空间查询优化

方 裕 楚 放
(北京大学计算机系,北京 100871)

摘 要 空间查询优化是空间应用的突破点.由于现有的关系优化不能适应空间数据的查询,因此空间系统必须具有自己的代价模型和优化器.为此,给出了一个空间查询优化的系统方案 FQPro,并在对空间查询优化的几个阶段做了一般性探讨后,将重点放在代价模型、谓词代价计算和优化方案的代价计算上,尤其对基于 R-树的代价模型给予了详细介绍.另外,参照关系优化器,FQPro 还定义了一套谓词代价公式和谓词选择性公式,并在此基础上定义了查询方案代价计算公式和算法.文章最后指出,代价模型和可扩展的体系机构是空间查询优化系统的发展方向.

关键词 空间查询优化 FQPro 代价模型 谓词代价 谓词选择性
中图法分类号:TP301 TP301.6 **文献标识码**:A **文章编号**:1006-8961(2001)04-0307-08

Spatial Query Optimization

FANG Yu, CHU Fang
(*Computer Department, Peking University, Beijing* 100871)

Abstract Spatial query optimization should be the focus of attention in spatial area. Since relational optimizer is unsuitable for dealing with spatial data, spatial systems should have their own cost model and optimizer. In this paper, a spatial query optimization system—FQPro was presented. After the overview of general phrases of spatial query optimization, we put the emphasis on cost model, calculation of predicate costs and plan costs. In particular, we give a detailed introduction of the cost model based on R-tree. Similar to relational systems, FQPro defines a set of formula for predicate costs and predicate selectivity respectively. They are used in G/SQL. On top of these formulas, the idea and algorithm for forming the optimum execution plan is defined. Concluding this paper are some issues that should be the concerns of research on spatial optimization, primarily the cost model and the extensible system architecture. Also, this paper summarizes the challenges and opportunites facing spatial query processing in the end.

Keywords Spatial query optimization, FQPro, Cost model, Predicate cost, Predicate selectivity

0 背 景

由于空间查询能够处理空间数据库中的多维数据和满足广泛的应用类型,如地理信息系统(GIS),计算机辅助设计(CAD),多媒体系统,医学或卫星图象库等,因此对空间查询的研究近年来颇受关注,这是因为空间数据量庞大,数据结构复杂,操作代价昂贵的缘故,所以空间查询的优化势必成为空间应用的难点和突破点.由于关系数据库现有的优化不能完全适用于空间数据的查询,因此空间查询的优化系统需要建立自己的代价模型.

这种代价模型的研究重点是空间操作的代价估计,我们尤其关注窗口查询(window query)和叠置(join)两个操作,因为它们是空间查询中最重要,也是代价比较昂贵的两个操作[1~3].操作代价主要包括:执行时间、I/O 时间,以及操作结果的输出时间[4],同时这种操作结果的大小涉及一个重要概念——谓词选择性(predicate selectivity),即满足谓词的结果集合的秩与源对象集合的秩的比值.设源对象数为 N,结果对象数为 r,,则选择性 $S=r/N$,其取值在[0,1]之间,且 S 越大,选择性越低,最低为 1,即所有对象都被选中,谓词选择性大小在相当程度上决定了优化结果中谓词的执行顺序,空间查询优化是一个有待开发的领域[4],至今还没有人提出一个完整的优化系统设

基金项目:国家"九五"攻关项目(96-B02)
收稿日期:2000-07-03;改回日期:2000-09-28

计方案.本文给出的空间查询优化系统 FQPro(Future Query Processor),是一个尚处于研究阶段的设计方案,它具有可扩展的体系结构,并采用结构化的空间查询语言 G/SQL[5,6].

1 优化过程概述

空间查询的优化过程大致分为如下 4 个阶段:

(1) 将查询转换为某种内部表示

其一般是将查询条件用语法树的形式表示.

(2) 进行某些提高效率的表达式变换

如,先进行选择再连接,先进行投影再连接等的变换,而且变换规则通常是有正确性保证的.

(3) 谓词代价计算

(4) 选择代价最小的执行计划

其中,前两个阶段与关系数据库优化的相应阶段类似[7],本文不做详细介绍;而谓词代价的计算与具体代价模型有关,另外,FQPro 系统允许多种索引结构共存,且每一种索引均对应自己的代价模型.目前 FQPro 系统的设计以基于 R-树的代价模型为实现重点.

FQPro 优化器主要进行语句中涉及空间数据成分的优化,而关系部分的优化仍然由底层关系数据库的优化模块进行.由于优化的主要工作是谓词顺序重组,且这种优化器假定关系型操作代价小于空间型操作,因此应遵循将关系型谓词尽可能提前的原则.另外,在决定两个或多个空间型谓词的顺序时,优化器应首先考虑所有可能的排列,然后选择代价最小的一个排列,若要进行代价比较,需事先进行代价估计,其基本谓词的代价估计可调用相应代价估计模块作代价分析,而基本谓词的合取范式的代价估计则不仅要考虑其中各个基本谓词的预计代价,还要考虑各个谓词的选择性,最后,优化器输出一个最佳执行方案.

2 代价估计模型

2.1 设计思想

为了有效地支持空间操作优化,FQPro 对查询处理器和优化器作了如下的空间扩展:

(1) 查询处理器能够对空间谓词进行代价估计.所谓空间谓词是包括空间比较符的谓词,或是包括空间操作符的谓词.

(2) 扩展代价模型中关于谓词选择性的部分,使之能够计算空间谓词的选择性.

(3) 用模块化实现代价估计,照顾多种空间数据组织形式,重点实现基于 R-树的代价模型.

(4) 在精确度和优化开销之间权衡.

由于代价估计模型与空间数据组织形式密切相关,因此基于已有空间组织形式即可给出若干代价模型,并应用到空间查询处理中去,这比研究更多的空间数据结构更有意义.目前常见空间数据的组织形式有 栅格文件,kd-树,四叉树家族,R-树家族等.其中,R-树家族是目前公认的效率比较高的空间索引,其可用的基于 R-树的高效空间操作算法也越来越多.鉴于此,FQPro 将设计的重点定位为基于 R-树的代价估计模型,同时,FQPro 的代价模型不限于某种数据组织形式,因为一方面,系统可提供除 R-树模型之外的若干模型;另一方面,系统允许用户根据需要,扩充代价模型,以便适应更多种数据组织形式.那么,是不是代价模型越精确越好呢?并非如此,因为模型精确度越高,优化所需的时间和空间代价越高.此外,保证精确度所必须的大量统计数据的实时维护会形成操作瓶颈,从而降低系统总体性能,如,关系数据库的优化模型就是一个不十分精确的模型[3],它是基于一些不完全符合现实的假设(如:索引键值平均分布,不同字段之间具有独立性等),以及一些并非实时维护的数据库统计信息(如:表的行数,索引最大最小值等).FQPro 优化系统中的代价模型同样不保证高精确度,而是力求在大多数情况下都能给出切实有效的优化方案.

2.2 基于 R-树的代价估计模型

由于窗口查询和叠置不仅是最重要的空间操作,而且其中叠置操作也是代价最大的空间操作,因此定义代价模型时,首先要考虑这两个操作;反之,如果一个代价模型能够准确估计这两个空间操作的代价,它就是一个比较好的模型.FQPro 采用雅典理工学院提出的一个模型,来估计窗口查询和空间叠置操作的代价[2,3].

该模型的分析过程是基于 R-树,但是最终代价公式只依赖于数据集本身的属性和查询条件,而操作代价则用操作中结点访问次数来衡量.该模型不考虑缓冲机制,这样,结点访问次数与磁盘访问次数成线性关系.实验数据证明,该模型估计的代价误差为 10% ~ 15%,因而具有实用价值.

2.2.1 R-树结构

R-树最早由 Guttman 于 1984 年提出,是 B^+-树在

二维空间的延伸[7].由于 R-树作为空间索引结构所具的高效性,从而已引起人们广泛的关注和研究,形成一个 R-树家族,其中,典型的有:

①压缩 R-树(packed R-tree),其适用于静态空间数据;

②R^+-树,能保证各个树结点互不相交;

③R^*-树,其采用比 R-树更为复杂的插入和结点分裂算法,从而能提高检索效率;

④希尔伯特 R-树,其采用分形维的 R-树.

R-树的非叶结点包括形如(ref,rect)的项,其中,ref 指向一个子结点,rect 为该子结点所有各项中矩形的最小外包矩形.虽然叶结点中包括同样形式的项,但是其中 ref 指向数据库中某个空间对象,而 rect 仅是该对象的最小外包矩形.

令 M 为一个结点所能容纳的最大项数,m 为结点所容许的最小项数,且 $2 \leq m \leq [M/2]$.另外,R-树具有如下特性:

(1) 根结点至少有两个子女,除非它是叶结点;

(2) 每个非叶结点有 m 到 M 个子女,除非它是根结点;

(3) 所有叶结点都在同一层上;

(4) 每个非叶结点的 rect 是其所有子女对应矩形的最小外包矩形.

R-树是完全动态的,即插入和删除操作可以和查询同步进行,而无须进行全局树结构的重组.图 1 是一个二维矩形数据集及其对应的 R-树索引的例子.

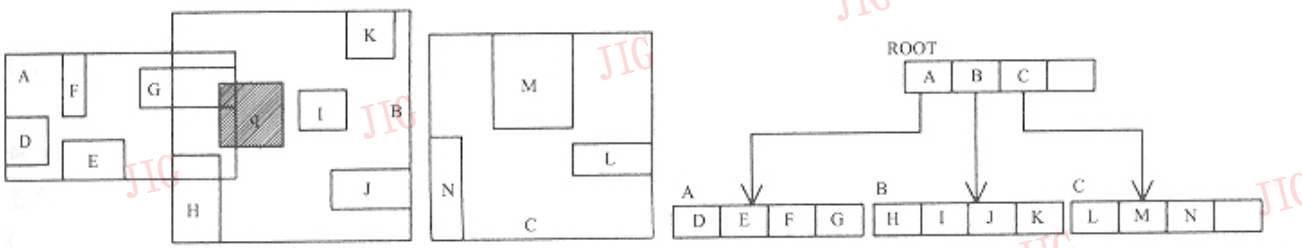

图 1　R-树索引

在对基于 R-树的代价模型进行分析时将使用到表 1 所列符号.

表 1　基于 R-树代价模型分析的使用符号

符号	说明
n	维度
N	数据集的秩,即集合包含的对象数
D	数据集密度
$q=(q_1,\cdots,q_n)$	查询窗口
m	R-树结点最小容量
M	R-树结点最大容量
F	平均结点容量,即 R-树的扇出
H	R-树高度
N_j	R-树第 j 层结点数
$s_j=(s_{j,1},\cdots,s_{j,n})$	R-树第 j 层平均结点大小
DA	空间操作中的磁盘访问次数
S	谓词的选择性

2.2.2　窗口查询代价模型

定义 1　窗口查询

给定 n-维单位工作空间 $W_S=(0,1)^n$,其中有 N 个数据对象,其平均大小 $s=(s_1,\cdots,s_n)$,查询窗口 $q=(q_1,\cdots,q_n)$,那么,窗口查询的定义即是求 N 个对象中落在 q 中的所有对象的数目.

定义 2　数据集密度 D

一组矩形对象,其平均大小为 s,则 D 定义为包含一个给定 n-维点的平均矩形对象数,D 也可以表示为对象面积总和与工作空间面积的比值.

给定 R-树参数:h,N 和 $s_{j,k}$,分别表示树结构的高度、树的结点总数,以及在每一维 k,每一层 j 的平均结点内容(假定根结点在 h 层,叶结点在 1 层).查询窗口 $q=(q_1,q_2,\cdots,q_n)$.若以结点访问次数表示估计执行代价,则估计执行代价 $NA(q)$ 由下式给出[9]

$$NA(q)=1+\sum_{j=1}^{h-1}\left\{N_j\cdot\prod_{i=1}^{n}(s_{j,i}+q_i)\right\} \tag{1}$$

R-树高度 h 由下式计算

$$h=1+\left[\log_f\frac{N}{f}\right] \tag{2}$$

其中,f 为平均结点数目,$[\cdot]$为四舍五入取整.

第 j 层的 N_j 为

$$N_j=\frac{N}{f^j} \tag{3}$$

第 j 层的平均结点大小(假定结点的外包矩形为正方形)为

$$s_{j,k}=\left(\frac{D_j}{N_j}\right)^{1/n} \tag{4}$$

其中,$D_0=D$,即数据集合的实际密度,D_j 表示第 j 层的结点矩形密度,是 D_{j-1} 的一个函数:

$$D_j = \left\{1 + \frac{D_{j-1}^{1/n} - 1}{f^{1/n}}\right\}^n \tag{5}$$

由此可以看出,上述代价模型在估算 n-维查询窗口 q 的区域查询代价时,只利用了有关数据集合本身的数据,即数据对象数 N 以及它们的最小外包矩形 MBR 的密度 D,以及查询窗口 q,而且窗口查询的选择性 S 也可以完全通过 N、D、s 和 q 表达.

式(1)~式(5)可以归约为:

$$NA = 1 + \sum_{j=1}^{1+\left[\log_f \frac{N}{f}\right]} \left\{\frac{N}{f^j} \cdot \prod_{i=1}^{n} \left(\left(D_j \cdot \frac{f^j}{N}\right)^{1/n} + q_i\right)\right\} \tag{6}$$

其中,$D_0 = D$,$D_{j+1} = \left\{1 + \frac{D_j^{1/n} - 1}{f^{1/n}}\right\}^n$

简化式(6),即得到窗口查询代价 NA 的近似公式

$$NA = O(N \cdot \log N) \tag{7}$$

定理 若给定 N 个矩形的集合,其平均大小为 s,以及查询窗口 q,则与 q 相交的矩形的平均数为

$$\text{intsect}(N, s, q) = N \cdot \prod_{i=1}^{n} (s_i + q_i) \tag{8}$$

显然,窗口查询的选择性 S,即结果集中对象数与总的对象数 N 之比,可通过下式计算

$$S = \frac{\text{intsect}(N, s, q)}{N} = \frac{N \cdot \prod_{i=1}^{n} (s_i + q_i)}{N} \Rightarrow \cdots \Rightarrow$$

$$S = \prod_{i=1}^{n} \left(\left(\frac{D}{N}\right)^{1/n} + q_i\right) \tag{9}$$

实际应用中,$D \ll N$,如一个中等规模的空间数据集,D 可能介于 0.1~0.3 之间,而 N 可能为10 000. 简化式(9),即得到如下 S 的近似公式

$$S = \prod_{i=1}^{n} q_i \tag{10}$$

2.2.3 空间叠置代价模型

叠置操作就是依据空间属性将两个数据集组合起来的操作,且这种叠置操作等价于一系列的窗口查询[3]. 这种参加叠置操作的两个集合中,一个可以看作是数据集合(内层遍历集合),另一个可以看作是查询窗口集合(外层遍历集合).

(1) 等高 R-树代价估算

R-树 R_1(R_2)高度为 h,在层次 j($1 \le j \le h-1$)上包含 $N_{R_1,j}$($N_{R_2,j}$)个结点,其结点平均大小为 $s_{R_1,j}$($s_{R_2,j}$),如叠置过程总代价为各个层次上的代价之和,而 R_1 和 R_2 在层次 j 上代价分别为 $NA(R_1, j)$ 和 $NA(R_2, j)$. 由于不考虑缓冲机制,因此遍历两个数据集 R-树的代价是相等的,即 $NA(R_1, j)$ 与 $NA(R_2, j)$ 相同,即

$$NA(R_i, j) = N_{R_1,j} \cdot N_{R_2,j} \cdot \prod_{k=1}^{n} \min\{1, (s_{R_1,j,k} + s_{R_2,j,k})\} \tag{11}$$

其总代价 $NA\_\text{total}(R_1, R_2)$ 为

$$NA\_\text{total}(R_1, R_2) = \sum_{j=1}^{h-1} \{NA(R_1, j) + NA(R_2, j)\} \tag{12}$$

其中,参数 h,$N_{R,j}$ 和 $s_{R_i,j,k}$ 分别由式(2)、式(3)和式(4)给出,式(12)给出的代价估算完全基于数据集合本身的对象个数 N_{R_i} 和密度 D_{R_i}. 代价估算公式相对于 R_1 和 R_2 是完全对称的.

简化式(12),即得到叠置查询代价 NA 的近似公式

$$NA = O(N_{R_1} \times \log N_{R_1}) \times O(N_{R_2} \times \log N_{R_2}) \tag{13}$$

(2)不等高 R-树代价估算

若 R_1 和 R_2 高度分别为 h_{R_1} 和 h_{R_2}. 且不失一般性,假定 $h_{R_1} > h_{R_2}$,则对于上面的 h_{R_2} 层结构,其叠置算法将不受高度不同的影响,此时 j 层($1 \le j \le h_{R_2}$)代价公式只需对式(12)稍加修改,因为现在 R_2 的层次不再等于 R_1 的层次,此时,R_2 的层次 j 用 j' 表示. 这样在处理 R_2 的叶结点时,$j' = 1$,而 R_1 的树遍历继续向下进行. 其代价估算公式如下

$$NA'\_\text{total}(R_1, R_2) = \sum_{j=1}^{h_{R_1}-1} \{NA(R_1, j) + NA(R_2, j')\} \tag{14}$$

其中

$$j' = \begin{cases} j - (h_{R_1} - h_{R_2}) & h_{R_1} - h_{R_2} + 1 \le j \le h_{R_1} - 1 \\ 1 & 1 \le j \le h_{R_1} - h_{R_2} \end{cases}$$

空间叠置操作的选择性将依具体操作而定,例如,常用的相离、外接、相等、交叠、包含、包含于、内接、内接于等 8 种拓扑关系[9],它们的选择性就有很大差异. 一般的,相离的选择性最低,外接、交叠、包含于、内接于的选择性较低,而相等、内接、包含的选择性则较高.

3 谓词代价

一般谓词的执行代价包括 CPU 代价和 I/O 代价,而且在基于 R-树做代价分析时,CPU 和 I/O 代价都与结点访问次数密切相关,其中 I/O 代价分为读取操作对象的 I/O(记为 $R_$COST)和输出结果的 I/O(记为 $W_$COST),而且可以近似地将 CPU 代价(记为 $CPU_$COST)和 $W_$COST 视为 $R_$COST 的线性函数,

即

$$\text{CPU\_ COST} = w \times R\text{\_ COST} \quad (15)$$

$$W\text{\_ COST} = S \times R\text{\_ COST} \quad (16)$$

其中,w 是一个经验值,S 为谓词选择性因子.这样,谓词的代价公式为

$$C_{\text{pred}} = (1 + w + S) \times R\text{\_ COST} \quad (17)$$

因此,基于 R-树的谓词代价的计算可以归结到R-树结点的读取次数计算上.

FQPro 支持如下 4 类空间谓词:

(1) 方向谓词:left_of,right_of,above,below

(2) 位置谓词:object_at_point

(3) 窗口查询谓词:in_window

(4) 叠置查询谓词:disjoint,meet,overlap,covers,covered_by,contain,contained_by,equal.

优化器计算谓词代价时需要查看系统表,以获取如表 2 的图层统计信息.

表 2　优化器计算谓词代价时需要获取的图层信息符号

符号	说明
N	空间对象数
D	数据集密度
$X_{\max}$	工作空间的上限 X
$X_{\min}$	工作空间的下限 X
$Y_{\max}$	工作空间的上限 Y
$Y_{\min}$	工作空间的下限 Y

3.1　谓词选择性因子

虽然利用图层统计信息,优化器即可赋予每个空间谓词一个选择性因子 S(该 S 是对结果集中对象数与源对象数比值的一个粗略估计),可是如果缺少需要的统计信息,则优化器将不经计算也可指定一个[0,1]上的分数作为谓词的选择性因子.

下面的谓词选择性列表中采用了表 3 所列的符号.

表 3　谓词选择性列表符号

符号	说明
OP_{SA}	空间代数操作符
GEOID	关系中的空间成分
column,column1,column2	关系字段
L,L_1,L_2	关系,或图层
N,N_1,N_2	关系中元组数,或图层中对象数
value,value1,value2	数值

谓词选择性列表:

(1) OP_{SA}(GEOID) = value,

或 column = value　等值谓词}

如果 N 已知,则 $S = 1/N$

如果 N 未知,则 $S = 1/10$

由于假定满足这一谓词的对象很少,因此选择 1/10 这样一个较小的分数.

(2) $OP_{SA}1(L_1.\text{GEOID}) = OP_{SA}2(L_2.\text{GEOID})$,

或 column1 = column2

如果 N_1、N_2 均为已知,则

$S = 1/\max(N_1, N_2)$

如果只有 N_i 已知,$i = 1$ 或 2,则 $S = 1/N_i$

否则

$S = 1/10$

(3) OP_{SA}(GEOID) > value,

或 column > value　开区域谓词}

$S = 1/3$

这里,1/3 只是一个粗略的估计,也可以选择介于 1/10 和 1/2 之间的其他分数.若大于 1/10,即本谓词选择性小于等值谓词;若小于 1/2,即假定查询很少会使用半数以上对象都符合条件的谓词.

(4) OP_{SA}(GEOID) between value1 and value2,

或 column between value1 and value2　区间谓词}

$S = 1/4$

这里,1/4 的选择使得本谓词的选择性小于等值谓词,而大于开区域谓词.

(5) OP_{SA}(GEOID) IN (value 列表),

或 column IN (value 列表)

$S = \min(1/2, \text{value 列表中的数值个数} \times \text{等值谓词选择性})$

本谓词选择性不大于 1/2

(6) GEOID left_of X_0

如果 $X_{\min}$、$X_{\max}$ 已知,则

$S = (X_0 - X_{\min})/(X_{\max} - X_{\min})$

否则 $S = 1/3$

(7) GEOID right_of X_0

如果 $X_{\min}$、$X_{\max}$ 已知,则

$S = (X_{\max} - X_0)/(X_{\max} - X_{\min})$

否则

$S = 1/3$

(8) GEOID above Y_0

如果 $Y_{\min}$、$Y_{\max}$ 已知,则

$S = (Y_{\max} - Y_0)/(Y_{\max} - Y_{\min})$

否则

$S = 1/3$

(9) GEOID below Y_0

如果 $Y_{\min}$、$Y_{\max}$ 已知,则

$S = (Y_0 - Y_{\min})/(Y_{\max} - Y_{\min})$

否则

$S = 1/3$

(10) object…at point

如果 N 已知.

$S = 1/N$

否则

$S = 1/10$

(11) in_window w

S 见谓词选择性列表(7)

(12) join

依具体叠置操作而定

(13) 谓词 1 OR 谓词 2

$S = S$(谓词 1) + S(谓词 2) - S(谓词 1) × S(谓词 2)

(14) 谓词 1 AND 谓词 2

$S = S$(谓词 1) × S(谓词 2)

(15) NOT 谓词 1

$S = 1 - S$(谓词 1)

3.2 谓词代价计算

FQPro 计算谓词代价分两步进行,其第 1 步,用代价公式进行粗略估算;第 2 步,加权细化.其中,第 1 步考虑操作本身及操作类别属性;第 2 步细化则利用了操作对象的属性和统计数据,以及操作本身区别于其所属类别的特性.由于两步相对独立,因此对其中任何一个实现模块的改进都不会影响到另一个.FQPro 目前设计的谓词代价公式比较粗糙,而对于空间操作的研究则日新月异,可以预计 FQPro 的优化器将逐步升级,并且代价计算的两步策略将有利于系统的性能提高.

由式(17)可知,空间操作代价取决于 $R\_COST(R)$,即结点访问次数.由于结点访问次数可以近似地用对象集中对象数 N 的阶来表示,因此,FQPro 的空间操作代价公式和谓词代价公式也可采用 N 来的阶近似表达.

空间操作的代价 C_{op} 列表如下,其中空间操作的具体说明参见文献[5,6].

表 4 空间操作的代价公式

操作	代价公式	操作	代价公式	操作	代价公式
GEOXCoor	$O(N)$	GEOYCoor	$O(N)$	GEOStartPnt	$O(N)$
GEOEndPnt	$O(N)$	GEOLength	$O(N)$	GEOSlope	$O(N)$
GEOArea	$O(N)$	GEOPerimeter	$O(N)$	GEOCentroid	$O(N)$
GEOInnerPnt	$O(N)$	GEOArcs	$O(N)$	GEOBuffer	$O(N)$
GEODistance	$O(N_1) + O(N_2)$	GEOOverlay	$O(N_1) + O(N_2)$	GEOMinus	$O(N_1) + O(N_2)$
GEOLeftCut	$O(N_1) + O(N_2)$	GEORightCut	$O(N_1) + O(N_2)$	GEOIntersect	$O(N_1) + O(N_2)$

空间谓词的代价公式 C_{pred} 列表如下:

(1) 含空间操作运算的数值比较谓词,谓词代价等于其中所有空间操作的代价之和

这样的谓词包括:OP_{SA}(GEOID) = value, $OP_{SA}1$(L_1.GEOID) = $OP_{SA}2$(L_2.GEOID), OP_{SA}(GEOID) > value, OP_{SA}(GEOID) between value1 and value2, OP_{SA}(GEOID) IN (value 列表).

(2) 方向谓词的代价为 $O(N)$

方向谓词包括:left_of, right_of, above, below.

(3) 谓词 object_at_point 的代价为 $O(\log N)$

(4) 窗口查询谓词的代价为 $O(N \cdot \log N)$

(5) 叠置查询谓词的代价为 $O(N_1 \cdot \log N_1) \cdot O(N_2 \cdot \log N_2)$

(6) 谓词 1OR 谓词 2

C_{pred} = C_{pred}(谓词 1) + C_{pred}(谓词 2)

(7) 谓词 1AND 谓词 2

C_{pred} = C_{pred}(谓词 1) + C_{pred}(谓词 2)

(8) NOT 谓词 1

C_{pred} = C_{pred}(谓词 1)

考虑到点、线、多边形等不同对象集合由于具有不同的复杂度,因此可以赋予空间操作不同的加权因子,如,点集合上空间操作的加权因子为 1,若设线对象集合的线上平均结点数为 N_1,则线集合上的空间操作加权因子为 N_1;若设多边形集合的平均边数为 N_2,则多边形集合上空间操作的加权因子为 $N_1 \times N_2$.这种加权因子可以通过对数据集的采样来计算.

虽然 8 种叠置查询谓词的代价使用同一个谓词代价计算公式,但是 8 种叠置谓词的查询代价还是有相当差别的.雅典国家理工大学、美国 UCSD 大学和缅因大学的联合研究结果表明[10],8 种叠置谓词根据代价差异分为 3 类,第 1 类包括一个谓词 disjoint;第 2 类包括:equal, cover, contain;第 3 类包括:meet, overlap, contained_by, covered_by.3 类的加权值与 D、s 有关.实现中,如果精确度要求高可以使用经验公式,否则可以使用固定的加权值.

4 查询方案代价计算

查询条件树可表示为简单谓词的合取范式,其中既有关系型谓词,也有空间型谓词,而且 FQPro 的优化器总是将所有关系型谓词放在所有空间型谓词的

前面,即关系型优先原则,且谓词顺序重排只考虑空间谓词,查询条件的基本元素是简单谓词(其代价计算在上一节已作了讨论).这里记空间型谓词为 P_S(下标 S 表示空间型 Spatial),考虑合取范式 $Cond = P_{S_1}$ and P_{S_2} and … and P_{S_n},记 $Cond1 = (P_{S_1}$ and P_{S_2} and … and $P_{S_{n-1}})$,则 $Cond = Cond1$ and P_{S_n}.计算谓词代价时,必须考虑到前面谓词(关系型或空间型)的执行对对象集合的改变,例如计算 P_{Sn}的代价,因为在此之前可能存在一个或多个关系型谓词的执行,然后又执行了 Cond1,其执行可能对某些对象集合作了筛选,而且这些作了筛选的对象集合将作为 P_{Sn}的输入,由于不是原数据库中的对象集合,因此,P_{Sn}的代价计算公式必须做某些修正.在实现中,优化器必须追踪对象集合秩的变化,在代价公式中应使用新的秩 N.这样,查询条件代价计算公式可以递归定义如下:

$$C_{cond}(P_S) = C'(C_{P_S}, Sprev\_rel, Relations\_in\_prev\_Rel \cap Relations\_in\_Cond(P_S)),$$

$$C_{cond}(Cond) = C_{cond}(Cond1) + C'(C_{P_{S_n}}, S_1, Relations\_in\_Cond(Cond1) \cap Relations\_in\_Cond(P_{S_n}))$$

其中,C' 有 3 个参数,分别表示谓词代价计算公式 C_{pred},用于修正公式 C_{pred}中的某些关系的 N 的选择性 S,以及需要对其 N 进行修正的对象集的集合.这种需要修正的对象集即为前后两部分作用对象的交集.Relations_in_prev_Rel 给出前面关系型谓词涉及的对象集合,Relations_in_Cond(Cond1)则用于计算 Cond1 涉及的对象集合.

下面为合取范式代价计算算法:

```
procedure Reordering_Algorithm
{
    for i:=2 to n do {
        for all S⊆ {P_S1,…,P_Sn} s.t. ‖S‖ = i do {
            bestPlan:= 一个具有无穷大代价的哑计划
            for all P_Sj, S_j, s.t. {P_Sj} ∪ S_j = S and {P_Sj} ∩ S_j = Φ
                do {
                P:= joinPlan(optPlan(S_j), P_Sj)
                If cost(P) < cost(bestPlan)
                    bestPlan:= P
            }
            optPlan(S) = bestPlan
        }
    }
    return(optPlan({P_S1,…,P_Sn}))
}
```

算法的输入是空间谓词 $\{P_{S_1},\cdots,P_{S_n}\}$的一个合取范式.运算时,算法对其进行顺序重组,并将合取范式经过顺序重组后的任何一个变形称为一个优化计划,简称计划.然后函数 JoinPlan(p,P_S)通过合取 p 和 P_S,来扩展原计划 p 即得到一个新计划,接着函数 cost(p)返回计划 p 的代价.算法通过逐步计算条件树中简单谓词子集的最小代价,使子集逐步增大,直到包括所有简单谓词,从而得到整个查询条件的最小代价.为了计算包括(i + 1)个简单谓词的 S 的最佳计划,需将 S 每一个包括 i 个简单谓词的子集都进行扩展,其中代价最小的计划即被选中.这种方法保证了结果的最优性,因为它满足最优律,即,一个集合的最优计划必然是该集合某个子集最优计划的一个扩展.

下面给出一个优化示例.

查询:在北京市区图(Beijing)上查找所有三环以内、二环以外的宾馆.通过以下操作即可查询:

```
G/SQL:select c.name, c.GEOID
      from Beijing c
      where c.class = "Hotel" and in_window(w3) and not
      in_window(w2);
```

其中,w_3 为三环窗口,w_2 为二环窗口.

查询中,有一个关系型谓词和两个空间型谓词.根据关系型优先原则,首先执行关系型谓词操作.虽然由窗口查询代价公式可知,三环窗口选择和二环窗口选择的代价相等,但由于窗口查询选择性公式为窗口各个维度上长度之积,所以三环选择性小于二环选择性.经过简单计算可知,二环选择应该先于三环选择执行.所以最优计划中谓词执行顺序为

(1) 关系谓词 c.class = "Hotel"

(2) 空间谓词 not in_window(w_2)

(3) 空间谓词 in_window(w_3)

另外,优化对效率的提高有较大影响.如假设三环长、宽约为二环长、宽的 2 倍,则三环窗口选择性在数值上是二环窗口选择性的 4 倍,则通过简单计算查询方案代价可知,优化前后效率相差超过 4 倍.

5　查询优化展望

在过去的 10a 里,空间查询是研究的热点领域,其中从过程化的空间查询到结构化的空间查询语言,成绩是有目共睹的,但在空间查询的优化方面,还面临很多挑战.

本文提出的 FQPro 系统是对空间查询优化的尝

试,它还存在很多需要改进的地方,譬如,空间代价模型需要进一步完善.由于系统的谓词代价近似公式仅为空间对象总数的某个表达式的阶,且这些粗略的公式使得一些代价上有相当差别的谓词没能区分出来,因此可行的解决方案是 在近似公式之上进行加权细化,或者研究新的代价模型.

由于查询优化的主要目标是提高查询处理的速度,因此应该使空间处理真正能够满足实际应用的需求,可是因为实际应用可能对数据表示和访问方式有不同的要求,即使同一种应用也可能随着时间的变化而提出新的要求,如变换一种更有效的索引方式,或是增加一种操作.由此可见,系统方案应该将可扩展性列为重要的设计目标,换言之,系统必须适应新的索引结构、新的空间操作和新的空间算法.

参 考 文 献

1 Codd E F. A relational model of data for large shared data banks. Comm. Of the ACM, June, 1970.

2 Theodoridis Y, Stefanakis E, Sellis T. Cost models for join queries in spatial databases. IEEE Trans. On Knowledge and Data Engineering, 1998.

3 Papadias D, Theodoridis Y, Sellis T *et al*. Topological relations in the world of minimum bounding rectangles: A study with R-trees. In: Proc. Of ACM SIGMOD Int'l Conf. on Management of Data, 1995.

4 Brinkhoff T, Kriegel H, Seeger B. Efficient processing of spatial joins using R-trees. In: Proc. Of ACM SIGMOD Int'l Conf. on Management of Data, 1993.

5 方裕,楚放,陈斌.空间结构化查询语言 G/SQL.中国图象图形学报,1999,4(11):901910.

6 Walid G Aref, Hanan Samet. Optimization strategies for spatial query processing. In: 17th Int'l Conf. On Very Large Data Bases(VLDB), Spain, Madrid, Sep. 1991.

7 Guttman A. R-trees: A dynamic index structure for spatial searching. In: Proc. Of ACM SIGMOD Int'l Conf. on Management of Data, 1984.

8 Kim W, Garza J, Keskin A. Spatial data management in database systems: Research directions. In 3rd Symposium on Large Spatial Databases, Zurich, Switzerland, Aug. 1995.

9 Date C J. An introduction to database systems. New York: Addison-Wesley Publishing Company, 1994.

10 Selinger P G, Astrahan M M, Chamberlin D D *et al*. Access path selection in a relational database management system. In Proc. Of the ACM SIGMOD Int'l Conf. On Management of Data, Boston, 1979.

方 裕 1946 年生,1968 年毕业于北京大学,现为北京大学计算机系教授。长期从事计算机软件工程、地理信息系统、计算机集成制造系统等方面的研究工作。

楚 放 1975 年生,北京大学计算机系硕士研究生。主要研究方向为地理信息系统、软件工程。