消息则意味着对物理库、LOB 和G.buf的管理请求,以及对 indexing 和 clus-

tering 的请求.

5) SMS 作为一个子系统,由下列类簇组成:PhB 类,即物理库类;LOB 类,即逻辑库类;Gbuf 类;Index 类,包括子类:Hash 类,B^+_tree 类和 Ordered_tree 类;clustering 类.

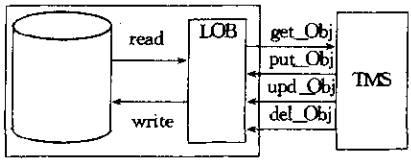

图3 SMS与TMS的消息接口

3 总　结

MIDS 基于虚拟存储管理思想实现了一个逻辑对象库 LOB,无格式转换和大量字节的拷贝,仅需要进行"从 UID 到内存地址"的动态转换,基本上实现了对象的一体化管理. 与 MIDS 存储管理相似的系统是 Bubba 和 Objectstore,它们与 MIDS 的本质差别在于对 UID 的设计不同,MIDS 的 UID 符合 surrogate 标识的要求,因此比 Bubba 和 Objectstore 对标识的支持更强,但因需要动态地址转换,效率要略低一些. Bubba 的库规模无法超越虚存,MIDS 则不受虚存大小的限制.

MIDS 的逻辑页表 LPT 的作用与常规的对象表作用相似. 与采用对象表的方法相比,MIDS 的优点是:(1)留下了挖掘 OS 虚存硬件支持(特别是快表)的潜力;(2)由于对象粒度的离散性大,MIDS 基于页式的管理方法,可把同样的思想用于管理 G. buf,可以方便地支持对不同粒度对象的存储管理. 对物理对象库和全局缓冲区 G. buf 都采用页式管理,提高了管理效率,并可简化管理的复杂度. Clustering 的要意在于对物理对象库中相关对象做相邻存储,以减少 I/O 次数,显示出系统效率. Clustering 对于 G. buf 的意义不大,因此按常规页式方法管理 G. buf 是合理的.

参考文献

1 车敦仁,麦中凡. 多媒体智能数据库系统 MIDS/BUAA 总体设计. 计算机科学,1994,**21(2)**:31—35.

2 车敦仁,麦中凡. MIDS/BUAA 的数据模型及其特征. 计算机科学,1994,**21(2)**:36—42.

3 车敦仁,麦中凡. MIDS/BUAA 模式管理器与对象管理器的设计与实现. 计算机研究与发展,1994,**31(4)**:14—19.

4 麦中凡,车敦仁,葛建. 持久性程序设计语言和数据库程序设计语言. 第十届全国数据库学术会议论文集,沈阳,1992.

5 Won Kim *et al*. Integrating an object oriented programming system with a database system. OOPLSA'88 Proceedings, 1988.

6 George Copeland *et al*. Uniform object management. LNCS416: EDBT'90, 1990.

7 Khoshafian N S *et al*. Object identity. OOPSLA'86 Proceedings, 1986.

8 Lamb C *et al*. The objectstore database system. Communications of the ACM, 1991,**34(10)**:50—63.

9 Deux O *et al*. The story of O2. IEEE Trans. Knowl. Data Eng., 1990,**2(1)**:91—107.

THE DESIGN OF STORAGE MANAGEMENT SUBSYSTEM OF MIDS/BUAA

Che Dunren Zhou Lizhu

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Mai Zhongfan

(*Department of Computer Science and Engineering of Beijing University of Aeronautics and Astronautics, Beijing* 100083)

Abstract Storage management, especially second storage management is the physical implementation level for every database system. Persistence is nonsense without adequate support of second storage management. This paper discusses three approaches for integrating programming language and persistent objects firstly, then introduces the design of storage management subsystem (SMS) of MIDS/BUAA, i. e. (1) the architecture of SMS, (2) storage structure of MIDS Object, (3) object identity design, (4) objects clustering and indexing, (5) persistence implementation, (6) dynamic UID transformation to memory address.

Key words Storage management, object, object identity, clustering, indexing, persistence.