ISSN 1000-9825 Journal of Software 软件学报 2000,11(10):1412~1419

支持OO查询物理优化的二维字典签名方法*

吴燕萍 施润身

(上海铁道大学计算机科学技术系 上海 200331)

E-mail: yuanxy@guomai.sh.cn

摘要 提出支持面向对象查询物理优化的二维字典签名方法.首先提出其基本设计思想,在定义二维字典及其数据结构之后,给出了二维字典签名的构造算法及其查询操作算法,最后构造了存储开销和查询代价模型.

关键词 二维层次,二维字典,二维字典签名,代价模型.

中图法分类号 TP311

查询优化是面向对象数据库系统(object-oriented database system,简称OODBS)的重要设计目标之一,它涉及逻辑优化和物理优化两个方面.物理优化主要是确定如何实现查询处理的逻辑方案,这与数据库的物理组织有着密切的关系.目前,这方面的研究主要集中于索引技术和签名方法.索引是数据库中实现快速查询的一项重要技术,OODBS中主要有3类新的索引,即继承类层次索引、嵌套属性索引和二维索引.目前的研究多数只考虑在聚集层次或继承层次上建索引,而且侧重于对聚集层次上索引技术的探讨.路径字典索引[1]就是其中一种有效的索引机制,它可提供沿聚集层次的高效遍历.近年来将早期支持文本检索的签名方法用于支持面向对象(object-oriented,简称OO)的查询也是面向对象查询物理优化的一个新的研究方向.如文献[2]中提出了树签名方法和路径签名方法,它们可对检索整个聚集层次或某一路径上的对象提供有效的支持.

从现有的文献来看,对OODBS中索引技术、签名方法的研究工作尚处于不成熟阶段,尤其缺乏对包含聚集层次和继承层次两个正交层次信息的OO查询的支持机制,本文将对这方面进行探讨.

1 基本设计思想

首先考察路径字典索引方法.它包括路径字典、属性索引和标识索引3个组成部分.路径字典是从数据库中抽出复杂属性表示沿一条路径的类中对象之间的嵌套关系,能提供沿聚集层次的高效遍历.在理想情况下,路径字典中不存在冗余的对象标识符,它能以较小的存储开销体现出同一路径上对象之间的共享调用关系.但路径字典索引方法也存在一些缺点,如受存储空间的限制,它只能对查询频率较高的少数属性建立属性索引,而且路径字典仅包含对象标识符(object identifier,简称OID),一般很难与谓词直接进行比较.此外,路径字典没有反映继承层次上的信息.

再考察签名方法.由于签名是直接存储或嵌套在对象中的信息的一种抽象,所以它的突出优点在于能对各种OO查询提供具有通用性、直接性的支持,包含的对象信息量大而且具有良好的过滤能力.

在考察路径字典索引和签名方法的基础上,本文提出新机制的基本设计思想如下:

(1) 要求是一种辅助文件机制,无需改变数据库的原有结构;

(2) 支持对继承层次上信息的查询;

(3) 能对各种OO查询提供有效的支持,具有一定的通用性;

(4) 可快速扫描路径字典及数据库;

* 本文研究得到铁道部科技研究开发基金(No.J92X014)资助.作者吴燕萍,女,1972年生,助教,主要研究领域为面向对象数据库,信息系统.施润身,1946年生,副教授,主要研究领域为数据库,信息系统.

本文通讯联系人:吴燕萍,上海 200331,上海市真南路500号同济大学沪西校区计算机系基础教研室

本文1998-11-17收到原稿,1999-06-21收到修改稿

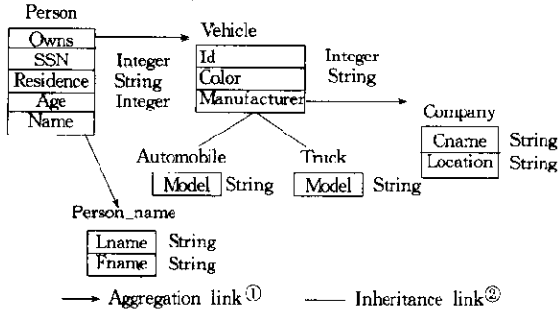
(5) 存储开销较小.

签名方法的存储开销和维护代价与签名产生的方法有很大关系,如文献[2]提出的树签名由于包含了大量冗余信息而导致存储开销大、更新代价高.本文在路径字典和签名方法的基础上提出了二维字典签名方法,这种方法在路径字典的基础上产生签名,可以把路径字典和签名方法的优点结合起来,达到基本设计思想的要求.

2 二维字典

2.1 二维层次结构

图1是一个二维层次的例子,它反映了面向对象数据模型的结构特点.



①聚集链,②继承链.

Fig. 1 Planar hierarchy
图1 二维层次

2.2 二维字典的定义及数据结构

路径字典是二维字典的基础,它仅包含聚集链上的对象之间的信息,通过S-表达式将这种信息表达出来.S-表达式方法把终止于叶类的同一对象的所有路径编码为一个递归表达式.例如,$E1$=Company[1](Vehicle[12](Person[4]))就是路径Person.Vehicle.Company上的一个S-表达式.

为了增加继承层次上的信息,需对路径字典进行如下扩展:

(1) C_i^* 表示包括继承层次的根类C_i及其子类在内的这一继承层次中所有类的集合.

(2) 将原来的在路径$C_1C_2\ldots C_n$上定义S-表达式,改为对路径$C_1^*C_2^*\ldots C_n^*$定义S^*-表达式,即路径字典中定义S-表达式所在路径上的每一个类$C_i(1\leqslant i\leqslant n)$从原来只属于聚集层次上的类$C_i$扩展为除原有聚集层次上的类$C_i$之外还包括该类的子类(只要该类有子类存在),这里把这种包含继承信息的路径称为继承路径.

继承路径$C_1^*C_2^*\ldots C_n^*$的S^*-表达式定义如下:

(1) $S_1^*=\theta_1$,其中θ_1是类集合C_1^*中的一个类的对象的OID或空;

(2) $S_i^*=\theta_i(S_{i-1}^*[,S_{i-1}^*]),1<i\leqslant n$,其中$\theta_i$是类集合$C_i^*$中一个类的对象的OID或空,$S_{i-1}^*$是继承路径$C_1^*C_2^*\ldots C_{i-1}^*$的一个$S^*$-表达式.$S_i^*$是第$i$级的$S^*$-表达式,其中与$\theta_i$相联系的链递归地包括了$\theta_i$的所有祖先对象的OID,该链称为$\theta_i$的祖先链.除了$C_1$中的对象,路径上的每一个对象都有一条祖先链.对于继承路径$C_1^*C_2^*\ldots C_n^*$,它的二维字典就是n级的S^*-表达式序列.

下面的$E1^*$是继承路径Person.Vehicle*.Company上的一个S^*-表达式:

$E1^*$:Company[1](Vehicle[12](Person[4]),Automobile[2](Person[10]),
Truck[1](Person[8],Truck[2](Person[9]))

S^*-表达式的数据结构需要进行如下改进:

(1) S-表达式数据结构的头部原来是指向沿路径各个类在一条S-表达式里该类中第1个对象的OID的指针序列,对于路径$C_1C_2\ldots C_n$,这个指针序列为$SP_nSP_{n-1}\ldots SP_2$.对于继承路径$C_1^*C_2^**\ldots C_n^**$,对应的指针序列修改为$SP_n^*SP_{n-1}^*\ldots SP_2^*$,$SP_i^*$ $(2\leqslant i\leqslant n)$为指向类$C_i$及其子类在一条$S^*$-表达式里第1个对应类中对象的

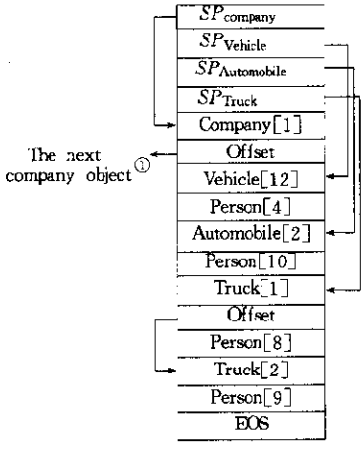
OID 的指针子序列,在一条 S-表达式里不存在某个类中对象标识符的情况下,对应于该类的指针为空.

例如,对于继承路径 Person.Vehicle*.Company,S-表达式数据结构的头部指针序列为 $SP_{Company}SP_{Vehicle}$,而对应的 S^*-表达式数据结构的头部指针序列为 $SP_{Company}SP_{Vehicle^*}$,$SP_{Vehicle^*}$ 是指针子序列 $SP_{Vehicle}SP_{Automobile}SP_{Truck}$. 图 2 是 S^*-表达式 $E1^*$ 的数据结构图.

(2) 对象标识符在通常情况下有物理地址和逻辑地址两种表示方法. 为了实现对继承层次信息的查询,本文假设 OID 中还包含对象所属类的类标识符. 例如,对应图中的 Vehicle 类及其子类 Automobile 和 Truck,可使它们的对象的 OID 分别包含按词典序划分的类标识符 B2,B2A,B2B. 这样就可以通过包含反映类及子类之间继承关系的对象标识符来实现对继承层次信息的查询处理.

3 二维字典签名

在 80 年代,签名文件主要用于文本检索. 近年来,签名文件的应用领域扩展到了面向对象数据库系统. 抽取签名的方法有多种,叠加代码法是较常用的一种[3]. 它将文档分成一些逻辑块,每个逻辑块为包含常数个字的文本片. 对每个字产生其中有几位被设置为"1"的一定长度的位模式. 这些位模式或运算的结果就是块签名,文档的所有块签名结合在一起就形成了文档签名.



①下一个company对象.

Fig. 2 Data structure of S^*-expression $E1^*$

图2 S^*-表达式 $E1^*$的数据结构

利用签名可以检查出一个给定的值是否在文档内. 字的查找处理过程如下:首先产生查询字的查询签名,然后检查每一个块签名,当且仅当块签名的对应位全部为"1"时才检索该块,这个过程称为查询匹配. 通过查询匹配可以筛选出候选签名,但筛选出的匹配文本块有可能存在误匹配的现象. 如果所查询的字不存在于块内,但查询匹配却得出了匹配成功的结果,这种假象就称为误匹配.

对象签名是二维字典签名产生的基础. 对于所有的对象,简单属性的属性签名是对其简单属性值运用 Hash 函数而得到的,嵌套属性的属性签名则是通过 Hash 嵌套对象的对象标识符来产生的. 叠加对象的所有简单属性签名和复杂属性签名就产生了对象签名.

每个二维字典签名在二维字典签名文件中都有入口〈对象签名,OID,指针〉. OID 是一条 S^*-表达式中某对象的对象标识符,指针指向该 S^*-表达式. 若二维字典签名与查询签名匹配,则可以通过指针找到二维字典中对应的 S^*-表达式,再借助候选的对象标识符就可以找到目标对象的对象标识符. 为了提高查询效率,所有二维字典签名可按类存放.

二维字典签名的构造算法如下:

输入:S^*-表达式[m]

输出:二维字典签名 $P\_sig[m]$

(1) 读取 S^*-表达式[m]的第 1 个对象;

(2) 在对象数据库中检索该对象每一个属性的属性值,并运用有关的签名产生方法产生属性签名;

(3) 叠加该对象的所有属性签名产生对象签名;

(4) 将该对象的对象签名、对象标识符及指向 S^*-表达式的指针连接起来,构造该对象在所属类的二维字典签名文件中的入口;

(5) 若 S^*-表达式[m]还有下一对象,读取它并重复步骤(2)~(4).

4 二维字典签名方法

4.1 二维字典签名方法的有关操作及算法

二维字典签名方法主要由二维字典签名文件和二维字典文件两大部分组成.

利用二维字典签名方法进行的检索和更新操作如下.

〈1〉检索操作

假设有一个查询Q,它以C_t^*为目标类,以C_p^*为谓词类,其中$1<t,p<n$.注意,这里假设检索的目标类C_t^*包括目标类C_t及其所有子类,谓词类C_p^*也是包含了类C_p和以类C_p为父类的所有子类,并假定查询返回的结果是符合查询条件的目标类中的对象标识符.

检索算法如下:

输出:以C_t^*为目标类、C_p^*为谓词类的查询Q

输出:满足查询条件的目标类对象标识符集 targetOIDset

(1) 首先由查询谓词产生查询签名,多条件查询时有多个查询签名;

(2) 对于每个查询签名,查找与查询谓词相关的类所对应的二维字典签名文件,通过查询匹配筛选出候选的对象标识符集和指向S^*-表达式的指针集(多条件查询时有多个指针集);

(3) 若为多条件查询,则对多个指针集进行交运算,若存在误匹配,则将多个指针集的交运算结果作为候选指针集,并相应地修改候选对象标识符集,剔除引起误匹配的对象;

(4) 在对象数据库中检索候选对象标识符集中的对象,若发现误匹配,则修改候选指针集;

(5) 利用候选指针集中的指针读取对应的S^*-表达式,借助候选对象标识符集中的对象筛选出目标类的对象标识符,并放入目标类对象标识符集中;

(6) 将目标类对象标识符集作为查询结果返回.

上述检索过程避免了顺序查找数据库和访问介于目标类和谓词类之间的对象.对于多条件查询,通过候选指针集的交运算可在检索数据库之前消除或减少可能存在的误匹配.

〈2〉更新操作

更新路径上对象的属性值有如下两种情况:

(1) 若对象的简单属性值发生变化,或对象的非路径上嵌套属性值发生变化,均只需重新生成对象的对象签名并写入二维字典签名文件中,而没有必要改变二维字典,因为二维字典中反映的是路径上嵌套对象之间的链接信息;

(2) 若变化发生在路径中对象的嵌套属性上,则这种变化将对二维字典签名文件和二维字典文件产生影响.在此假设类C_i的对象O_i有一个嵌套对象O_{i+1},更新操作将要把O_{i+1}变为O'_{i+1}.O_{i+1}和O'_{i+1}均为类C_{i+1}的对象,且O_{i+1}和O'_{i+1}所在的两个S^*-表达式不相同,分别为$P1$和$P2$.

综合考虑更新操作的以上两种情况,给出如下更新算法.

输入1:对象O_i新的简单属性值或其非路径上嵌套属性的新属性值

输入2:对象标识符O_i,O_{i+1}和O'_{i+1}

输出:更新后的二维字典签名文件和二维字典文件.

(1) 检索类C_i的二维字典签名文件,找到对象O_i及指向S^*-表达式$P1$的指针;

(2) 检索数据库中的对象O_i,更新其有关属性值,产生新的对象签名;

(3) 这里分两种情况:

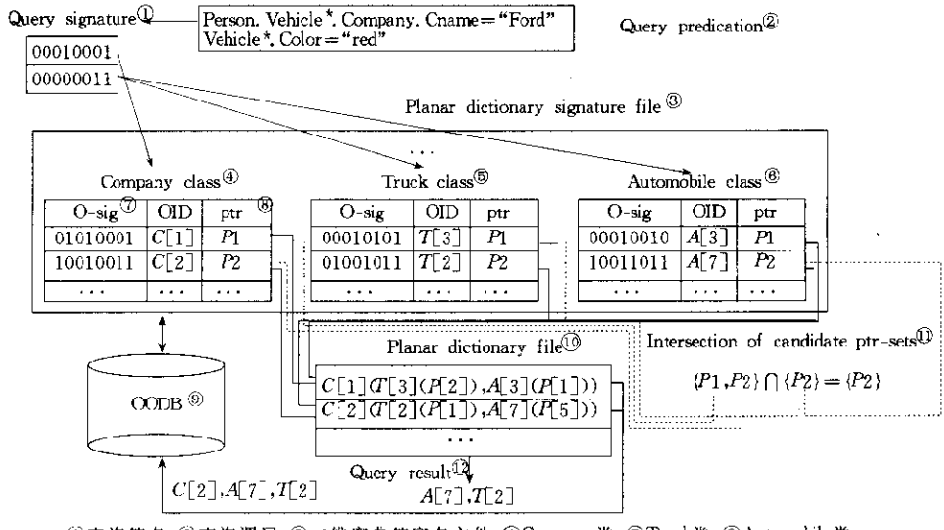
① 若改变的是对象O_i的简单属性值或其非路径上嵌套属性值,则将O_i的新对象签名重写入二维字典签名文件;

② 若在对象O_i的路径上复杂属性值从O_{i+1}变为O'_{i+1},则先利用第(1)步中的指针检索S^*-表达式$P1$,将O_{i+1}及其祖先链从$P1$中移出,然后在类C_{i+1}的二维字典签名文件中查找O'_{i+1},从而获得指向S^*-表达式P_2的

指针,用 O_{i+1} 的祖先链更新 O'_{i+1} 的祖先链,最后修改 O_i 原来的对象签名,并将 O_{i+1} 的祖先链上所有对象的指针由指向 $P1$ 修改为指向 $P2$.

4.2 查询实例

下面举一个二维层次上多条件查询的实例来说明如何运用二维字典签名方法进行查询处理.设多条件查询为"检索 Ford 公司生产的所有红色的汽车",它的目标类为类 Vehicle 及其子类 Automobile 和 Truck.该查询处理过程如图 3 所示.



①查询签名,②查询谓词,③二维字典签字名文件,④Company类,⑤Truck类,⑥Automobile类,⑦对象签名,⑧指针,⑨对象数据库,⑩二维字典文件,⑪候选指针集的交运算,⑫查询结果.

Fig. 3 Multi-Condition query on planar hierarchy
图3 二维层次上的多条件查询

首先要产生两个查询签名,并经过查询匹配把匹配的对象签名、对象标识符以及指向 S^*-表达式的指针找出来.由图 3 可见,对于谓词 Person. Vehicle*. Company. Cname="Ford",查询匹配得到的匹配 OID 为 $C[1]$和 $C[2]$,相应的指针为 $P1$,$P2$,而对于谓词 Vehicle*. Color="red",经过查询匹配得到匹配对象的标识符为 $A[7]$和 $T[2]$,相应的指针为 $P2$.由于是多条件查询,就要对各条件经过查询匹配所得到的指针集进行交运算,以减少误匹配,进行交运算后得到的候选指针为 $P2$.然后检索对象数据库中的对象 $C[2]$,$A[7]$和 $T[2]$的相关属性值,看是否满足查询谓词,从而完全消除误匹配.在此,假设已不存在误匹配,则利用候选指针 $P2$ 检索对应的 S^*-表达式,得到目标对象标识符 $A[7]$和 $T[2]$,这就是查询结果.

5 代价模型

本文采用以下参数来描述在路径 $C_1^*C_2^*\ldots C_n^*$ 上的类及其属性的特征,其中部分参数来自文献[1].参数 i,j,r 满足 $1\leqslant i\leqslant n$,$1\leqslant j\leqslant m$,$1\leqslant r\leqslant d$,且所有长度和大小都以字节为单位.

n:路径上类 C_i^* 的个数,

m:每个类 C_i^* 的子类个数(假设相同),

d:每个类的属性个数(假设相同),

$N_{i,j}$:类 C_i^* 的子类 $C_{i,j}$的对象个数,

N_i:类 C_i^* 中的对象个数(为 $N_{i,j}$之和),

S_i:类 C_i^* 中一个对象的平均长度,

A_r:路径上的复杂属性,

$D_{i,j}$:类 C_i^* 的子类 $C_{i,j}$的属性 A_r 的不同值个数,

D_i：类 C_i^* 的复杂属性 A_r 的不同值个数(为 $D_{i,j}$之和)，

k_i：类 C_i^* 中对象与 A_r 的值之间的共享调用比率($k_i=N_i/D_i$)，

$A_{i,r}$：类 C_i^* 的第 r 个简单属性，

$U_{i,r}$：类 C_i^* 的简单属性 $A_{i,r}$的不同值个数，

$q_{i,r}$：类 C_i^* 的对象与属性 $A_{i,r}$的值之间共享属性值的比率($q_{i,r}=N_i/U_{i,r}$)，

K：共享调用比率(如 k_i)与共享属性值比率(如 $q_{i,r}$)的平均比率，

$L_{sig}(i)$：类 C_i^* 的对象签名的长度，

pp：一个页指针的长度，

P：页大小，

$UIDL$：一个对象标识符的长度，

$OFFL$：二维字典里一个位移域的长度，

SL：二维字典的第1个域的长度，

EL：EOS的长度，

FSL：自由空间目录的自由空间域长度.

查询性能用I/O存取次数来衡量.由于页是主存与外存之间数据传输的基本单元，本文采用页为单位来评估存储开销和查询代价.

为探讨面向二维层次的二维字典签名方法的代价模型，首先作如下假设：

① 每个类 C_i^* 的子类之间均存在类 $C_{i,k}$为类 $C_{i,j}$子类的关系($j+1\leqslant k\leqslant m$)；

② 不存在部分实例，即每一个类 C_i 的实例都被类 C_{i-1}中的实例调用；

③ 所有属性都是单值的，且属性值均匀分布于对象之间；

④ 查询为二维层次上的单条件查询；

⑤ 任意类 C_i^* 的所有对象签名长度相等；

⑥ $k_i=q_{i,r}=K$，其中 K 为平均比率.

(1) 存储开销

二维字典签名方法的存储开销是所有二维字典签名文件和二维字典文件的存储开销总和.假设入口 $L_{entry}(i)$表示一条 S^*-表达式所包含的类 C_i^* 中一个对象在该类所对应的二维字典签名文件中所占字节总数.类 C_i^* 的签名文件入口为

$$L_{entry}(i)=L_{sig}(i)+UIDL+pp, \tag{1}$$

类 $C_{i,j}$的二维字典签名文件的大小为

$$F_{sig}(i,j)=\begin{cases}\lceil D_{i,j}/\lfloor P/L_{entry}(i)\rfloor\rceil, & L_{entry}(i)\leqslant P\\ D_{i,j}*\lceil L_{entry}(i)/P\rceil, & L_{entry}(i)>P\end{cases}, \tag{2}$$

S^*-表达式的平均大小 SS_1 为

$$SS_1=SL\times(n\quad 1)*m+(UIDL+OFFL)*NOBJ+EL. \tag{3}$$

二维字典中所有 S^*-表达式所需的页数 SSP_1 为

$$SSP_1=\begin{cases}\lceil D_n/\lfloor P/SS_1\rfloor\rceil, & SS_1\leqslant P\\ D_n*\lceil SS_1/P\rceil, & SS_1>P\end{cases}, \tag{4}$$

自由空间目录所需的页数 FSD_1 为

$$FSD_1=\lceil SSP_1*(pp+FSL)/P\rceil. \tag{5}$$

存储二维字典签名文件和二维字典文件所需要的总开销 S_{PDS}为

$$S_{PDS}=\sum_{i=1}^{n}\sum_{j=1}^{m}F_{sig}FSD_1+SSP_1. \tag{6}$$

(2) 检索代价

为便于讨论检索代价模型，假设以 $C_{i1,i2}^*$来表示类 $C_{i1,i2}$及以类 $C_{i1,i2}$为根的所有子类，以 $C_{j1,j2}^*$来表示类 $C_{j1,j2}$

及以类 $C_{p1,p2}$ 为根的所有子类,其中 $1\leqslant t1,p1\leqslant n,1\leqslant i2,p2\leqslant m$. 这里所探讨的检索代价是关于以 $C^*_{i1,i2}$ 为目标类、以 $C^*_{p1,p2}$ 为谓词类的单条件查询 Q. 假设经查询匹配检索到的候选对象有 $candi$ 个,且数据库中每个对象所占有的存储空间小于 1 页,候选对象存储在不同页. 此外,假设不存在误匹配现象. 对象检索所存取的页数 $R_{PDS}(p1,p2)$ 如下:

$$R_{PDS}(p1,p2)=\sum_{j=p2}^{m}F_{sig}(p1,j)+candi+candi*\lceil SS1/P\rceil. \tag{7}$$

(3) 更新代价

这里探讨的更新代价模型是针对更新对象的路径上复杂属性值的情况而构造的. 假设要求将 $O_{i,j}$ 的复杂属性值从 O_{i+1,r_i} 变更为 $O_{i+1,w}$,其中 $1\leqslant i\leqslant n,1\leqslant j,r_i,w\leqslant m$,设含 O_{i+1,r_i} 和 $O_{i+1,w}$ 的表达式在不同页,且 O_{i+1,r_i} 原来的祖先链为 $O_{1,r_1}O_{2,r_2}\cdots O_{i,r_i}$,其中 $1\leqslant r_1,r_2,\ldots,r_i\leqslant m$. 注意,检索二维字典签名文件时不存在误匹配现象,因为是对取值唯一的 OID 进行检索. 为简化分析,不考虑由更新操作而引起的页溢出消耗的代价. 更新总代价为

$$U_{PDS}=F_{sig}(i,j)+F_{sig}(i+1,w)+2+4*\lceil SS1/P\rceil+F_{sig}(1,r_1)+F_{sig}(2,r_2)+\ldots+F_{sig}(i,r_i). \tag{8}$$

6 结束语

由于签名产生的方法不同,现有的签名方法往往存在冗余信息量大、误匹配率大、签名的维护代价高等缺点,而二维字典签名是在二维字典的基础上产生的,每个对象都在所属类的二维字典签名文件中有对应入口,因此误匹配率较小,维护代价也较低.

从现有的文献来看,二维层次上的索引方法较少. 文献[4,5]分别提出了通用索引(generalized index)和 U 索引(uniform index),介绍了索引结构及有关操作,但只是通过分析有关操作算法的步骤来与其他索引方法进行比较而得出结论,没有构造代价模型,也没有进行实验,缺乏说服力. 文献[6,7]分别提出了嵌套-继承索引(nested-inherited index)和通用的嵌套-继承索引(generalized nested-inherited index). 它们都是与聚集层次上的索引通过实验进行比较,给出了实验的有关数据和结论,但具体的代价模型不明确. 在这种情况下,本文只能选取聚集层次上的路径索引和路径字典索引作为实验的比较对象,在 Matlab 环境下进行仿真. 仿真结果表明,二维字典签名方法是一种有效的签名方法.

与索引方法相比,二维字典签名方法有以下几个特点.

(1) 索引往往建立在查询频率较高的少数属性上,但 OO 查询的查询语义比传统的要丰富得多,也复杂得多,要想对各种 OO 查询都提供有效的支持,索引方法所需的存储开销必然很大,维护代价也很高. 在查询谓词未知的情况下,索引通常只能对部分查询提供支持. 而在二维字典签名方法中,二维字典有助于高效遍历聚集链上的对象,而且签名文件包含了继承路径上所有对象的抽象信息,所以能对各种类型的 OO 查询提供较通用的支持.

(2) 在多条件查询时,通过检索时多个候选指针集的交运算,可以对二维字典和数据库进行快速扫描.

(3) 目前,二维层次上的索引方法还很少考虑共享调用这种对象之间常见的关系,因此,往往随着共享调用度的增大,存储的冗余信息也越多. 而实验表明,二维字典签名方法的存储开销随着平均共享调用度的增大而大幅度下降,并且检索代价和更新代价也在降低,这主要是因为二维字典采用了 S^*-表达式方法来表现对象之间的共享调用关系.

(4) 仿真结果表明,二维字典签名方法的检索性能要比路径索引和路径字典索引好.

总而言之,二维字典签名方法更适用于查询谓词未知、多条件查询、聚集层次上的对象之间共享调用度高以及检索操作多于更新操作的应用环境,是支持面向对象查询物理层优化的一种有效机制. 但它仍有不足之处,如当对象之间的平均共享调用比率为 1 时,存在着存储开销较大的缺点,这有待于在今后的研究中进一步加以改进.

参考文献

1 Lee W C, Lee D L. Combining indexing technique with path dictionary for nested object queries. In: Ling T W, Masunaga

Y eds. Proceedings of the 4th International Conference on Database Systems for Advanced Applications. Singapore: World Scientific Publishing Co. Pte Ltd., 1995. 107~114

2 Lee W, Lee D L. Signature file methods for indexing object-oriented database systems. In: Proceedings of the 2nd International Computer Science Conference. 2000. http://www/cs.ust.hk/~dlee/

3 Davis R S, Ramamohanarao K. A two level superimposed coding scheme for partial match retrieval. Information Systems, 1983,8(4):273~280

4 Fotouhi F, Lee T G, Grosky W I. The generalized index model for object-oriented database systems. In: Institute of Electrical and Electronics Engineers eds.. Proceedings of the 10th Annual International Phoenix Conferencee Computers and Communications. Los Alamitors, CA: IEEE Computer Society Press, 1991. 302~308

5 Gudes E. A uniform indexing scheme for object-oriented databases. Information Systems, 1997,22(4):199~221

6 Bertino E. An indexing technique for object-oriented database. In: IEEE Computer Society ed. Proceedings of the 7th Conference Data Engineering. Los Alamitors, CA: IEEE Computer Society Press, 1991. 160~170

7 Shidlovsky B, Bertino E. A graph-theoretic approach to indexing in object-oriented databases. In: Stanley Su W U ed. Proceedings of the 12th International Conference on Data Engineering. Los Alamitors, CA: IEEE Computer Society Press, 1996. 230~237

Planar Dictionary Signature Method for Physical Optimization of Object-Oriented Query

WU Yan-ping SHI Run-shen

(*Department of Computer Science and Technology Shanghai Tiedao University Shanghai* 200331)

Abstract Planar dictionary signature method which supports the physical optimization of object-oriented query is proposed in this paper. The basic design idea of the proposed method is introduced firstly. After the definition of planar dictionary and its data structure, the construction algorithm for planar dictionary signature and the query operation algorithms are proposed. Finally, the storage cost and the query cost models are constructed.

Key words Planar hierarchy, planar dictionary, planar dictionary signature, cost model.