基于 R 树的空间查询连接处理优化与实现

吕闽晖[1],吕敏蓉[2]
(1.海军工程大学 装备经济研究所,湖北 武汉 430033;
2.湖南女子学院,湖南 长沙 410004)

摘 要:空间索引作为空间数据库的关键技术,其性能的高低决定着整个空间数据库的效率。通过对现有的多种空间索引结构进行比较分析,基于开源数据库 Ingres 实现了广度优先 R 树连接算法(BFRJ),并对其进行了局部优化和全局优化。基于真实数据的实验结果分析,证实了采用适当的全局优化方法的 BFRJ 优于其他已知的空间连接算法方法。

关键词:空间索引;空间连接;Hilbert R 树;BFRJ
中图分类号:G304 **文献标识码**:A **文章编号**:1674-7720(2011)13-0066-03

Optimization and implementation of R tree-based processing on spatial join query

Lv Minhui[1], Lv Minrong[2]
(1.Institute of Equipment Economics, Naval University of Engineering, Wuhan 430033, China;
2.Hunan Women College, Changsha 410004, China)

Abstract: As the key technique of spatial database, the performance of spatial index determines the efficiency of the entire database. This paper works on the open-source database named Ingres. Through a comparative analysis of existing spatial join structures, we propose to implement the Hilbert-R-tree. Then, we implement the Bread-First R-tree Join Algorithm (BFRJ), and provide local and global optimazition. At last, through a series of experiments based on the real world data sets, we comfirms that, with the appropriate choice of global optimazition, the BFRJ is better than other spatial join algorithms.

Key words: spatial index; spatial join; Hilbert R-tree; BFRJ

常见的空间查询有点查询、窗口查询(或称范围查询)、相交查询(或称区域查询)、被包围查询、毗连查询、最近邻查询、空间连接查询等,其中空间连接查询是最重要、最耗时的空间查询[1]。本文首先对 Ingres 原有空间连接方式进行研究。然后参照已有的相关研究论文,采用基于宽度优先的 R 树空间连接算法,并对其进行全面测试(包括功能和性能测试),并与 Ingres 原有空间连接进行性能对比。

1 Ingres 空间连接算法

Ingres 原有的空间连接执行时,如果两个表都建有 R 树索引,其 QEP 为 TID Join。如有两个表 RTreeJoin1 和 RTreeJoin2,对应的空间索引为 RTree1 和 RTree2,如果执行 Select*from RTreeJoin1,RTreeJoin2 where RTreeJoin1.obj overlaps RTreeJoin2.obj,QEP 如图 1 所示。

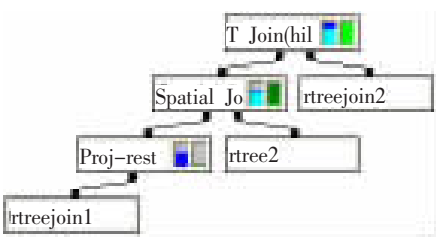

图 1 Ingres 原有空间连接 QEP

图中 Spatial Join 即 Key Join,首先是将 RTreeJoin1.obj 的值作为键值查找索引 RTree2,所得对应 RTreeJoin2 中的 TID,然后再以此 TID 直接访问 RTreeJoin2 中的元组,即 TID Join。此种方式没有利用两个基表都建有 RTree 的优势,因此在部分情况下不能获得最优的性能。这种方法称为“One-Rtree-Only”空间连接方法[2]。

2 Ingres 空间连接算法改进

针对上述分析，希望能够在过滤阶段即利用两个 R 树索引。通过对两个索引进行按深度或按宽度的遍历，执行过滤运算，不可能满足连接条件的空间对象迅速排除[3-4]。本文将参考文献[5]使用广度优先遍历算法优化 R 树空间连接。

2.1 在遍历 R 树时剪枝

在 R 树中维护的一个重要信息就是它的层次性蕴含着它的包含性。即一个树结点的 MBR 总是包围着它的子孙结点的 MBR。利用这一特点可知，一对结点 ${}_nR_i^r$ 和 ${}_nS_j^s$ 需要进行连接仅当它们父结点的 MBR 相交，称为剪枝。简单的从顶至底的图遍历算法可以在任何层次上使用这种剪枝。参考文献[5]中剪枝是在同时深度优先遍历两棵输入的 R 树时进行的 (DFRJ)，而参考文献[6]则是在同时广度优先遍历两棵 R 树时进行的(BFRJ)。在 R 树的所有层次施行剪枝达到的效果是从顶层起，分别来自两棵 R 树的两个结点被遍历到仅当它们父结点的 MBR 相交。这样，相比简单的嵌套循环的遍历方式，剪枝将使得被遍历的不相交的结点对数大大减少。

2.2 BFRJ 算法

基于广度优先遍历的 R 树空间连接算法 (BFRJ)步骤为：(1)对两棵 R 树的根结点(N_R, N_S)做一次结点连接。所谓结点连接，就是对输入的两个非叶子结点，返回其相交的元素项对。对根结点做结点连接的结果是一个二元组(oidR0，oidS0)的集合，称为 0 阶中间连接索引 IJI_0 (intermediate join index at level 0)。因为本文主要着眼于对相交运算的空间连接，因此每一个二元组(oidR0，oidS0)意味着这两个元素项的 MBR 相交。(2)对于 IJI0 的每一个二元组，BFRJ 找出其在两棵 R 树中对应的结点，进行结点连接。当 BFRJ 从 IJI0 中读入一个二元组进行计算时，它会以二元组(oidR1，oidS1)的形式保存结果，加入当前的 1 阶中间连接索引 IJI1。当 BFRJ 处理完 IJI0 中的所有二元组后，它会释放 IJI0 并开始对 IJI1 进行相应的连接操作。这个过程随着 BFRJ 算法逐层遍历 R 树而继续着。当中间连接索引由两棵 R 树的叶子结点产生时，算法终止。如果两棵 R 树不等高，算法将在到达一棵树的叶子结点(如 R)时，枚举叶子结点中的每一个元素项对另一棵树 S 中对应子树做查询操作。到此，空间连接过滤环节已经结束，当前的(叶子级的)IJI 已经不在空间连接的范畴之内了。

2.3 局部优化技术

局部优化技术是在结点连接的过程中进行的。在参考文献[6]中介绍了两种局部优化技术，分别为搜索空间限制(Search Space Restriction)和平面扫描(Plane Sweep)。

2.3.1 搜索空间限制

在一对结点 ${}_nR_i^r$ 和 ${}_nS_j^s$ 进行连接时，它们的 MBR 的交叉区域仍然是 MBR(称为 intersect-MBR)。如果 ${}_nR_i^r$ 中的元素项$(oidR_i^r, mrbR_i^r)_x$ 同 ${}_nS_j^s$ 中的元素项$(oidS_j^s, mrbS_j^s)_y$ 相交，则 $mrbR_i^r$ 和 $mrbS_j^s$ 必须同两个结点的 intersect-MBR 相交。因此，可以首先对 ${}_nR_i^r$ 和 ${}_nS_j^s$ 的元素项分别进行扫描，排除掉那些 MBR 同 intersect-MBR 不相交的元素项。在进行结点连接时，仅对剩余项进行连接。

2.3.2 平面扫描

这一优化方式同合并两个普通的数据集合时采用的排序-归并技术类似。在一对结点 ${}_nR_i^r$ 和 ${}_nS_j^s$ 进行连接的过程中，首先分别对两个结点中的元素项依 MBR 进行排序。为了对多维数据进行排序，用 MBR 的最小 x 值作为关键字。在合并阶段，依次对排序的 MBR 进行扫描。对于一棵树中的 MBR，仅对于其 x 坐标相交的 MBR 进行相交测试。

这两种优化技术各有侧重，搜索空间限制技术可以对于结点容量较大但元素项分散的数据进行较大优化，而平面扫描对于多维数据的优势更大。

2.4 全局优化技术

在 BFRJ 算法框架下，i 阶中间连接索引(IJI_i)在两棵树中所有的 i 层结点连接完毕后产生。而在 i 层进行结点连接的结点对来自于由更高的 ($i-1$ 层) 结点连接产生。因此可以在对一层结点进行结点连接之前，获取该层所有结点访问预计(包括它们可能的访问顺序和重复访问次数)的全局信息，可以利用一些技术来对中间连接索引进行高效的管理。

2.4.1 中间连接索引排序

设结点的 MBR 同 R 树 S 的 k 个 t 级结点的 MBR 相交，则 ${}_nR_i^t$ 的索引 ID 在 IJIt-1 中出现了 k 次。在计算 t 层的结点连接时，${}_nR_i^t$ 将被恰好计算 k 次。对于一个固定大小的 LRU 系统缓冲，如果 ID 在 IJIt-1 中出现的比较分散，则结点 ${}_nR_i^t$ 将从硬盘中被多次读取(最多 k 次)。这是因为第一次和后面出现的对 ${}_nR_i^t$ 的调用之间可能会比较远，在需要被再次读入时可能已经从缓冲区中删除。因此希望 IJI 能够维护在一种有序的状态下，使得多次出现的相同结点的 ID 在 IJI 中出现的位置不要分散得太过稀疏。

2.4.2 中间连接索引内存管理

IJI 可以被存储在内存中或是磁盘上。前者能提高 IJI 排序的效率并减少读取 IJI 时多余的 I/O 操作；而后者能解放出更多的内存资源用于连接计算。

2.4.3 中间连接索引缓冲管理

IJI 排序技术倾向于维护索引的顺序，使得任意两个相同 ID 出现的位置不会相隔太远。然而，对两个项目都实现很好的聚类效果是不可能的，在连接运算中，对一个结点的多次磁盘读取依然存在。如果缓冲管理可以预测哪个结点已经连接完毕，而哪个结点将来还会参与运算，则这种多次读取可以被进一步缩减。用这种方式，缓冲管理可以保留那些将来还会参与运算的结点页面，而将已经结束所有连接运算的结点页面清理释放。

3 实验结果

采用实际数据进行实验。实验数据来自 www.rtreeportal.org 的两个数据组：

(1)“Germany”数据组，包含四个数据包：

①“roads”：包含了德国的 30 674 条街道的 MBR；

②“rrlines”：包含了 36 334 条铁路线的 MBR；

③“utility”：包含了 17 790 条公用网络的 MBR；

④“hypsogr”：地势图，由 76 999 个 MBR 组成。

(2)“Greece”数据组，包含两个数据包：

①“roads”：包含了希腊的 23 268 条街道的 MBR；

②“rivers”：包含了 24 650 条河流的 MBR。

将这些数据包组成四个连接对进行实验，即“Germany：roads，rrlines”，“Germany：roads，utility”，“Germany：roads，hypsogr”，“Greece：roads，rivers”。

在实验中，固定了 R 树的页面大小，这样，影响连接性能的主要参数除了使用的方法外即为缓冲大小。用缓冲区最多存放 R 树页面的个数作为衡量缓冲大小的参数，这样在不失一般性的同时可以简化缓冲管理操作在程序内实现。

表 1~表 4 给出了连接对“Germany：roads，rrlines”的实验结果，其中的数据表示在给定的缓冲大小下给定空间连接方法在给定缓冲大小下对应的页面 I/O 次数 m。

表 1 连接对“Germany：roads，rrlines”实验结果

Buff size	One-Rree-Only	DFRJ	OrdOneStor DiskPinNo	OrdOneStor MemPinYes	OrdSumStor MemPinYes
1 000	406 381	156 034	74 739	N/A	N/A
2 000	406 373	146 654	74 011	N/A	N/A
4 000	406 373	144 478	73 178	N/A	N/A
6 000	406 373	142 463	66 289	22 982	26 516
8 000	406 373	137 063	63 059	20 857	23 124
10 000	406 373	132 281	61 101	19 207	18 970
12 000	406 373	131 661	55 060	19 207	18 970

表 2 连接对“Germany：roads，utility”实验结果

Buff size	One-Rree-Only	DFRJ	OrdOneStor DiskPinNo	OrdOneStor MemPinYes	OrdSumStor MemPinYes
1 000	332 302	206 569	31 678	N/A	N/A
2 000	332 302	158 600	31 607	N/A	N/A
4 000	216 754	64 110	30 309	N/A	N/A
6 000	216 754	57 694	28 368	7 660	8 838
8 000	216 754	54 768	27 839	6 952	7 708
10 000	216 754	51 695	18 525	6 422	6 353
12 000	216 754	51 695	18 525	6 401	6 322

表 3 连接对“Germany：roads，hypsogr”实验结果

Buff size	One-Rree-Only	DFRJ	OrdOneStor DiskPinNo	OrdOneStor MemPinYes	OrdSumStor MemPinYes
1 000	795 631	202 764	105 742	N/A	N/A
2 000	795 613	196 070	103 414	N/A	N/A
4 000	795 607	189 695	102 946	N/A	N/A
6 000	795 607	188 861	102 293	N/A	N/A
8 000	365 784	184 357	95 582	18 306	16 113
10 000	365 784	183 693	93 892	16 613	16 113
12 000	365 784	182 086	91 722	16 613	16 113

表 4 连接对“Greece：rivers，roads”实验结果

Buff size	One-Rree-Only	DFRJ	OrdOneStor DiskPinNo	OrdOneStor MemPinYes	OrdSumStor MemPinYes
1 000	481 321	197 986	72 232	N/A	N/A
2 000	481 321	168 818	71 745	N/A	N/A
4 000	481 321	142 304	69 585	N/A	N/A
6 000	481 321	139 365	64 287	10 759	17 634
8 000	481 321	135 481	63 974	7 961	6 613
10 000	481 321	130 196	63 974	7 366	6 613
12 000	481 321	130 196	63 974	7 366	6 613

从实验结果分析各连接方法，Buff size 表示缓冲区大小，One-Rree-Only 表示只对一个数据包建 R 树；DFRJ 表示使用 DFRJ 方法进行 R 树连接；OrdOneStorDiskPinNo 表示采用基于磁盘存储的非特定排序优化组合 BFRJ 方法进行空间连接；OrdOneStorMemPinYes 表示采用基于主存存储的对单棵树排序优化组合 BFRJ 方法进行空间连接，OrdSumStorMemPinYes 表示采用基于主存存储的对中值和排序的优化组合 BFRJ 方法进行空间连接。由于 StorMem 意味着要将 IJI 保存在主存中，因此对缓冲大小有要求，当缓冲较小时则不适用，对应的表格项为 N/A。

在真实数据的条件下，对于任意的缓冲大小，基于两棵 R 树的空间连接算法(DFRJ 和 BFRJ)的性能总是优于基于一棵 R 树的算法(One-Rree-Only)。而在 R 树空间连接方法中，使用适当全局优化组合的 BFRJ 又明显优于 DFRJ。在缓冲较小时，OrdOneStorDiskPinNo 最适用；在适中的缓冲条件下，StorDisk 的性能要比 StorMem 差，OrdOneStorMemPinYes 相对更优；而在较大的缓冲条件下，OrdSumStorMemPinYes 能够获得最好的空间连接性能。

参考文献

[1] KAMEL I, FALOUTSOS C. Hilbert R-tree: An improved R-tree using fractals [M]. In: Proceedings of the 20th VLDB, Santiago, Chile, 1994,500-509.

[2] HUANG P W, LIN P L, LIN H Y. Optimizing storage utilization in R-tree dynamic index structure for spatial databases[J]. Journal of Systems and Software, 2001,55(3):291-299.

[3] BRAKATSOULAS S, PFOSER D, THEODORIDIS Y. Revisiting R-tree construction principles [M]. In: Proceedings of the 6th ADBIS, Bratislava, Slovakia, 2002,149-162.

[4] BRINKHOFF T, KRIEGEL H. P, SEEGER B. Efficient processing of spatial joins using R-trees[M]. In: Proceedings of ACM SIGMOD, Washington DC, 1993,237-246.

[5] MARTYNOV M. Spatial joins and R-trees. In: Proceedings of the 3rd ADBIS[M], Moscow, Russia, 1995,295-304.

[6] HUANG Y W, JING N, RUNDENSTEINER E. Spatial joins using R-trees: Breadth First traversal with global optimizations. In: Proceedings of the 23rd VLDB, Athens, Greece, 1997,396-405.

(收稿日期：2011-01-05)

作者简介：

吕闽晖，男，1975 年生，硕士，讲师，主要研究方向：软件工程、装备经济信息管理。

吕敏蓉，女，1979 年生，硕士，讲师，主要研究方向：财务信息化管理、数据库应用。