关系数据库与面向对象数据库的集成*

车敦仁 周立柱

(清华大学计算机系 北京 100084)

摘要 本文从RDB(relational database)的缺欠、OODB(object-oriented database)的不足和新的数据库应用需求3个方面指出了统一关系范型和OO范型、集成RDB与OODB的趋势,讨论了集成RDB与OODB的3种途径,最后介绍了一个集成关系与对象模型的多媒体数据库系统TH-CIMHYPER的总体设计.

关键词 面向对象数据库,对象关系型数据库,多媒体数据库.

关系数据库RDB(relational database)经过了30年的发展,已获得了极大成功,无论在理论和技术上,或是在市场和应用上都堪称为主流数据库;面向对象数据库OODB(object-oriented database)在最近10年来受到了学术界特别的青睐,在理论、技术、产品及应用方面均积累了一定的经验.进入90年代,特别是最近几年,人们对于RDB的优势与缺陷和OODB的承诺与不足已获得了比较明确的认识,把RDB与OODB进行集成(Integrate)乃至合一(Unify)已成为一种势在必行的选择[1~3],研究把关系范型和OO范型统一起来的理论与模型是数据库领域里新的研究方向和挑战.[4]

促成对象与关系合一的原因,可从如下3个方面加以概括:

(1)RDB的欠缺:RDB代表着数据库技术的杰出成就,提出和发展了一系列标准的数据库技术[2],如非过程式查询机制、自动查询优化、动态模式更新、事务管理、并发控制以及故障恢复、安全和授权管理等.但是随着数据库应用的日益发展,RDB表现出下列主要缺陷:①不能有效地表示和处理复杂对象(如含有对象引用的层次数据——Hierarchical Data):层次数据通常被表示成多个关系元组,在应用时又不得不通过多关系联接(JOIN)来恢复这些数据,而JOIN是一个开销很大的操作.②不支持用户定义的数据类型(类型系统是封闭的):建模能力差,数据转换的开销大.③不支持对象封装:尽管现今RDB已能够支持存储过程(Stored Procedure),但不支持OO范型所倡导的对象封装,又因为没有继承,故不能自动支持重用.显然,增强RDB的数据类型机制是克服RDB上述缺陷的关键.

(2)OODB的不足:面向对象范型极力倡导的封装(Encapsulation)和继承(Inheritance)概念从技术上赋予了OO范型和OODB 2个基本好处:①强有力的类型定义设施;②减少了

---

* 本文研究得到国家自然科学基金和国家863高科技项目基金资助.作者车敦仁,1964年生,博士后,副教授,主要研究领域为数据与知识工程,软件体系结构.周立柱,1947年生,教授,博士导师,主要研究领域为数据库,软件工程.

本文通讯联系人:周立柱,北京100084,清华大学计算机系

本文1995-10-11收到修改稿

开发和维护复杂软件系统的难度,从而使OO方法较之其它方法在软件的生产率(Productivity)、重用性(Reusabitity)和软件质量上(Quality)上作出了更大的承诺(Promise).但是OODB目前仍未在市场以及关键任务应用(Mission-Critical Applications)方面被广泛接受,其主要的原因包括以下2个方面:①OODB作为一个DBS还是不太成熟,如缺少完全非过程性的查询语言以及视图、授权、动态模式更新和参数化性能调协(Parameterized Performance Tuning)等;②OODB与RDB之间缺少应有的兼容性,因而使得大量的已经建立了庞大的RDB应用的客户不敢轻易地再去选择OODB.

(3)新的DB应用需求:RDB的性能优势在于对海量(简单)数据的检索,OODB的性能优势在于对复杂数据对象的导航式访问,尤其是当采用了物理对象标识(OID)或指针混洗(Pointer Swizzling[1])技术时(指针混洗是指在数据加载时自动将磁盘指针转换为虚存指针).象CAD/CAM,CASE这些所谓的非标准应用往往既要求检索访问又要求导航访问,即同时要求体现RDB与OODB的性能优势.

总之,传统RDB仍要求继续发展(如ORACLE,INGRES,SYBASE,ANSI SQL3等都在向OO方向扩充),OODB也要求进一步成熟并与在市场和应用中占有霸主地位的RDB保持尽可能兼容,于是就导致了关系范型和OO范型合一这样一个必然趋势.INGRES,UniSQL/x是目前最能够体现具有"对象-关系"合一范型特点的典型代表.Won Kim博士甚至直接把自己的UniSQL/x称为对象关系型数据库(Object-Relational Database).[5]

本文第1节先介绍集成RDB与OODB的3种基本途径,第2节介绍一个已经开发完成的具有"关系-对象"合一范型的、面向CIMS工程的、多媒体数据库系统TH-CIMHYPER的总体设计,最后对全文进行小结.

1 RDB与OODB的集成

目前对RDB与OODB的集成可以概括成3种不同的途径:Gateway方法、OO-layer方法和合一(Unification)方法.

1.1 Gateway方法

Gateway方法是在OODB应用与RDB服务器之间增加一个能对二者起到"沟通"作用的中间环节(见图1).该方法的特点是:Gateway就象是RDB的一个普通用户;OODB需求受限过多(面向对象特色难于充分体现);性能欠佳——由于对每次服务请求和结果都要进行必要的翻译;可用性差——用户必须了解2种不同的DB.

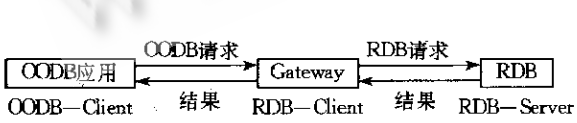

图1 Gateway方法

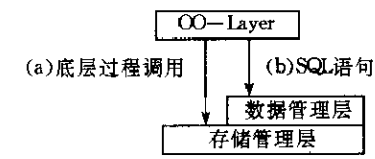

图2 OO—layer方法

1.2 OO-layer方法

OO-layer方法是在一个现成的RDB引擎(Engine)上增加一层"包装",使之在形式上表现为一个OODB(见图2).该方法的特点是:OO-layer与RDB引擎之间有3种接口方式

(见图 2)：a；或 b；或(a 且 b)；翻译的开销仍很高；在实现上，一般不为适应上层 OO-layer 的需要而对下层的 RDB 引擎作相应的修改；由于关系模型和对象模型的独立存在，当涉及复杂的数据库设施时，会导致严重的性能和操作问题，如模式演进要求对类层次进行的原子性封锁，RDB 原本是不支持的；OO-layer 方法可作为多数据库系统的一种实现基础.

1.3 合一(Unification)方法

合一方法是指在模型和系统上把 RDB 与 OODB 集成为一体，使之同时具备 OO 范型和关系范型的特色(见图 3). 对模型上的合一将导致把下列概念等同起来：关系与类、关系的元组与类之实例、表列(Column)与属性(Attribute)、过程与方法、关系继承与类继承(关系继承是对 RDM 的新扩充)等. 该方法的特点是：在实现上一般要对 RDB 的存储层和管理层作必要的修改；数据模型一致化(Unified)；查询与操纵一致化；支持数据库语言所允许的各种机制，如动态模式更新、自动查询处理和优化、并发控制、恢复、事务管理、授权等；实现比较困难.

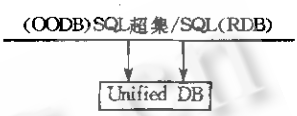

图3 合一方法

1.4 小　结

这 3 种方法的性能和支持 OO 特征的充分程度依次增加，但实现的难度也依次增加.

2 TH-CIMHYPER 系统的总体设计

TH-CIMHYPER 是一个在微机网络环境下已初步开发完成的、面向 CIMS 工程特点的、具有关系与对象的集成模型的多媒体数据库系统. 系统的主要功能包括：

(1)在数据模型上支持对象模型和关系模型的集成模型，在当前实现中以支持 OO 模型为主.

(2)能够对包括声、图、文等的多媒体对象和常规 RDBMS 所支持的字符、数字等基本数据对象进行有效且一体化的存储和管理支持.

(3)具有 Hypertext/Hypermedia 系统的基本功能.

(4)支持 2 种类型的用户界面：①支持复杂应用开发的 C++程序设计环境；②提供一个功能丰富的交互式用户界面，该交互式界面能够集中体现面向对象数据库和面向对象 Hypertext/Hypermedia 系统的典型功能. 提供用户交互式访问数据库和采集多媒体(音、视频)对象的手段，并能够建立和维护对象间的超媒体信息网络.

2.1 系统的构成

TH-CIMHYPER 系统的构成可划分为如图 4 所示的 4 层：

(1)第 1 层(level 1)

底层(level1)是系统的存储管理层，在本系统中我们选用了具有关系模型特点的 CodeBase 作为存储子系统的实现基础. TH-CIMHYPER 的存储子系统对 CodeBase 所做的最本质的扩充是其多媒体存储管理部分.[6]

(2)第 2 层(level 2)

(1)按 RDB 的基本要求对 CodeBase 进行基本扩充，比如扩充索引和数据字典等，形成一个能够反映关系数据模型 RDM 的基本特点的所谓关系视图 R-Vision. (2)在 R-Vision 的基础上，形成一个与 R-Vision 地位平等的能够反映对象数据模型 ODM(object data mod-

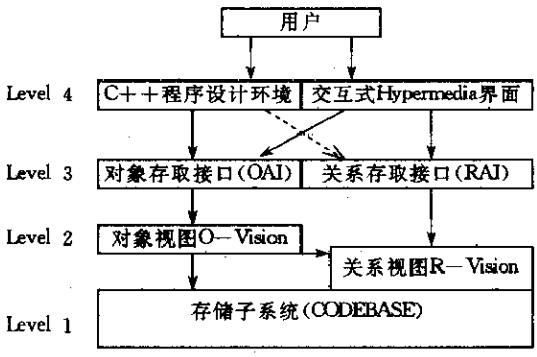

图4 系统的构成

el)主要特点(如对象标识OID、对象方法和类的继承等)的所谓对象视图O-Vision. O-Vision的基本内容包括类定义字典、类型定义字典、属性字典和方法字典等.

R-Vision和O-Vision实际上都对应着RDBMS的常规三级模式结构中的第2级,即属于概念模式或逻辑模式. R-Vision与O-Vision不是2种不同模式的简单并置,而是2种不同模式的集成,应该具有"联动"效应,它们是对相同的数据对象的2种不同视图(即从不同角度对同一对象的观察).

我们试图从R-Vision与O-Vision的集成中获得RDB与ODB的高度统一.

〈3〉第3层(level 3)

在第2层提供支持的基础上,建立对数据库的2种存取接口:对象存取接口OAI(object access interface)和关系存取接口RAI(relation access interface).

(1)OAI应向C++提供一组基本库函数(实际上具有部分DML的作用,这些库函数允许嵌入到C++中去使用,使C++成为TH-CIMHYPER的一个计算完全的DBPL,也使C++成为一个持久性语言). 目前的OAI函数集包括下列4类函数:

Put类函数:对应传统数据库系统的检入(Check-in)功能;

Get类函数:对应传统数据库系统的检出(Check-out)功能;

Find类函数:对应传统数据库系统的查询功能;

辅助类函数:用于辅助以上函数的应用.

(2)RAI提供了实现RDB基本库操作的一组函数,如CreateTable(),DestroyRelation(),CreateIndex(),InsertTuple(),DeleteTuple(),FindTuple(),UpdateTuple(),PrintTuple()等.

〈4〉第4层(level 4)

第4层面向用户提供2种形式的基本接口,一个是支持复杂应用开发的C++程序设计环境,另一个是功能丰富、方便易用、基于MS-Windows的交互式用户界面,其中包含一个交互式Hypertext/Hypermedia子系统.

2.2 模式管理

〈1〉模式的定义

数据库模式是对数据库的描述,传统RDB的三级模式概念是从3种不同的抽象级上对数据库的描述:内模式(系统程序员视图)、概念模式(DBA视图)和外模式(子模式,用户视图). 模式的内容由一组元数据(Meta-Data)项组成,包括关系定义、属性定义等.

模式也可以看成是从不同的角度,用不同的观点看待或解释同一个数据库的方法,这正是本系统中模式的特点.

本系统中的模式分为3种:从OO范型看到的是数据库的对象模式O-Vision;从关系范型看到的是数据库的关系模式R-Vision;从程序语言C++看到的则是由C++的类型子

系统所描述的模式(即一组C++的".H"文件).它们之间的差别仅在于从不同的角度看待和使用相同的数据库.它们之间是有冗余的,但其冗余却是为不同的用途而存在的.对象模式O-Vision支持用户按OO范型存取数据库;关系模式R-Vision支持用户按传统关系范型访问数据库;而把模式表示为C++的头文件形式的用意则是为了方便地实现数据库与C++语言的无缝性集成,把库模式中对象及类的定义直接纳入C++应用程序的数据定义部分,使得持久对象一旦被从库中检出便可直接进入C++应用程序运行空间,接受同样的操作,操作结束后还可被再次检入库中.

〈2〉三种模式之间的关系

(1)关系视图R-Vision直接建立在CodeBase之上,与关系存取接口RAI能够共同实现普通RDBMS的基本功能.R-Vision虽不直接依赖于O-Vision和C++的".H"文件,但却需要与之保持一致性.为了简化实现工作,本系统暂不拟支持对R-Vision的直接交互式操作.

(2)对象视图O-Vision是建立在R-Vision之上的符合OO范型的数据库视图.O-Vision与R-Vision没有直接的对应的关系,O-Vision的实现要部分地以R-Vision为基础.

(3)C++的".H"文件,就其内容而言,它是对O-Vision的一种冗余表示,是C++程序员看到的数据库结构,严格来讲它不是真正的"数据库模式",而只是库模式的一种C++表示.O-Vision中的每个类或类型都必须在应用程序的C++".H"文件中有一个相应的类或类型的定义声明.这是把数据库的持久性机制与C++应用程序联系起来的一种简便方法.从而也把C++变成一种持久性的程序设计语言.

应用程序中的类/类型并不是都需要持久性,这也是持久性正交性原则[7]的基本要求.因而C++应用程序的".H"文件中类/类型实际上可被分为2种,一种要求持久性,另一种不要求持久性.要求持久性的类/类型与O-Vision中的类/类型应具有一一对应的关系.本系统默认C++现有类库中的类/类型均为持久性的类/类型,用户定义的类/类型是否需要持久应由用户在定义时给予标记,如约定在持久性类/类型的定义之前冠以"db"标记.

〈3〉模式的建立

3种模式之间的一致性是需要时刻加以维护的(即所谓"联动"的).在本系统中我们禁止对O-Vision和R-Vision直接操作,以便简化3种模式间一致性的维护工作.本系统只允许通过对C++".H"文件的编辑修改来完成对库模式的定义或修改.对作为库模式的C++头文件进行修改后,需经模式捕获器SC再次进行预处理.

建立库模式的具体过程如下:

(1)用户象编写普通的C++程序一样,利用编辑方式把所有的数据定义建立成一个或多个C++".H"文件.把需要持久的类/类型前面打上标记"db".

(2)这一组C++".H"文件需要经过模式捕获器SC(schema captor)的预处理,把所有捕获到的模式信息写进O-Vision中相应的数据字典,进而也写进R-Vision相应的数据字典.SC有下列3个基本作用:①从O-Vision的角度对C++".H"文件进行语法错误检查,比如检查是否有重定义错误、模式的定义是否完全等;②从数据库管理(即实现OAI)的角度对C++".H"头文件进行必要的预处理,如把系统的管理类"Administrate"(该类中封装有OAI中的全部函数)指定为所有持久性类的友员(friend)类.③把从C++".H"文件捕获

到的全部模式信息,填写到O-Vision的各类字典中,进而传播到R-Vision的相应的数据字典中.

3 结束语

在数据库环境(Context)下统一关系范型和对象范型,集成RDB与OODB是数据库发展的一个重要方向.从系统实现的角度,有下列几个关键技术问题必须加以解决:

(1)对象表示的规范化:OO数据模型不存在象关系模型那样的范式,对象粒度的离散性极大,因此统一这2种模型的第1个问题就是采用一定的策略把同一个类中本来不定长的每个对象(尤其是包含多媒体成分的对象)规范化.在TH-CIMHYPER中,是通过把不定长的对象转换成一个定长的对象和若干个不定长的(下层)隶属子对象.让定长对象对应R-Vision中的一个普通元组,其中含有一些特殊的外键(Foreign Key),这些特殊外键本身是定长的,但指向的却是某些堆文件空间中的可能不定长的区段.在TH-CIMHYPER中,我们把不定长的数据(特别是多媒体数据)在存储组织上按堆的方式来组织,让每种不定长的数据类型对应一个堆文件,如"image"堆、"audio"堆等.

(2)对象聚集:聚集指的是根据某种原则把相关对象在磁盘上进行相邻存放,以便减少I/O的次数,提高系统的性能.在RDB中,元组是按关系(表)来聚集的;而在OODB中,一般存在2种基本的聚集策略:一种是按类聚集,另一种是按引用关系IPO (is-part-of)聚集,对于每个对象实体来说2种聚集策略只能选择其一.在本系统中,对象按类聚集于一个文件中.IPO聚集与关系模型较难相容,故放弃之.

(3)查询模型:作为OO范型的核心概念的继承和封装以及被称之为预定义联接(Predefined Join)的对象引用(或导航),恰又是对象模型根本不同于关系模型的地方.在对象与关系合一的数据库中,应该有一个统一的、概念上一致的查询模型.这样的查询模型目前在研究界还是一个开式的问题(Open Probolem).但解决这一问题的出路似已比较明了,那就是有所保留地对SQL进行OO扩充,如引进关系(表)间的继承.SQL3工作报告和UniSQL/x也正是这么做的.在TH-CIMHYPER的研制中,我们对这一问题基本采取了回避的作法.数据库查询是通过一些预定义的库操作函数来实现的,没有提供统一的查询模型,因而所能进行的查询优化非常有限.此外,系统底层选用了CodeBase是因为CodeBase扩充性较好,易于把OO特征扩充进去.如要追求较高的系统性能,在将来产品化时可把CodeBase弃之,重新开发底层存储管理部分.

TH-CIMHYPER实现中的一个显著的技术特点是,采用了对象与关系的合一范型后,给Hypermedia子系统的实现带来了方便,也赋予了系统更大的性能潜力.因为Hypermedia中的语义网是典型的"对象关系网",对象间的语义链(Link)用关系(表)实现既直观又方便.

TH-CIMHYPER系统目前虽已开发完成,但对某些研究性问题,如对象关系型数据库的更合理的体系结构、存储结构、查询模型、查询优化、封锁技术等,我们还将结合新的项目进行更多的探讨.

参考文献

1 Ananthanarayanan R *et al*. Using the co-existence approach to achieve combined functionality of object-oriented and

relational systems. ACM SIGMOD, 1993,**22**(2):109～118.

2 Won Kim. Object-oriented database systems: promises, reality and future. In: Agrawal R *et al.* eds. VLDB'93, USA: Morgan Kaufman Publishers, Inc., 1993. 676～687.

3 Alfons Kemper, Guido Moerkotte. Object-oriented database management-applications in engineering and computer science. USA: Prentice-Hall Inter. Inc., 1994.

4 车敦仁,周立柱. 概念代数——新一代数据库系统的理论. 计算机研究与发展,1996,**33**(1):32～38.

5 Won Kim. Object-relational database systems. In: Proc. of 4th International Conference on Database Systems for Advanced Applications, Singapore, 1995.

6 熊胜峰,周立柱. CIMS 多媒体数据库的存储管理技术. 计算机集成制造系统—CIMS,1995,**1**(2):29～32.

7 车敦仁. 面向对象的智能数据库系统:研究、设计与实现[博士论文]. 北京:北京航空航天大学,1994.

INTEGRATE RELATIONAL DATABASE AND OBJECT-ORIENTED DATABASE

Che Dunren　Zhou Lizhu

(*Department of Computer Science　Tsinghua University　Beijing*　100084)

Abstract　Due to limitations of RDBs (relational database), deficiencies of OODBs (object-oriented database) and requirements of new applications, a new trend in database area is becoming more and more obvious that relational paradigm and OO paradigm are to be unified in the database context, and RDB and OODB are to be integrated transparently. This paper first discusses three approaches for integration of RDB and OODB, and then introduces the design of a multimedia database system TH-CIMHYPER in which the authors adopt the unification approach.

Key words　Object-oriented database, object-relational database, multimedia database.