维普资讯 http://www.cqvip.com

计算机科学 1997Vol.24№.6

面向对象数据库系统的索引技术

An Index Technique for Object-Oriented Database Systems

廖春维 施伯乐

（复旦大学计算机科学系 上海 200433）

摘 要 索引技术是数据库系统中的一项重要技术。在面向对象数据库系统中，由于模型本身具有许多复杂的特性，索引技术显得更有意义，也更为复杂。本文给出了一种高效的、切实可行的索引技术——嵌套继承索引技术，阐述了创建嵌套继承索引的方法，以及在这种索引下，检索、插入和删除操作的具体实施。

关键词 面向对象数据库，复合对象，继承，查询，索引

在面向对象数据库系统中，对象是由结构和行为构成的，所以面向对象数据库系统的索引技术可以是结构索引技术，也可以是行为索引技术。结构索引技术基于对象属性，能高效地计算包含有属性的查询谓词；行为索引技术基于方法，能高效地执行包含有方法调用的查询。由于对象之间存在着两种关系：复合关系和继承关系，所以在结构索引技术中，有的索引技术能对嵌套谓词提供支持，有的索引技术能对类继承层次提供支持。

本文中，我们将给出结构索引技术，它的新颖独到之处在于：对沿着复合（聚集）层次和继承层次的查询能提供整体支持，而且我们的索引技术也能同行为索引技术有机地结合在一起。

1 预备知识

首先，我们列出一些预备定义和记号以开展论述。

C^*：表示以类 C 为根的类继承层次中所有类的集合。

P：一条路径，表示某聚集层次中的一条分支，它可以形式地定义如下：

$P=C_1.A_1.A_2.\cdots.A_n (n\geqslant 1)$

其中：C_1：是数据库模式中的类；A_1：是类 C_1 的一个属性；A_i：是类 C_i 的一个属性，而类 C_i 是类 C_{i-1} 的属性 A_{i-1} 的域（类），$1<i\leqslant n$。又有：

$len(P)=n$，表示路径的长度

$class(P)=C_1\cup\{C_i | C_i$ 是类 C_{i-1} 的属性 A_{i-1} 的域类，$1\leqslant i\leqslant n\}$

$scope(P)=\bigcup_{C_i\in class(P)}C_i^*$，其中 C_1 称为（范围）scope（P）的根

给定路径 P，类 $C\in scope(P)$，如果 C 是以类 C_i 为根的类继承层次中的类，而 $C_i\in class(P)$，那么 $pos(C)=i$，称 C 的位置是 i。

其实，路径 P 的范围 scope(P)就是路径上所有类及这些类的子类所组成的集合。

在后面的讨论中，我们还需要区分这样两个概念：例示和成员。我们说一个对象是一个类 C 的例示，是指类 C 是类层次中同该对象相联系的最具体的类，我们说一个对象是一个类 C 的成员，是指该对象是类 C 或类 C 的某个子类的例示。所以，类 C 的例示是属于 C 且又不属于 C 的任何子类的那些对象。

2 嵌套继承索引技术

我们，首先阐述嵌套继承索引是如何组织的：

对于给定的一条路径 $P=C_1.A_1.A_2.\cdots.A_n$，一个嵌套继承索引让属性 A_n 的一个值 v 同 scope(P)中所有各类满足条件——例示的（嵌套）属性 A_n 持有键值 v——的所有例示的 OID 集合相对应。

嵌套继承索引能支持快速的检索操作。对于更新操作，嵌套继承索引不需要遍历所有对象，因为索引中保存有某些附加的有用信息。在嵌套继承索引中，非叶结点的结构同传统的基于 B^+ 树索引的结点

维普资讯 http://www.cqvip.com

的结构很相似，而叶结点中的记录（我们称之为主记录）的结构则有很大的不同。主记录中包含有以下信息：

□记录长度；□键长度；□键值；□类-目录。

对于scope(P)中每个类，主记录中还包括：列表L的元素个数，列表L本身。而列表L本身又具有如下的结构：（首先假设Pos(C)=i）

－若A_n是单值属性，或者i=n，那么列表L中的元素是类C的所有满足条件－该例示的（嵌套）属性A_n持有键值v--的例示的OID。

－若A_n是多值属性，那么列表L的元素是形如(OID,numchild)的偶对，其中，OID是满足条件--其（嵌套）属性A_n持有值v--的某个对象O的对象标识；而numchild是该对象O的孩子数目。（注：这里的父亲-孩子关系是沿着类复合层次的引用关系，被引用的例示称为孩子例示，引用孩子例示的例示称为父亲例示，以下如果没有特殊说明，则与此相同）。

类-目录中记录的条数等于满足以下条件的类的数目：存在该类的某个例示，其（嵌套）属性A_n持有键值v。

对于每个满足上述条件的类C_i，目录中都有相应的一条记录，记录包含有以下信息：

□类标识符

□偏移量（在主记录中的相对地址），存放有类C_i的例示的OID列表。（如果A_i是多值属性，且$i<n$，则存放的是(OID,numchild)偶对的列表）

□指针（辅助记录的地址），存放有类C_i的每个例示的父亲例示的OID列表

我们的嵌套继承索引为scope(P)中的每个类（除路径的根及其子类外）分配一个辅助记录。一个辅助记录由一个如下的四元组的序列组成：

(OID,指针,父例示数目n,{P-oid_1,…,P-oid_n})

类C_i总共有多少个例示，辅助记录中也就有多少个这样的四元组。具体地，对于类C_i的某个例示O，相应的四元组中的元素是：对象O的OID、主记录的地址（指向主记录的指针）、父亲例示的个数、以及一个父亲例示的OID的列表；在以上四元组的格式定义中，P-oid_j就表示对象O的第j个父亲例示。

将辅助记录和主记录存放在不同的页内，一个主记录可以同多个辅助记录有联系。我们还为辅助记录创建一个二级B^+树索引，这个索引是辅助记录中的那些四元组的基于OID的索引。所以，在我们的索引组织中，实际上有两个索引：一个是主索引，另一个是辅助索引。主索引是基于属性A_n的值的，它将A_n的一个值v同scope(P)中各类满足条件--例示的（嵌套）属性A_n持有键值v--的所有例示的OID联系起来；而辅助索引是基于对象O的OID的，它将对象O的OID同对象O的父亲例示的OID（列表）联系起来。主索引的叶结点中有一个指向辅助索引的叶结点的指针；反之亦然。所以，我们说，主索引是针对（嵌套）属性A_n的值进行倒排的，主索引常用于检索操作；而辅助索引是针对scope(P)中各类（除路径的根及其子类）的例示的OID而进行倒排的。对于删除和插入某指定对象O等这一类更新操作来说，利用辅助索引可很快地找出相应那些其中存放有该例示O的OID的主记录。另外，在我们的索引组织中，辅助的B^+树结构也可以很好地处理多值属性的情形。

3 各类操作的实施

在这一节中，我们阐述在嵌套继承索引组织中，检索、插入和删除等操作是怎样实施的。

A. 检索。嵌套继承索引能够快速地计算出包含某索引属性的谓词，所以对于下面这样的查询，嵌套继承索引技术将会大有用武之地：查询目标是scope(P)中的类（或类层次），而路径P本身又是以该索引属性结尾的。检索操作的执行步骤简述如下：

(1)用给定的键值，查找主索引；

(2)访问主记录，查找其中的类-目录，找出查询目标（类）的例示的OID所在的相对地址（即偏移）；

(3)取回这些OID，并将它们作为查询结果返回。

B. 插入。已知，有路径$P=C_1.A_1.A_2.\cdots.A_n$，类$C\in scope(P)$，pos(C)=i，对于路径P，建立了相应的嵌套继承索引。现在要将类C的一个例示、-对象O插入到数据库中去。下面，我们就给出这个插入操作的具体执行步骤：

(1)确定对象O的属性A_i的值的SCH，如果A_i是单值属性，SCH中就只含有一个值；

(2)使用SCH中的值作为键值，搜索辅助索引；

维普资讯 http://www.cqvip.com

(3)检索出同 SCH 中各个值相对应的那些四元组,对于每个找到的四元组,将对象 O 插入到四元组里的父亲例示列表中,并且确定所有指向主记录的指针(主记录的地址)的集合:S。

(4)对每个其地址出现在 S 中的主记录 p 做以下工作:□访问主记录 p;□在其中查找类-目录,确定类 C 的例示的 OID 列表在主记录中的相对地址(偏移),确定类 C 的辅助记录的地址;□在 OID 列表中插入对象 O 的 OID(如果 A_i 是多值属性,且 $i<n$,那么在(OID,numchild)偶对列表中插入相应的偶对,其中 OID 标识对象 O,numchild 域也要进行适当的初始化工作)。

(5)在类 C 的辅助记录中,插入一个新的四元组,该四元组里的第一个元素是对象 O 的 OID。

C. 删除。已知,有路径 $P=C_1.A_1.A_2.\cdots.A_n$,类 $C\in scope(P)$,$pos(C)=i$,对于路径 P,建立了相应的嵌套继承索引。现在要将类 C 的一例示--对象 O 从数据库中删除掉。

一次删除操作对索引组织的整体影响是:将对象 O 的 OID 从每一个包含有它的主记录中删除掉;如果在某个这样的主记录中已经不再含有对象 O 的兄弟例示了(对象 O 的父亲例示的其他孩子例示),还要将所有引用 O 的那些例示从主记录中删除;对引用 O 的那些(父亲)例示从主记录中删除掉;对引用对象 O 的那些(父亲)例示(除路径的根及其子类的例示外)的四元组进行修改;同时,将对象 O 的四元组从类 C 的辅助记录中删除掉;最后,还要对对象 O 的所有孩子例示的四元组进行修改,从这些四元组里的父亲例示列表中将对象 O 删去掉。具体地,这个删除操作执行以下步骤:

(1)确定对象 O 的属性 A_i 的值集 SCH,如果 A_i 是单值属性,SCH 就只含有一个值。

(2)使用 SCH 中的值和对象 O 的 OID 作为查找键,搜索辅助索引。

(3)检索出同 SCH 中各值相对应的那些四元组,将对象 O 从这些四元组里的父亲例示列表中删除掉。

(4)访问对象 O 的四元组,确定对象 O 的父亲例示的 OID(可能是多个),确定所有指向主记录的指针(主记录的地址)的集合:S,将对象 O 及其所有的父亲例示的 OID 保存到一个临时列表 listparent 中,而后将该四元组删除掉。

(5)只要列表 listparent 非空,就执行如下操作:

(a)对每个其地址出现在 S 中的主记录 p,做以下工作:

□访问主记录 p。

□查找其中的类-目录,确定列表 listparent 中各个 OID 在主记录中的相对地址(偏移),如果外层循环是刚开始的话,那么还要从类-目录中检索出对象 O 所属的类。

□对 listparent 列表中的每个 OID,做这样几件事:如果该 OID 标识的是对象 O,或者在主记录里的有关列表中,OID 是单独地存放的,而不是和"numchild"域一道存放的话,那么就将该 OID 从主记录的 OID 列表中删除掉;如果该 OID 标识的是对象 O 的某个父亲例示,就将该 OID 同主记录 p 的地址一道插入到一个辅助列表 L 中;如果是其他情况,只是将该 OID 相关的 numchild 的值减去 1 即可,如果 numchild 的值已减到零了,就将该偶对(OID,numchild)从主记录中删除掉;如果该 OID 不是路径的根的某个成员的话,就将该 OID 同主记录 p 的地址一道插入到辅助列表 L 中去。

(b)用列表 L 中的 OID 作为键值,搜索辅助索引,并将 listparent 列表置空。

(c)访问列表 L 中各 OID 的四元组,从四元组中删去所有这样的主记录的地址(指向主记录的指针):该 OID 已经从这个主记录中删除了。然后,确定各个四元组里的父亲例示的列表,并将它们链接成一个新的 listparent 列表。

重复步骤(5),直到 listparent 列表为空。

另外,还有一种检索四元组的策略:(1)首先,查找主记录;(2)其次,查找类-目录;(3)最后,扫描辅助索引。这种策略只有在一个记录仅放存于一页或少数几页中时,才可行。

参考文献

[1]P. Valduriez. Jion Indices. ACM Trans. Database Systems. [2(2)]1987

[2]E. Bertino and C. Guglielmino. Path-Indexing: An approach to the efficient execution of object-oriented queries. Data and Knowledge Eng., North-Holland, 1993