第 33 卷 第 6 期 计 算 机 工 程 2007 年 3 月
Vol.33 No.6 Computer Engineering March 2007
·软件技术与数据库· 文章编号：1000—3428(2007)—06—0092—03 文献标识码：A 中图分类号：TP311.13

GML 空间数据索引机制研究

兰小机[1]，刘德儿[1]，闾国年[2]
(1. 江西理工大学环境与建筑工程学院，赣州 341000；2. 南京师范大学地理信息科学江苏省重点实验室，南京 210097)

摘 要：随着 GML 规范的不断完善及 GIS 软件厂商的广泛支持，GML 已经成为事实上的空间数据编码、传输、存储、发布的国际标准，越来越多的空间数据开始以 GML 格式存储。如何有效地存储管理 GML 空间数据，已经成为 GIS 研究的热点问题。结合 XML 文档编码和传统的空间数据索引，对 GML 空间数据的索引进行了深入的研究，提出了基于空间索引的 GML 一体化索引机制以及相应的查询处理策略与相关算法，并以 R 树索引为例，对一体化索引的查询处理性能进行了实验分析。实验结果表明，该文提出的基于空间索引的 GML 一体化索引机制是可行的、高效的。

关键词：GML；GML 数据索引；空间索引；XML

Research on Index Techniques for GML Spatial Data

LAN Xiaoji[1], LIU Deer[1], LU Guonian[2]
(1. College of Environmental and Architecture Engineering, Jiangxi University of Science and Technology, Ganzhou 341000;
2. Jiangsu Provincial Key Lab of GIS Science, Nanjing Normal University, Nanjing 210097)

【Abstract】 With the development of GML specification and its broad supporting by many GIS software vendors, GML becomes the de facto international standard for spatial data encoding, transmission, store and publish; more and more spatial data is stored in GML format. Issues as how to manage the spatial data in GML format efficiently become the hot ones in GIS. Integrating XML document encoding techniques and traditional spatial index methods, this paper conducts a deep research on indexing GML data; one unified indexing model based on spatial index for GML data is suggested; and its query processing strategies and related algorithms are discussed. Taking R-tree as an example, the performance of the unified indexing model is tested and analyzed through a set of experiments. Experimental results show that the unified indexing model is feasible and efficient.

【Key words】 GML; GML data index; Spatial index; XML

1 概述

GML是开放式地理信息系统协会 (Open Geospatial Consortium, OGC)制定的基于XML的中立于任何厂商、任何平台的地理信息编码规范，用于地理信息的传输、存储和发布[1]。GML已经成为事实上的空间数据编码、传输、存储、发布的国际标准，随着互联网技术的不断发展及网络GIS的广泛应用，GML格式的空间数据已经开始大量涌现，如何有效地存储、查询GML空间数据，还有很多不确定性和问题需要研究。

传统的关系数据库查询语言SQL是针对平面的二维关系数据而设计的，并不适合XML/GML半结构化数据的查询；商品化GIS软件的查询系统都是针对自身的封闭数据模型和数据结构而设计的，只能查询自身的空间数据而无法查询其它GIS系统的空间数据；XML查询的研究为GML查询奠定了一定的基础。对XML查询的研究，国内外学者提出了多种XML查询语言，比较著名的有Lorel、XML-QL、XQL、Quilt等。为了规范XML查询语言，W3C于 1999 年 9 月正式成立了XQuery工作组，先后颁布了多个XQuery草案，还在不断的修订和完善之中[2]。XQuery将成为XML查询语言标准，扩展XML查询语言XQuery是GML查询实现的最佳选择。

国内外对 GML 查询、索引的研究较少，Corcoles 等人在文献[3]中提出基于 SQL 的 GML 空间查询语言，这种处理方式与 XML 查询语言标准 XQuery 不相符。Vatsavai 在文献[4]中比较了几种 XML 查询语言，并提出了对 XQuery 语言进行扩展以支持 GML 查询的设想，但文中并没有涉及如何实现、GML 索引等更深层次的问题。由南京师范大学承担的国家自然科学基金项目“ GML 空间数据存储索引机制研究(编号：40401045) ”正在对 GML 存储管理、查询、索引、集成等问题进行研究。

为了提高 GML 查询处理效率，GML 数据索引是关键。本文结合 XML 文档编码和传统的空间数据索引，基于 GML 本原查询(以 XQuery 为基础进行扩展)的思想，对 GML 空间数据的索引进行研究。

2 GML 一体化索引的基本思想

GML 数据是基于 XML 格式的文本数据，它既具有普通 XML 数据的特征，又具有空间数据的特点，既包括普通的属性数据，也包括空间定位数据。为了提高 XML 数据的查询处理效率，可以先对 XML 文档进行编码，如先序-后序编码，XML 文档编码之后，文档中的元素、属性就转换为二维空间

基金项目：国家自然科学基金资助项目(40401045)；地理信息科学江苏省重点实验室开放基金资助项目(JK20050302)；江西省教育厅科技项目“ GML空间数据库关键技术及应用研究 ”
作者简介：兰小机(1965－)，男，博士、副教授，主研方向：GML 空间数据库，空间数据共享与互操作等；刘德儿，硕士、讲师；闾国年，教授、博导
收稿日期：2006-03-19 **E-mail：**landcom8835@163.com

中的点，既然是空间数据，自然就可以使用空间索引来加速其查询检索。基于这点考虑，本文提出基于空间索引的 GML 一体化索引的思想，即属性数据、空间定位数据都使用传统的空间索引。

2.1 基于扩展的前序-后序 GML 文档编码

文献[5,6] 提出利用前序-后序(preorder，postorder)对 XML 文档树中的结点进行编码。对于树中的任意两个节点 u 和 v，如果 preorder(u)<preorder(v)，且 postorder(u)>postorder(v)，则 u 是 v 的祖先节点。

这种编码方法的优点是能够在常数复杂度的时间内实现任意两个结点间祖先-后代关系的判断。缺点是缺乏灵活性，当一个新的结点插入时，许多结点的前序、后序值都要重新遍历计算。针对这种前序-后序编码存在的缺陷，本文提出扩展的前序-后序编码方案，基本思想是相邻的前序(或后序)留有一定的间隔，即假如相邻前序(或后序)为 i、j，j=i+k，k≥1。k=1 时为普通的前序-后序编码。k 为常整数或由 GML 应用模式 maxOccurs 属性或 GML 文档的统计信息确定。

对于 XML/GML 文档中的任一节点 v，其编码形式为(DocID(v)，preorder(v)，postorder(v)，level(v)，Type(v))，其中 DocID 为 XML/GML 的文档编号，preorder、postorder 为节点 v 的前序、后序遍历值，level 为节点 v 所在的层，Type 为节点 v 的类型(元素节点、属性节点或文本节点等)。

2.2 XPath 的关键轴和前序-后序平面分割特性

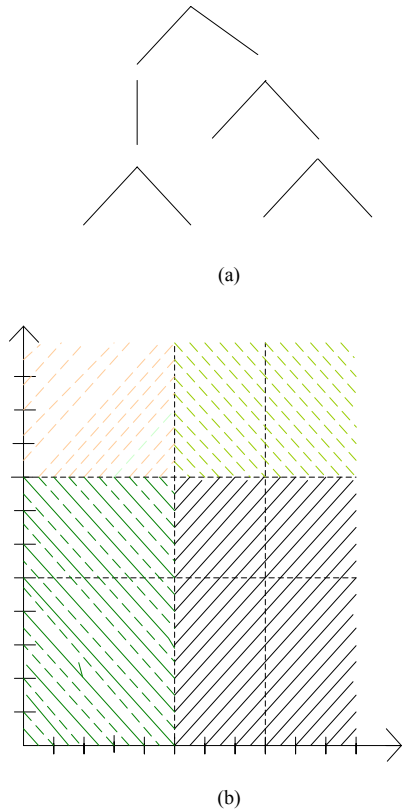

图 1 前序-后序平面分割特性

XPath 是 XQuery 的重要组成部分，是 XML/GML 文档树导航查询的基本工具，支持丰富的路径查询功能。XPath 有轴 13 个，其中 ancestor、descendant、following、preceding 以及 self 5 个轴合称为关键轴，它们将一个 XML/GML 文档划分为互不重叠的 5 部分(4 个文档区域，1 个上下文节点)，而且它们组合在一起则包含了这个 XML/GML 文档的所有节点。如果一个 XML/GML 文档采用扩展的前序-后序编码方案进行编码，得到各节点的编码值(preorder, postorder)，它们对应于前序-后序平面(以下简称 preorder/postorder 平面)中的点。图 1 为一简单的 XML/GML 文档各节点(preorder, postorder)分布图。从图中容易看出：对于 XML/GML 文档的任一节点 v，在 preorder/postorder 平面中过点 v 分别作 preorder、postorder 轴的垂线，将 preorder/postorder 平面分为 4 个子区域，其中：左上角区域 R 包含 v 的所有祖先节点；右下角区域 U 包含 v 的所有子孙节点；左下角区域 T 包含 v 的所有前轴节点；右上角区域 S 包含 v 的所有后轴节点。

利用以上特性，XML/GML 中的路径查询处理就转换为 preorder/postorder 平面中的一系列的窗口查询(对于一个给定的 XML/GML 文档，其前序、后序值必定有边界，即最大值)，这类查询处理自然可以使用空间索引(如 R 树)来提高其查询处理效率。

3 基于一体化索引的 GML 属性数据查询处理

GML 数据查询处理包括常规的空间数据(即空间定位数据或图形数据)和属性数据的查询处理两部分。常规的空间数据的查询处理可以对传统的空间数据查询处理算法进行扩展，使其适合 GML 空间数据。GML 属性数据查询处理主要是指 GML 文档路径查询处理。

GML 文档经过扩展的前序、后序编码后，GML 中的路径查询处理就转化为 preorder/postorder 平面中的一系列的窗口查询，如表 1 所示。表中 Max 为文档中节点编号的边界值，*为可以不考虑。

表 1 XPath 轴对应的查询窗口 window(α; v)

Axis α	Query Window window(α; v)			
	preorder	postorder	level	Type
child	(preorder(v), Max)	[0, postorder(v))	Level(v) + 1	元素
descendant	(preorder(v), Max)	[0, postorder(v))	*	元素
descendant-or-self	[preorder(v), Max)	[0, postorder(v)]	*	元素
parent	[0,preorder(v))	(postorder(v),Max)	Level(v) -1	元素
ancester	[0,preorder(v))	(postorder(v),Max)	*	元素
ancester-or-self	[0,preorder(v)]	[postorder(v),Max)	*	元素
following	(preorder(v), Max)	(postorder(v),Max)	*	元素
preceding	[0,preorder(v))	[0, postorder(v))	*	元素
following-sibiling	(preorder(v), Max)	(postorder(v),Max)	Level(v)	元素
preceding	[0,preorder(v))	[0, postorder(v))	Level(v)	元素
attribute	(preorder(v), Max)	[0, postorder(v))	*	属性

基于 R 树索引的窗口查询相对比较简单，只要从 R 树的根节点开始，依次往下检测各节点的 MBR 与查询窗口是否满足查询条件(包含)，若条件成立，则继续往下搜索、检测，直到 R 树的叶节点；若条件不成立，该节点以下的子节点就不用往下检测。

a(0,45)
b(5,15)
f(25,40)

4 GML 查询索引系统的实现

本文研究的所有开发都是在JBuilder X环境下完成的。在GML查询索引系统GMLQEngine实现过程中，首先针对GML空间数据的特点，对JTS[7]提供的空间数据类型和空间操作算子进行修改、扩充；同时根据本文提出的GML一体化索引的思想，开发了GML索引模块；在此基础上，对开放源代码XQuery引擎XQEngine[8]进行了扩展，增加了空间数据类型和空间操作算子及空间索引功能，使其支持GML空间数据查询。GML数据为基于XML编码的文本数据，查询结果为GML数据的子集，即为文本数据。在GIS应用中，一般都要将查询结果以图形的方式显示给用户。本文研究开发中，将GML格式的查询结果转换为SVG格式，然后以图形方式显示给用户；或直接采用本系统开发的内部图形系统显示。

5 实验结果与性能分析

5.1 实验环境与数据

为了对本文提出的 GML 一体化索引性能进行测试，我们组织了 10 个大小不等的 GML 文档，各文档大小、所包含的几何对象个数、节点个数详见表 2。这些 GML 文档内容包括 roads、land、buildings、water、rail 等 9 个主题，其几何类型有 Point、LineString、LinearRing、Box、Polygon 和 MultiPoint、MultiLineString、MultiPolygon。

表 2 实验使用的 GML 文档大小及所含的节点个数、几何对象个数

文档大小(MB)	几何对象个数	节点个数
0.5	198	4 367
1.0	589	11 698
1.5	1 129	21 467
2.5	2 089	38 757
5.0	4 610	84 337
10.0	9 467	172 094
15.0	13 903	269 828
20.0	20 060	363 476
25.0	24 742	470 621
50.0	49 322	894 940

空间查询包括 3 类典型的空间查询：包含，相交和 k-邻近查询，查询语句详见表 3。

表 3 实验中的查询样例

编号	查询语句	查询类型
Q1	for $c in doc('GML 文档')// GMLQLFeatureMember where Containment($c,"461800 152000 461900 152510") return $c	空间包含查询
Q2	for $c in doc('GML 文档')// GMLQLFeatureMember where Intersection($c,"461800 152000 461900 152510") return $c	空间相交查询
Q3	for $c in doc('GML 文档')// GMLQLFeatureMember where nearestNeighbor (k,$c,"空间对象") return $c	空间 K-邻近查询

实验中使用的软件为本课题组用 Java 开发的 GML 空间数据查询索引系统 GMLQEngine。实验平台是 Mobile Intel 4 Compaq 笔记本电脑，主频 2.4GHz，内存 512MB，硬盘 40GB。操作系统为 Microsoft Windows 2000 Advanced Server。实验中，每个查询对每个文档大小都运行 10 次，实验结果中的运行时间是取 10 次的平均时间。

5.2 实验结果与分析

GML 空间查询表达为 GMLQEngine 中的 where 条件，即在 where 子句中增加表达空间查询的操作算子。空间包含、相交和 k-邻近查询的结果分别如表 4、表 5 和表 6 所示。对于空间包含查询(Q1)，由于主要计算是各几何对象的 MBR 与查询窗口的比较，因此 R 树索引的查询效率特别明显。有空间索引和无空间索引的查询相比，查询效率(时间比)都超过 27，并且随着文件大小的增加，查询效率越高，查询时间比越大。当文件大小超过 20MB 以后，没有空间索引已经无法进行空间查询了。这也充分说明了空间索引的作用。表 5 中的候选几何对象是指其 MBR 与查询窗口相交，但该几何对象不一定与查询窗口相交，最终查询结果才是真正与查询窗口相交的几何对象。对于 K-NN 查询，如果没有索引，则要计算所有几何对象的 MinDist 和 MinMaxDist，并要对所有的 MinDist 和 MinMaxDist 进行大小比较、排序，计算工作量相当大。所以本文研究中没有考虑无索引的 K-NN 查询，而只考虑了基于 R 树索引的 K-NN 查询，实验结果如表 6 所示。

表 4 包含查询效率比较(Query: Q1)

文档大小(MB)	0.5	1	1.5	2.5	5	10	15	20	25	50
无索引查询时间(ms)	836	1 467	1 840	2 062	2 756	6 098	11 308	175 657	内存溢出错误	内存溢出错误
R 树索引查询时间(ms)	30	40	55	70	90	110	175	222	243	271
时间比	27.86	36.68	33.45	29.46	30.62	55.44	64.62	791.25		
查询结果个数	23	26	26	36	28	36	36	29	29	87

表 5 相交查询效率比较(Query: Q2)

文档大小(MB)	0.5	1	1.5	2.5	5	10	15	20	25	50
无索引查询时间(ms)	994	1 499	1 582	2 199	3 429	6 532	13 420	内存溢出错误	内存溢出错误	内存溢出错误
R 树索引查询时间(ms)	160	175	246	280	320	441	561	850	962	1352
时间比	6.21	8.57	6.33	7.85	10.72	14.81	23.92			
候选几何对象个数	26	52	43	80	67	77	96	109	109	109
最终结果个数	9	9	19	12	22	64	25	105	105	27

表 6 k-邻近查询效率比较(Query: Q3)

文档大小(MB)	0.5	1	1.5	2.5	5	10	15	20	25	50
R 树索引查询时间(ms)1-NN	40	50	70	90	120	140	180	220	内存溢出错误	内存溢出错误
5-NN	60	80	90	120	150	180	220	280	内存溢出错误	内存溢出错误
10-NN	90	100	125	140	170	210	260	320	内存溢出错误	内存溢出错误

6 结束语

论文从 GML 空间数据本原查询的角度，对 GML 数据的索引进行了深入的研究，提出了基于空间索引的 GML 一体化高效索引机制，并实现了基于 XQuery 的 GML 查询语言。

(下转第 97 页)