ORDBS 对象组装 对象导航 数据库

43-46,74

计算机科学1996Vol.23№5

ORDBS中复杂对象的组装与导航*)

TP 311.13

马秀莉 李天柱 黄 炜 任建利

(河北大学计算中心 保定071002)

摘 要 In this paper, we will discuss the frame and value separate structure of objects in buffer, and lazy and partial assembly technology adopted in our prototype system O-RBASE. This structure and technology can reduce the assembly of those unnecessary information, so can significantly improve the efficiency of navigation and query.

关键词 Object-relational database, Object buffer, Type-value separation, Object assembly, Navigation.

下一代数据库的特点之一是结合关系模型和O-O模型的优点,我们研制的原型系统*O-RBASE*着眼于NF^2关系模型与O-O模型相结合,扩展NF^2模型成核心O-O模型,改造RDB的存储管理和数据管理层,且考虑与关系模型的兼容性,称之为对象-关系模型。复杂对象数据库研究中的一个重要课题是对复杂对象查询的高效支持。这种查询包含两个方面:谓词计算和构造查询结果。这两者都涉及从复杂对象至其成分对象的导航问题。目前,关于可支持谓词计算的索引方案的研究有不少,但对于支持高效导航的结构研究还很少。虽有一些文章讨论了加快整对象的组装速度[5],但极少文献考虑减少冗余信息的组装。文[6]虽给出一种称为对象骨架的导航结构,但把所有对象的骨架都生成并存入磁盘,这种静态的构造方法的缺陷是物理OID的实现方法十分不利于高阶属性值的更新(即增、减、替换成分对象),页面指针和偏移量的维护十分繁琐。本文提出一种可减少I/O,支持高效导航的型值分离对象缓冲结构。该方案已在原型系统O-RBASE中实现,效果十分显著。

1 对象-关系模型

O-RBASE中的原子对象就是其值本身,元组型对象(嵌套的复杂对象)具有对象标识sys-id,sys-id是系统生成和管理的实体标识,对象的概念标识[1]大致为sys-id+c-id(c-id为对象所在精确类的类标识),与型有关。任何时候,对象的状态都通过其属性的值来定义。零价属性具有原子值,域是原子类型;高阶属性的域是类。若两个类C和C',其中C'是C中一属性的域,则在C和C'之间建立了一种嵌套结构,称为复杂对象嵌套结构,该结构中类C'的属性称为C的嵌套属性。高阶属性的值为对象集,若对象A是对象B的某高阶属性的值—对象集中之一,则称A为B的成分对象,而将B称为A的宿主对象或复杂对象。O-RBASE采用常规存储模式(normalized storage model),类中各嵌套层之间由连接索引(Join Index)连接,由相关类的sys-id构成。在O-RBASE的模式中,也可按子类与超类间的ISA关系把类组织成继承类层次。O-RBASE采用纵向分割存储模式[2,3]。每个子类中的对象在其特有属性集下的值及该类的所有子类中所有对象在该属性集下的值聚集存储在一个文件里,此文件对应的类叫S-子类[4],所以一个整对象的状态就按继承类层次分存在不同的S-子类文件里,对应的各记录称为S-子记录。一个整对象的所有S-子记录由该对象的sys-id联系起来。对象组装的目的是把复杂对象由外存格式转变成一个可快速导航的内存格式,所以首先是把各S-子记录按ISA关系链接成一个顶层整对象,其次是把内层成分对象按照嵌套结构与顶

*)本课题由国家自然科学基金和河北省自然科学基金资助

层链接成一个复杂对象，对复杂对象的内嵌属性的访问，中间要经过一个从复杂对象到该属性所在的成分对象的导航过程。

图1是O-RBASE模式的一个例子，图中虚线代表嵌套关系，实线代表继承关系。

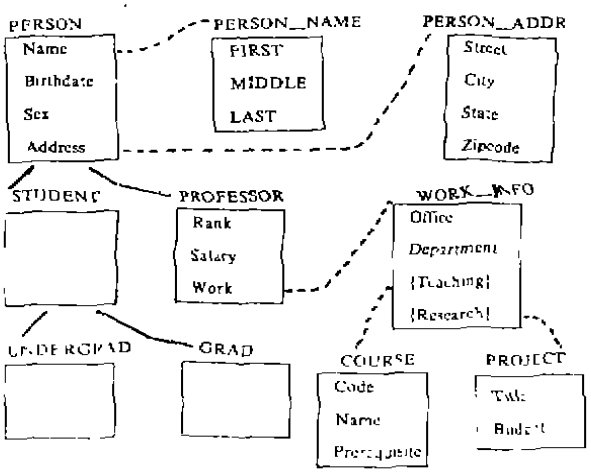

图1 嵌套结构和继承类层次

2 型值分离对象缓冲结构

对象缓冲是系统缓冲与对象操作之间的缓冲，完成对象的组装和拆卸，对系统效率有较大影响，减少I/O和不必要的操作，提高查询效率，是O-RBASE缓冲设计的主要指导思想之一。若对图1所示数据库进行如下查询：

```
SELECT Name, Work. Teaching. Name, Work. Research. Budget
FROM PROFESSOR
WHERE PROFESSOR. Work. Teaching. Prerequisite = 'OODB'
```

谓词中涉及的属性在复杂结构中嵌套很深，且查询表达式一边为常量，为此，可在嵌套属性Prerequisite上建立索引，快速定位满足条件的对象[5]。但是，若查询条件发生改变，

```
WHERE PROFESSOR. Work. Teaching. Name = PROFESSOR. Work. Research. Title
```

此时，就难以使用索引来进行谓词计算，需用导航办法取得内嵌属性Teaching. Name和Research. Title，然后计算谓词。对于查询结果中的内嵌属性，也需经导航取出。在O-RBASE中，复杂对象通过sys-id引用它的成分对象。虽然sys-id提供了沿嵌套结构导航的一种机制，但这种机制效率很低，尤其是当谓词或查询结果中涉及的属性在复杂对象的嵌套结构中嵌套较深时，沿复杂结构导航至这些属性所在的成分对象可能需做许多次磁盘操作，所以在O-RBASE中需有一种新的存取结构来支持高效导航，从而使查询效率得到真正的提高。

我们还应注意到，该查询的结果只需某些层上属性，而非所有层上属性值，导航路径上各中间结点只起导航作用，结果中不需要中间结点对象的属性值，由于必要的属性值只是整个复杂对象状态的一小部分，若把路径上涉及到的每一个对象的所有属性值都组装到内存，就有大量冗余信息，影响组装速度与导航效率。

下面简单介绍一下本系统的缓冲区管理，然后详细描述型值分离结构。

2.1 缓冲区管理

O-RBASE设有对象缓冲区，对象缓冲中采用型值分离结构来存放组装好的对象，并通过ROT(Resident Objects Table)来管理这些对象。ROT为驻内存对象的sys-id与型值分离结构(的内存地址)的映照表。当访问一对象时，首先要在对象缓冲区中组装该对象，当对象缓冲区满时，要按一定的策略拆卸一部分对象，此后若再访问已拆卸的对象，需重新组装。

对象缓冲区是按对象的嵌套层次划分的，对象的每一层有自己的局部缓冲区，但同一类层次的类中的对象使用同一组局部缓冲区。

型值分离结构分为两部分：骨架和值表，与此相应，每个对象缓冲区也分为两部分：TB(Type Buffer)和VB(Value Buffer)。TB中存放骨架，VB中存放值表。

2.2 型值分离结构

骨架是该结构的核心，所以主要讲骨架的结构。

O-RBASE的纵向分割存贮模式决定了顶层整对象的骨架是由与S-子类一一对应的子骨架通过ISA关系链接而成的，O-RBASE中约定每个类至多有三个直接超类，子骨架的结构如下：

suplink1	suplink2	suplink3	s-record	h1first	h1last	···	hnfirst	hnlast

suplink1,suplink2,suplink3:分别为与该 S-子类的三个直接超类对应的子骨架的内存地址。

s-record:根据该层 S-子记录是否已取至 VB 而定,在对象刚被组装时,S-子记录尚未取入内存,该域为空值;当数据库操作第一次涉及到对象在该 S-子类的零阶属性的值时,该层 S-子记录就从外存调入与 TB 对应的 VB 中,并把该 S-子记录的内存地址存入 s-record 域。

其后每两个域对应一个多值高阶属性(若为单值高阶属性,分配一个域),把高阶属性的值(对象集)组织成一链表,在 hifirst 域中存放该链链头地址,在 hilast 域中存放链尾地址,而该链表中每一结点的结构如下:

sys-id	next	parent

sys-id:存放该成分对象的对象标识,在适当时机指针转换会把该 sys-id 转换为指向该成分对象的描述器的指针,而描述器又指向该成分对象。

next:指向该链表中下一结点的指针。

parent:指向该高阶属性所在的子骨架的指针。复杂对象与成分对象间为双向链。

可见,我们已把顶层各子骨架通过 ISA 关系联系起来,又把顶层和内层嵌套的各成分对象通过嵌套关系联系起来,形成了一个复杂对象的骨架,对象骨架其实构成了复杂对象中各成分对象的 sys-id 网络。

假设该复杂对象的一部分状态信息(S-子记录)已调入 VB,我们把这些 S-子记录称为值表,把一复杂对象的这种内存形式(骨架+值表)称为该对象的型值分离结构。

2.3 指针转换

指针转换技术主要是为解决对驻内存的永久对象的访问。对象的型值分离结构实质上是一个 sys-id 网络,为加速沿骨架的导航,可把对象骨架的高阶属性的成分对象链中的 sys-id 转变成成分对象的骨架的内存地址形式的指针,从而避免通过查 ROT 表来间接定位一个已驻内存的永久对象,把复杂对象间的导航转变为内存指针的跟踪。

在 O-RBASE 中采用间接转换,同时也是延迟转换,为每个被引用对象设立一个描述器(descriptor)[7],sys-id 被转换成 descriptor 内存地址。采用 descriptor 后,实际上把原来的 ROT 表转变为"sys-id 与描述器内存地址映照表"。对象缓冲区中管理对象是通过 ROT 管理描述器来实现的。每一缓冲区都有一个当前对象,系统可根据用户需要改变当前类,访问当前类中的对象时取出的成分对象,在改变当前类后,可以变成新当前类中的当前对象,这又可减少 I/O 次数。

把指针转换融入型值分离结构中,可进一步加快导航的速度。

3 部分组装与高效导航

对象组装的工作就是在缓冲区中形成复杂对象的内存格式。首先根据该对象的概念标识(sys-id+c-id,其中 c-id 为对象所在精确类的类标识)和数据字典中模式信息,在相应 TB 中,生成该对象顶层骨架,包括所有超类上的子骨架,以后访问涉及到其 S-子类属性时,扩展相应部分。

首次组装时不生成骨架中的成分对象链,而是在以后对高阶属性进行第一次访问时生成相应的链。在导航过程中,当路径中涉及到高阶属性时,则依数据字典和连接索引生成该高阶属性的成分对象链;若被引用对象尚未组装,则如前由 sys-id 组装出相应顶层骨架(应在该高阶属性的域所属类层次对应的 TB 中,可能与引用对象的 TB 不同,除非有域递归问题),并把复杂对象骨架与成分对象骨架间的联系进行指针转换,而其它未用到的成分对象目前并不组装,这种组装应称为骨架的延迟组装,其实就是需要访问哪个高阶属性,就生成 sys-id 网络中的相应分支,当针对某个特定属性处理一批对象时,这种方案可大大减少对冗余信息的组装。随着对已在缓冲区的对象的访问的积累,复杂对象嵌套的各内层成分对象逐渐被组装至对象缓冲区,sys-id 网络渐趋完整。假设一复杂对象的骨架已形成最完整的 sys-id 网络,此时就可沿骨架在该复杂对象的嵌套结构上自由导航,而且在导航的过程中,无须任何外存操作。

至此,假设访问不涉及零阶属性值,如查 professor,work,teaching,则所有零阶属性值都尚未取至内存,而高阶属性值在骨架中也只是 sys-id 链。这其实是实现了本方案的一个核心思想:型值分离。

即：只组装对象的模式骨架，不组装对象的状态。由于导航无须知道路径上各对象的状态，骨架对于导航来说已足够，而骨架的生成只用到数据字典和连接索引，且二者均很小，从而也极大限度地减少了导航的1/O次数。

当导航最终定位到一零阶属性时，只需从外存把该属性所在S-子记录调入相应VB中，整个复杂对象的其它S-子记录均不必调入内存，这其实是实现了值的延迟组装，就是访问哪个零阶属性，就把它所在的S-子记录调入内存。骨架的延迟组装和值的延迟组装是本文部分组装的两个方面。

对嵌套的复杂对象，O-RBASE的存储模式为规范模式[9]，即每个类中的所有对象的顶层存在同一文件中，并具有对象标识，对象顶层与内层高阶属性值，即与高阶属性域中对象联系由连接索引实现，也就是说，数据聚集是按型，非按复杂对象本身，一个复杂对象的不同层散落在不同文件中，因此将一复杂对象读入或卸出（内存）就需组装和拆卸，这在对象缓冲区中完成。对类层次，采用纵向分割存储模式，支持以扩展方式生成对象骨架，这些都是对缓冲管理中只装配"有用"属性值的高效支持的基础。

4 小结

对象缓冲管理（对象的组装和拆卸）对查询效率有较大影响。在O-RBASE中，对象在缓冲区的组装用了型值分离结构和延迟部分组装技术。对任一带路径的访问A.B.C.…f，对非叶结点和非零阶属性的叶结点，只组装骨架；若有零阶属性，只会出现在叶结点，也只组装这个零阶属性值；这是对导航的高效支持。据此，任何时候，只组装有用的信息（骨架和值表）。有用信息就是谓词和查询结果中出现的高阶和零阶属性。因此，查询中不组装任何"无用"信息，大大减少了1/O，提高了查询效率。

骨架和值表是各自独立增长的，也就是说，组装的信息可以积累重用，这进一步减少了1/O。

特别，只凭数据字典中模式信息和连接索引就可逐步组装起完整的对象骨架，借此，由当前对象可导航至其任何成分对象，而无需取任何零阶属性值，此时的1/O极少。

（下转第74页）

Ⅰ,What(Ⅰ)→How(Ⅰ) (7)

利用关系式(5)代换 W_1,W_2,并在一个 QFD 表内表达二交互任务Ⅰ,Ⅱ,称作交互 QFD(I-QFD),如图 8 所示。

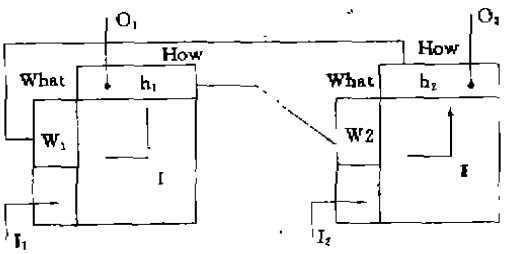

图 7 交互任务的 QFD 模型

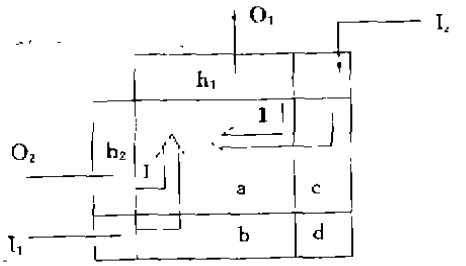

图 8 交互 QFD(I-QFD)

3.5 I-QFD 过程

I-QFD 是一个反复迭代,逐步完善的过程,可形式化表达如下:

Step 1:初启:$I_1\rightarrow h_1^0$;$I_2\rightarrow h_2^0$

Step 2:交互 $I_1\cup h_2^j\rightarrow h_1^{i+1}$,$i=0,1,2,\cdots$;$I_2\cup h_1^i\rightarrow h_2^j$,$j=1,2,3\cdots$

Step 3:稳定验证:

If $h_1^{i-1}\cup h_1^i=h_1^{i-1}\cap h_1^i$;$h_2^{j-1}\cup h_2^j=h_2^{j-1}\cap h_2^j$;$R_{m,n}^{i-1}=R_{m,n}^i\forall\ m,n\in R_{ij}$;$C_{m,n}^{j-1}=C_{m,n}^j\forall\ m,n\in C_{ij}$($R_{ij}$,$C_{ij}$分别是关系矩阵和自相关矩阵)

Then stop

Else go to step 2

最后,对稳定后的 I-QFD 表输出项 h_1,h_2 赋值,赋值时参考关联矩阵、自相关矩阵及可能得到的竞争评价值及对应实测值(图 7,图 8 中未画出)而定。

I-QFD 使 QFD 的结构表达扩展到动态交互活动,增强了 QFD 的建模能力,实现了纵向/横向映射的集成,解决了用 QFD 作基元实现 MDIMS 的主要难题。

结论 本文中提出的 MDIMS 系统作为规划和实施 CIMS 的体系结构有以下的特点:

1. 在体系结构中明确定义了人和组织结构。

2. 定义了面向技术——经济总体优化的纵向映射,并实现了纵向映射与横向映射的交互。MDIMS 结构化地表达了信息,技术,人和组织,技术——经济优化的全面集成,符合最新的 CIMS 哲理[2]。

3. 提出了实现 MDIMS 的系统化方法。它有一系列突出的优点:a)采用统一的 QFD 构造基元实现整个体系结构,使 MDIMS 有很强的一致性,便于实现各类映射的集成。b)QFD 是一种并行工程多功能小组的概念工作平台[4,5,6],用 QFD 构造 MDIMS,使得在 CIMS 的规划和实施过程中采用了并行工程的先进工作模式。c)QFD 表格表达方式有很好的直观可视性和相关项的可追溯性,因此有极强的可用性,容易为我国现阶段一般工程技术人员所接受和逐步完善,这对于在我国推广 CIMS 十分有利。d)容易将 QFD 模型计算机化,可以方便地将 QFD 表转换为关系数据库,还可用专家系统来实现 QFD 的知识获取及推理机制。

因此,基于 QFD 基元的 MDIMS 是一种有前途的 CIMS 体系结构,对 MDIMS 的实验研究正在进行之中。(参考文献共 11 篇略)

(上接第 46 页)

参考文献

[1]Li Tianzhu, Object Identity In Database Systems, J. of Comp, Scie. & Tech., No. 4, 1995

[2]李天柱、肖计田,强 isa 关联和弱 isa 关联,第十一届全国数据库会议,西安,1993

[3]王珊等 DBMS/IDKE 中对象存储/存取子系统的若干关键问题,计算机科学,No. 2, 1992

[4]李天柱,NF^2 关系模型及属性继承,软件学报,No. 4, 1995

[5]D. Maier, Development of an Object-Oriented DBMS, In Proc. 1st Intl. Conf. on Object-Oriented Programming Systems, Portland, Oregon, Oct., 1986

[6]Kien A. Hua 等,Object Skeletons: An Efficient Navigation Structure for Object-Oriented Database Systems, In Proc. of the 10th Intl. Conf. on Data Eng.

[7]Alfons Kemper 等,Object-Oriented Database Management, chapter Pointer Switching, Prentice-Hall, 1994

[8]Tom Keller 等, Efficient Assembly of Complex Objects, Proc. ACM SIGMOD Conf., Denver, CO, 1991

[9]Valduriez P. 等, Inplementation Techniques of Complex Objects, VLDB, 1986