SCOP 对象管理系统*

黄 涛 陈家晚 冯玉琳

(中国科学院软件研究所计算机科学开放研究实验室 北京 100080)

摘要 对象管理系统OMS(object management system)是SCOP组合框架软件的核心.本文简要介绍SCOP对象管理系统的设计与实现,包括内存对象管理系统、外存对象管理系统以及基于Client/Server的并发库系统等.

关键词 对象实例,体,类,范畴,对象管理系统,并发,锁.

随着计算机系统结构的发展,面向对象技术的发展经历了语言为中心、数据库为中心到今天对"操作系统"为中心的研究[1],即以互操作为主要目标的、语言独立、平台独立和位置独立的"操作系统"级支持.SCOP[2,3]组合框架软件即是为此而研制的.

SCOP系统是一个支持应用软件系统开发的面向对象的组合软件构造环境,作为SCOP环境核心部分的对象管理系统OMS(object management system),其任务就是以面向对象的观点来统一管理在软件开发过程中涉及到大量的工具和数据对象,提供一个统一的界面供不同的用户和工具使用对象库.由于SCOP环境支持多用户、多任务,OMS还提供了并发控制机制、权限控制机制等,使用户能在SCOP环境中共享各种对象,安全、可靠地进行软件的协作开发.

SCOP对象管理系统是一个以客户-服务器形式实现的对象库管理系统,具体地说,可分为3个部分:(1)支持对象持久存储的外存对象库.其底层的基本库系统用C++实现,摆脱了对关系数据库系统的依赖;(2)内存对象存储器,为各类用户操作对象库提供了一个统一的界面,并为内外存对象设计了透明的换入换出机制;(3)基于client-server机制的OMS,设计了多级粒度的对象加锁机制,并为死锁、cache一致性和client进程异常终止等问题给出了相应的解决方法.

1 数据模型和总体结构

SCOP对象管理系统采用面向对象的数据模型,支持多继承等各种面向对象的特色.为

---

* 本文研究得到国家自然科学基金、国家863高科技项目基金和国家"八五"攻关项目基金资助.作者黄涛,1965年生,博士,副研究员,主要研究领域为软件工程,程序设计方法学,对象语义理论.陈家晚,1969年生,硕士,主要研究领域为软件工程环境,面向对象的方法和技术.冯玉琳,1942年生,研究员,博士导师,主要研究领域为软件工程,程序设计方法学,面向对象的理论和技术.

本文通讯联系人:黄涛,北京100080,中国科学院软件研究所计算机科学开放研究实验室

本文1996-08-27收到修改稿

了增强系统的描述能力,还引入了范畴(category)、体(body)2 种对象.

当用户在 SCOP 环境中构造一个新软件时,首先利用概念模型分析子系统 OOAnalysis 进行面向对象的分析(分析结果可存于 OMS 中,作为该系统的设计文档),然后用户可在 OMS 中查询所需的对象组件,在新系统中进行复用;如果某些组件在库中找不到,用户可使用系统提供的 ODL 语言定义新的对象类,并将新的对象组件加入到 OMS 中;组合构造软件的全过程是由 PCL 程序控制的,该程序也可以存到 OMS 中以待重用.

从以上的软件开发过程中,我们可以看到对象管理系统的首要任务是管理软件组件,使得用户可以方便地检索并复用这些组件.广义的软件组件不单指某个类或某个模块的源代码,还应包括与之相关的各种文档、中间代码、动态链接库、规范说明、检索信息、持久数据对象等.由于 OMS 系统采用面向对象的数据模式,所以有较强的数据建模能力,可以准确描述以上各种不同类型的数据和它们之间复杂的关系.

SCOP 系统中的编程语言为对象描述语言 ODL,它是一种面向对象的程序设计语言,并支持数据类型共享和持久性对象.为此,OMS 提供了一个函数库来支持持久性编程,包含持久对象的创建、删除、修改、权限控制、并发加锁等.

1.1　对象范畴和对象体

OMS 的主要功能是管理对象组件,以方便软件开发者的复用.库中的对象组件(类)之间主要通过引用、继承等建立联系.但实际使用中用户往往有这样的要求,即按对象组件的功能、适用的领域等方面特征进行分组归类,使库中的对象更有条理,对象组件的查询也更方便.因此,我们采用了对象范畴这一分组形式,用户可以将具有某一共同特征的对象的集合组织在某一范畴中,范畴之间可以有交集.

在 SCOP 系统中,一个对象类还可以有多个实现体,用户可以选择不同的程序语言和不同的算法给出一个类的实现描述,因此每个体对象需记录环境信息(如基语言、库、编译器等)、抽象数据到具体数据的映射、版本信息等.多体的引入,使得 SCOP 系统中可以容纳一个类的不同语言、不同算法的实现,扩大了软件复用的范围.

1.2　SYSOBJ 类

在 OMS 中,所有的数据都被看成是对象,主要有:范畴(category)、类(class)、对象体(body)和对象实例(instance).如图 1 所示,系统提供了 3 个元类:CATEGORY,CLASS,BODY.所有的范畴都是 CATEGORY 类的实例,所有的体都是 BODY 类的实例……这 3 个元类有一个共同的父类 SYSOBJ 类,它是所有对象类的祖先.

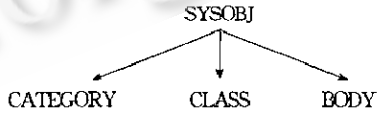

图1　SYSOBJ类和它的子类

1.3　总体结构

OMS 是一个多用户、多任务的对象管理系统,采用 client-server 结构,其结构的剖面图如图 2 所示.client 进程中的应用程序通过 OMS 库接口将对象操作传给内存对象管理系统;内存对象存储器中有一个局部对象 cache,存放着最近存取的对象,如果内存对象管理系统在局部 cache 中找不到所要操作的对象,它就通过下层模块向 server 发出对象传送的请求;server 进程接到请求后,先由内存对象管理系统在其全局对象 cache 中进行查找,若没有,就得调用外存对象管理系统从磁盘上检索到该对象,然后将它传送给相应的 client

进程.

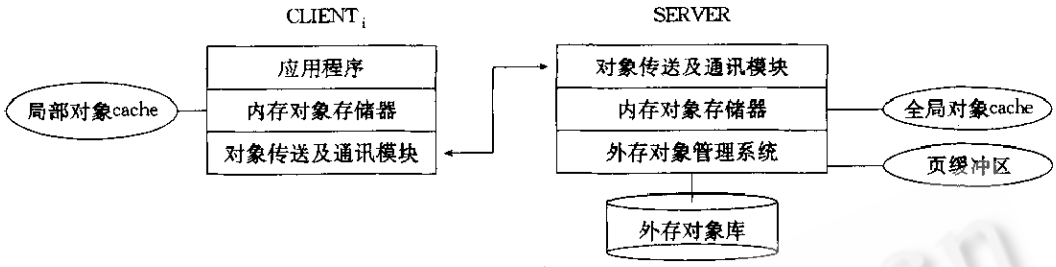

图2 OMS结构剖面图

可以看出,OMS 包含 2 个主要模块:内存对象管理系统和外存对象库. 外存对象库存储持久对象,而内存对象管理系统则需承担临时、持久 2 类对象的管理.

2 内存对象管理系统

内存对象管理系统是 OMS 的顶层模块,它的主要任务是维护一个对象 cache,并完成对象权限管理和换入换出等功能.

2.1 OMS 内存对象管理的特点

OMS 对象管理有如下特点:(1)内存中同时存在着临时对象和持久对象,系统必须统一看待它们,使它们的区别对用户是透明的;(2)一般的面向对象数据库系统中的对象指的是对象实例,而在 SCOP 系统中,将对象范畴、类、体实现、实例都看成对象,OMS 必须用较为统一的方式将它们管理起来,并维护它们之间从属、继承、引用和实例等关系.

2.2 对象标识符的格式和对象的访问方式

根据 OMS 对象管理的特点,我们采用带类型的代理键(typed surrogate)[4]作为对象标识符. 带类型的代理键包含一个类型 ID 和对象 ID 部分,对象 ID 部分由每种类型的计数器产生. 这样,一个对象只要根据它的标识符就可以判断它的类型. 由于采用带类型的代理键形式的标识符,一方面使得对象标识符独立于对象的存储位置;另一方面,标识符的类型信息则有助于多种对象的管理.

在 SCOP 环境中,对象是通过间址的方式来访问的,即通过对象的唯一标识符来访问它. 对象的标识符在其生命周期是不会变化的,而对象在内存中的位置是可以浮动的,浮动对用户或应用来说是透明的. 对象实例在内存中的存在方式如图 3 所示.

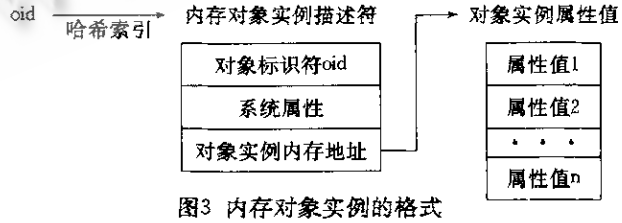

图3 内存对象实例的格式

当程序依据对象标识符(oid)来访问一个对象实例时,首先内存对象管理系统通过哈希索引得到内存对象描述符的地址,进而得到对象实例的内存物理地址. 如果内存查不到该实例,就从外存对象库中读取.

SCOP 用户一般是以对象的名字(字符串)来访问对象的,所以内存对象管理系统用-

个字符串哈希表来管理对象名到对象标识符的映射,以维护对象的名字空间.

2.3　持久对象的换出

由于内存容量有限,当内存对象达到一定数量后,应将一些对象换出内存. OMS 的对象换出采用加权的 LRU 算法,优先换出最近一段时间较少使用的对象,并结合考虑对象类型、对象大小的加权因素,这就保证了内存 cache 中的对象访问有较高的命中率.

2.4　对象权限管理

SCOP 系统是一个多用户的软件开发环境,用户可以在该环境中共享或交换各自的对象. 所以 OMS 必须提供适当的存取权限控制机制,防止 SCOP 系统中的对象被越权访问或恶意删除. SCOP 系统对象的权限分为 3 个部分:owner,group,other. 权限系统的运作方式与 UNIX 系统的文件权限管理类似. 在 OMS 系统中,同样也有特权用户,特权用户是该系统的管理员,可以对所有的对象进行读写操作.

3　外存对象库

外存对象库是 OMS 的底层模块,它的主要任务是记录对象管理系统中需要持久保存的信息,完成对象在外存的物理存储.

3.1　外存对象库中存储的信息

在 SCOP 对象管理系统中,外存对象库主要负责存储持久对象. 其中不仅有普通对象数据库存储的对象—持久实例,还有范畴、类、体这 3 种对象. 以类为例,OMS 需要为它保存的信息包括:该类所属的范畴的信息、该类的参数名表、父类表、继承父类表、子类表、继承子类表、引用类表、被引用类表、属性字典、操作字典、所使用的类型链、权限信息、查询信息、抽象规范文本(即类定义的源代码)等.

由此可见,这些对象在存储器中不是一块定长、连续的数据区,而是一个拥有动态链表的"立体结构",外存对象库必须处理好这些结构的存储.

3.2　外存对象库的总体结构

实现外存对象库大致有 2 类方法:①利用现有的数据库(如关系数据库);②从最底层(文件系统)开始,自己设计对象库. 为了保证系统的独立性和高效率,我们采用后一种方法. 我们首先根据要存储的基本数据设计了一个基本库系统,然后在此基础上构造一个完整的对象库系统. OMS 外存对象库的总体结构如图 4 所示. 不难看出,外存对象库主要由 2 个部分组成,上层是外存对象管理模块,下层是基本库管理系统.

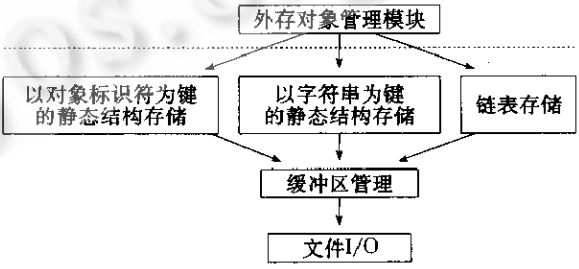

图4　外存对象库的总体结构

根据实际需要,我们设计并实现了 3 种基本库:(1)以对象标识符为键(key)的静态结构库,主要用来存储类和体的主结构以及范畴和实例,这些结构在外存为一块连续、定长的存储区,以对象标识符为键可进行高效的存取;(2)以字符串为键的静态结构库,用来建立对象

名到对象标识符的映射等;(3)链表库,属性字典、操作字典、父类链、子类链等都是动态链表,这些链表的存储要求是:存取效率高,并能做动态的插入和删除.

外存对象库的上层是外存对象管理模块,它负责对象的"拆卸"和"组装"工作,完成内存对象和基本库记录之间的变换,它和基本库一起构成一个完整的外存对象库.

3.3 外存基本库的结构

我们以"以对象标识符为键的静态结构库"为例,介绍外存基本库的结构. 范畴、实例以及类和体的主结构存在以 oid 为键的静态结构库中,对这类基本库的设计要求是:给出一个对象标识符,就可以迅速、高效地对它所对应的结构实施读、写、修改等操作.

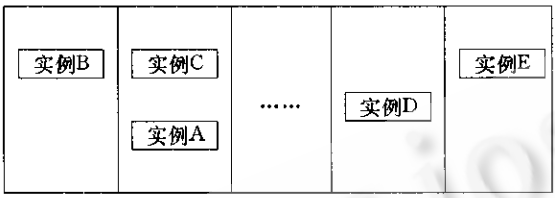

图5 基本库的结构

以实例存储为例,介绍一下静态结构在该库中的存储. 如图 5 所示,每个体的实例集中存储在一个文件中,而一个文件又分成若干页,页是 I/O 的基本单位,每页中含有一定数量的实例. 每个实例依据其 oid 经哈希索引映射到某个页.

"以字符串为键的静态结构库"的结构与上述的结构类似,区别仅在于哈希索引的键(key)不同. 链表库也采用分页结构,但存储策略是:尽量让一个链表存在单页里,这样可以提高链表的存取速度.

3.4 文件档案库

类定义和类实现的 ODL 文本、类设计的文档、体的中间代码等也是对象信息的一部分,也必须存到对象库中. 我们的处理方法是:将这些文件加入到它所属范畴的文件档案库(即 UNIX 系统的 archive 文件)中,将这些文件的有关信息和它们之间的关系记在对象的主结构中,这样只要知道一个对象的对象标识符就可以查到与它有关的所有文件.

4 SCOP 对象管理系统的分布并发实现

SCOP 环境是一个支持多用户、多任务的软件构造环境,OMS 必须提供并发控制机制以支持用户对库中对象的并发访问.

4.1 client-server 结构

SCOP 对象管理系统采用 client-server 结构,是一个"应用程序分布"的系统,它所支持的应用程序可以分布在局部网的各个站点机上,而对象存取则由服务器上的 server 进程统一管理.

如图 6 所示,在服务器上,server 进程和监听进程 daemon 总是处于等待状态. 当 client 进程初始化时,向 daemon 进程发出一个联结请求,daemon 接到这个请求,就与该 client 进程建立一个 TCP 联结(connection),然后 fork 一个子进程 mirror,mirror 进程从父进程继承了 TCP 联结,以后 mirror 与 client 之间的网络通讯就是通过这个 TCP 联结进行的. client 初始化完成后,就可向 server 发出对象操作请求. client 进程先将请求发给与它对应的 mirror 进程;mirror 进程将请求作相应的处理后,再将具体的对象操作发给 server 进程;server 进程在全局对象 cache 上实施该操作,然后将操作的结果发给相应的 mirror 进程,最后由 mirror 进程将结果发还给 client 进程.

4.2　对象加锁机制

在 SCOP 环境中,多个用户可以并发地对库中对象进行存取,所以 OMS 必须提供并发控制机制来保证并发存取的正确性. SCOP 对象管理系统将并发控制集中在 server 机器上,client 机器不参与并发管理,这样可以避免分布的并发控制所需的昂贵代价,使并发控制流程更加简明可靠.

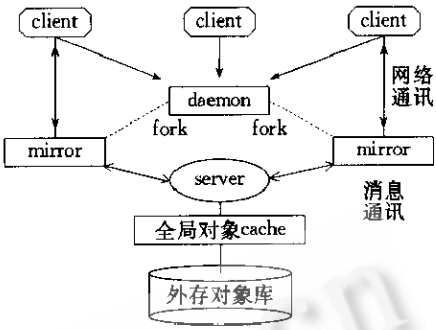

图6　OMS各进程关系示意图

并发控制的基本手段是加锁. 在面向对象系统中,由于类的继承性以及类和实例等多种粒度加锁的需要,数据加锁变得较为复杂. Gaza 和 W. Kim 在 Orion 系统中对这个问题作了详尽的分析,提出了多级粒度加锁的规程[5],是面向对象数据库系统的一个较为完整的数据加锁方案. 在 OMS 中,不能采用 Gaza,W. Kim 这个方案. 因为首先该方案是针对数据库并发事务处理的,其次 SCOP 系统中的加锁粒度比 ORION 多,可在范畴、体 2 个粒度上对对象进行加锁. 我们根据 SCOP 环境并发控制的实际需求设计出与之相适应的加锁机制.

面向对象系统中并发存取的基本要求可归纳如下:

(1)当读写某实例时,不能允许对实例所属的类有写操作.

(2)当对某类做读写操作时,不能允许对其父类(以及父类的父类等)有写操作.

而在 SCOP 对象管理系统中共有 4 级对象:category→class→body→instance,上级对象与下级对象的一个集合相对应,所以当对某一级对象做读写操作时,其上级对象都不能被修改,这是 SCOP 对象管理系统有别于其它面向对象系统的特色.

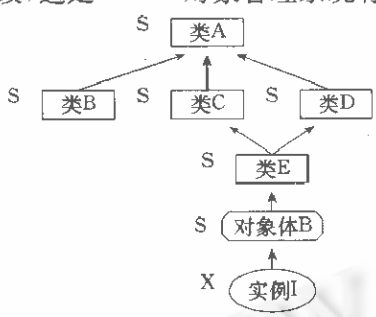

图7　对象加锁实例

考虑类的层次性及多级粒度的加锁要求,我们提出以下加锁方案:

(1)设置基本锁为:独占锁 X 和共享锁 S.

(2)当对某对象加锁时,其所有上级对象也都加上 S 锁.

(3)当对某类加锁时,其所有父类(以及父类的父类等)都加上 S 锁.

在 SCOP 对象管理系统中,由于对象层次关系而派生出来的锁,我们称之为派生锁. 图 7 给出一个对象加锁的实例.

给一个对象加锁,首先应判断该加锁请求是否与锁表状态相容. 当对象锁判定为与锁表状态相容时,可以将本锁和派生锁都加入到锁表中去,完成对象加锁. 判定锁相容的过程如下:(1)求出对象锁的所有派生锁(含上级对象锁和类层次锁). (2)逐一判定本锁和派生锁与锁表的相容性,只有当这些锁全部与锁表相容时,才可以判定此锁与锁表状态相容.

等待队列和锁表是 server 核心的数据结构. 当对象读写或加锁请求因与锁表状态不相容而不能马上实施时,就需将该请求置入等待队列,当条件满足时再实施. 锁表是并发控制的核心结构,它记录着当前系统中各对象的加锁信息(包含加锁的进程和加锁的方式等).

4.3　死锁的检测

在SCOP对象管理系统中,死锁的检测要比普通的并发系统复杂,其根本原因在于面向对象系统中锁的层次性.特别是在SCOP对象管理系统中,允许category,class,body,instance 4个层次的锁以及类固有的层次锁.

SCOP对象管理系统的死锁检测方法的原理与一般的死锁检测法是类似的,也是一种环判定算法,不过判定过程中需要考虑锁的层次性.死锁的判定过程如下:当进程A提出的加锁请求不能实现时,就立即进行死锁判定;首先求出该加锁请求因层次关系派生的所有基本锁,在这些基本锁中,必然有一些无法实现,求出阻碍这些基本锁的进程;然后从等待队列中找出这些进程未能实施的加锁请求,再由加锁请求求阻碍这些请求的进程…

如此往复,如果进程A出现在这些进程中,就说明出现了一个环,发生了死锁.这时,server就可以将进程A所加的锁强行解去,避免死锁的发生.

4.4 cache一致性

在SCOP对象管理系统中,每个client进程有一个局部cache,client进程的读写操作在它的局部cache中进行,这就要求解决cache一致性问题.

在SCOP对象管理系统中,cache一致性的实现分为2个阶段:(1)执行写操作的client进程向server发出同样的写请求,使得server的全局cache与其保持一致;(2)server执行写操作后,给其它client进程发"更新消息",指明那些对象是"脏"对象,其它client进程接到"更新消息"后,将"脏"对象从其局部cache中删除,这样就保证了其它client进程的局部cache与server进程的全局cache的一致性.

5 结束语

可以看出OMS与ODBMS有很多相同的地方,但作为软件开发环境的信息存储管理机制,OMS又有其自身的特点.ODBMS管理的对象是大量的对象实例,而作为这些实例的模式的对象类相对较少,所以ODBMS注重的是对象实例的查询优化、模式演变等.而OMS管理的对象不仅有对象实例,还有范畴、对象类、体等,并且OMS管理的数据主要是大量的对象类型,每一类型的对象实例则相对较少.因此OMS的主要任务是维护各种对象之间的复杂关系,并为用户操作对象库提供一个统一的界面,实例查询优化方面不用过于强调,而对象类的查询功能则大大加强.

SCOP环境中的对象管理系统OMS为环境中各种对象数据提供一个统一的操作界面,为各种工具在环境中集成提供一个平台.主要特色如下:

(1)范畴和多体的实现提高了抽象化程度,增强了系统的建模能力;

(2)OMS是一个独立的对象库系统,可自行设计外存对象库;

(3)提供持久对象支持,对象一致性、内外存映一致性的维护以及完整的对象权限管理机制;

(4)考虑到现代软件工程环境多用户、多任务的要求,设计并实现了基于client-server机制的对象管理系统.给出了category,class,body,instance多级粒度加锁规程;

(5)针对死锁、cache一致性提出并实现了相应的解决方案.

目前,SCOP系统已在SUN Solaris2.4上实现.其对象管理系统OMS基本满足SCOP环境提出的对象管理方面的要求.但还有许多地方有待改进,如模式演变是面向对象数据库

研究中的一个重要问题,即在类的定义被修改时,系统如何通过自动修改其实例内容等来维护系统的完整性.在类多、实例少的对象管理系统中,模式演变可以简化为删除其所有实例然后重新加入.但如果以后 OMS 要接受更多的数据对象,实现面向对象数据库的一些功能,可以考虑采取一些较为简明的模式演变策略;错误恢复是实用数据库系统必备的一项功能,它主要处理数据库系统由于软、硬件故障而出现的不一致状态的恢复.目前我们仅对 client 进程异常终止问题作了相应的处理,因此这个方面还有很多工作要做.

参考文献

1 Lewis Ted G. Where is client/server software headed. Computer, April 1995.

2 冯玉琳,黄涛,武小鹏.面向对象的组合软件工程研究.计算机学报,1996,**19**(3):237～241.

3 冯玉琳,黄涛,李京.面向对象的软件构造.软件学报,1996,7(3):129～136.

4 Cattel R G G. Object data management. Addsion-Wesley Publishing Company, 1992.

5 Kim W. Introduction to object-oriented databases. Cambridge: The MIT Press, MA., 1990.

SCOP OBJECT MANAGEMENT SYSTEM

Huang Tao　Chen Jiawan　Feng Yulin

(*Laboratory of Computer Science　Institute of Software　The Chinese Academy of Sciences　Beijing*　100080)

Abstract　OMS(object management system) serves as the kernel part of SCOP component software. This paper outlines the design and implementation of OMS, including memory object management, disk storage management and concurrent object base based on client/server architecture.

Key words　Object instance, body, class, category, object management system, concurrency, lock.