高技术通讯 2013 年 第23 卷 第1 期:55 ~59

doi:10.3772/j.issn.1002-0470.2013.01.009

原生 GML 数据库存储模型研究①

张书亮② 孙玉婷 闾国年

(南京师范大学虚拟地理环境教育部重点实验室 南京 210046)

摘 要 针对地理标记语言(GML)数据管理的技术难题和 GML 对象关系数据库存储冗余度高、空间数据查询效率相对较低的弊端,在分析现有原生 GML 数据库系统存储特点和方法的基础上,提出了一种面向原生 GML 数据库的逻辑存储模型——NGDLST 模型。基于以元素为单位的细粒度存储组织方式和元素节点区间编码方案,分别定义了组成模型的元素名、属性名、元素值、属性值、元素节点、元素集合和文档集合,设计了模型的数据和索引存储结构。进行了该存储原型系统与三种主流原生 XML 数据库系统在 GML 存储与查询方面的对比实验,结果表明该模型是实用和高效的。

关键词 地理标记语言(GML), XML, 原生 XML 数据库, GML 存储, 原生 GML 数据库

0 引 言

地理标记语言(geography markup language, GML)自 2000 年提出以来一直受到地理信息系统(GIS)领域的广泛关注。由于 GML 有诸多优点,GML 和类似 GML 形式的数据大量涌现[1]。但在表达相同地理空间信息的情况下,GML 文档一般比其他格式的 GIS 文档大很多[2],因而如何有效地存储和管理 GML 文档数据,既是 GML 关键技术研究的重要内容[3],又是空间数据库领域中亟待解决的课题。目前,GML 存储的研究主要集中在基于对象关系数据库或原生 XML 数据库开展存储模型的相关研究。诸多研究通过直接或不同映射模式的方法建立了不同的 GML 对象关系数据库存储方法[4-6]。但事实证明,基于对象关系数据库的 GML 存储研究借鉴 XML 向关系数据库存储的方法,注重 GML 数据而非格式。由于对象关系数据库的平面二维结构与 GML 的树型结构差异较大,对象关系数据库并不是存储 GML 的最佳解决方案。虽然闫杰等提出了一种直接面向 GML 文档数据的即时查询引擎[7],但在面向更大 GML 文档时,该方法依然面临查询效率低下的问题。因而随着原生 XML 数据库技术的逐步成熟,利用该技术改善 GML 的存储方法和效率,研究并建立原生 GML 数据库系统,将是未来 GML 存储研究的重点[8-10]。但仔细分析现有原生 GML 存储的研究成果发现,直接将 GML 文档存入现有原生 XML 数据库的存储方法局限于在原生 XML 数据库外围扩展空间操作,忽略了存储时对 GML 空间特征与语义特性的保持,且很难与 GML 空间索引相结合,这种存储方法远远不能满足对 GML 文档的各种属性及空间操作。本研究针对传统 GML 存储方法重视 GML 文档"存"而忽略"取"所导致的 GML 存储冗余、GML 文档语义信息丢失、GML 空间查询能力有限等弊端,借鉴原生 XML 数据库相关技术,在充分考虑 GML 文档结构、空间特征及语义特性的基础上,面向原生 GML 数据库系统,从数据的组织方式及模型定义、数据和索引的存储结构等几个方面进行了探索,进而建立了原生 GML 数据库逻辑存储(native GML database logical storage, NGDLST)模型,并通过实验验证了其实用性和高效性。

1 NGDLST 模型定义

NGDLST 模型主要由元素名、属性名、元素值(简单元素非坐标值和坐标值)、属性值、元素节点、元素集合和文档集合组成,作为 GML 文档在数据库中的内部表达,它们的具体描述详见以下定义 1 -

① 国家自然科学基金(41171301)和 863 计划(2006AA12Z221)资助项目。

② 男,1974 年生,博士,教授;研究方向:地理信息共享,GML-GIS;联系人,E-mail: zhangshuliang@njnu.edu.cn

(收稿日期:2012-02-13)

7,通过这些定义,可比较清楚地了解模型中不同部分之间的关系。

定义 1 $E = \{e_1, e_2, \cdots, e_n\}$ 是一个 GML 文档的所有元素名集合,且 $\forall e_1, e_2 \in E$ 都有 $e_1 \neq e_2$。

定义 2 $A = \{a_1, a_2, \cdots, a_n\}$ 是一个 GML 文档的所有属性名集合,且 $\forall a_1, a_2 \in A$ 都有 $a_1 \neq a_2$。

定义 3 $V_s = \{s_1, s_2, \cdots, s_n\}$ 是一个 GML 文档所有坐标值集合。设 s 是 GML 文档中一个几何对象的坐标值,s 可以使用一个四元组描述为 $< geotype, env, ptnum, coord >$。其中 $geotype$ 是几何对象的类型;env 是几何对象的闭包,由以空隔分割的左上角坐标与右下角坐标表示;$ptnum$ 是坐标中点的数量;$cood$ 是坐标值的具体内容。$V_t = \{t_1, t_2, \cdots, t_n\}$ 是一个 GML 文档所有简单元素非坐标值集合。则 GML 文档的元素值集合 $V_E = \{ve \mid ve \in V_s \cup V_t\}$。

定义 4 $V_A = \{va_1, va_2, \cdots, va_n\}$ 是一个 GML 文档所有元素属性值集合,$\forall a \in A, \exists V_a \subseteq V_A$,$V_a$ 是属性名 a 在 GML 文档中的属性值集合。

定义 5 设 n 是 GML 文档中的元素节点,则 n 可以表示为 $n = (Code(n), e, ve, ((a1, va1), (a2, va2), \cdots, (an, van)), pre_{\text{nextsibling}}, pre_{\text{firstchild}})$,其中 $Code(n)$ 为 n 的节点编码;$e \in E$ 为节点的元素名;ve 为元素的值,当 n 为简单元素时 $ve \in V_E$,当 n 为复杂元素时 $ve = \varnothing$;$((a1, va1), (a2, va2), \cdots, (an, van))$ 是元素的属性名值对,$a_i \in A(0 < i < n), va_i \in V_A$;$pre_{\text{nextsibling}}$ 为元素直接后继的前序序号,当元素无后继节点时 $pre_{\text{nextsibling}} = -1$;$pre_{\text{firstchild}}$ 为元素第一个子节点的前序序号,当元素无子节点时 $pre_{\text{firstchild}} = -1$。

定义 6 设 $N = \{n_1, n_2, \cdots, n_n\}$ 是一个 GML 文档的所有元素集合,GML 文档 D 可以定义为 $D = (E, A, V_E, V_A, N)$,其中 E, A, V_E, V_A 记录文档的内容信息,N 记录文档的结构信息。

定义 7 设 $C = \{D1, D2, \cdots, Dn\}$ 是具有相同模式的文档集合,$EC = \bigcup_{i=1}^{n} E_i, AC = \bigcup_{i=1}^{n} A_i$,则文档集合 C 可以表示为 $C = \{EC, AC, (V_{E1}, V_{E2}, \cdots, V_{En}), (V_{A1}, V_{A2}, \cdots, V_{An}), (N_1, N_2 \cdots N_n)\}$。

NGDLST 模型以 GML 应用模式为单位将具有相同模式的 GML 文档存储在同一个数据集中,并为每一个 GML 文档设置一个标识符,用以唯一识别这个文档。对于单个 GML 文档的存储,在借鉴、分析和对比 XML 的基于元素和基于子树的存储策略的基础上,结合 GML 文档的要素特性,顾及到子树存储会带来 GML 文档子树划分的新难题,NGDLST 模型采用了以元素为单位的细粒度存储。它将表达元素的信息分为元素名、属性名、元素值、属性值、结构信息以及空间信息 6 类,并为各类信息设计独立的存储结构,依次形成名文件、值文件、结构信息文件以及空间坐标文件。元素内部以结构信息为主体,与元素的“名/值”、属性的“名/值”以及空间信息相关联。元素之间通过区间编码记录位置关系。

2 NGDLST 模型的数据存储结构

数据存储结构是 NGDLST 模型中数据结构的实体化,它与底层的物理存储方式相分离,无论底层采用何种方式,数据的存储结构不变。NGDLST 的数据存储结构包含空间数据、非空间数据和结构数据三部分内容。

2.1 空间与非空间数据的存储

空间数据的存储结构如表 1 所示,它由 6 部分、47 个字节的辅助存储域和 1 部分不限长度的空间数据存储域组成。

表 1 空间坐标值存储结构

1B	12B	12B	2B	16B	4B	不限长度
坐标类型	几何对象节点编码	要素节点编码	几何类型	空间闭包	点数	数据

在辅助存储域中,第一个字节主要存储 GML 文档中的坐标类型(坐标值属于某一几何对象,此时坐标类型取值 1;否则取值 0,如 BoundedBy 子节点中存储的几何坐标);12 字节的“几何对象节点编码”以 4 个字节为单位,分别存储当前元素的节点前序遍历序号、后序遍历序号和节点深度(树的根节点深度为 0);12 字节的“要素节点编码”存储坐标值所属的 GML 要素节点的节点编码;2 字节的“几何类型”存储坐标值所属几何对象的类型(0 表示“点”,1 表示“线”,2 表示“面”);16 字节的“空间闭包”以 4 个字节为单位,分别存储坐标值所表示的几何对象矩形左上和右下的 X 及 Y 坐标;4 字节的“点数”存储数据域中存储的坐标点个数。

NGDLST 模型的非空间数据主要指 GML 文档中的元素名、属性名、简单元素值以及属性值信息,

其存储结构较为简单。通过引入压缩的思想,元素名、属性名、简单元素值以及属性值的存储结构的前4字节(或2字节)用于来存储编码,其余部分表示具体的数据。

2.2 结构数据的存储

NGDLST 的结构数据利用区间编码记录节点间的层次关系,并通过对要素名、属性名、普通要素值、坐标值以及属性值的编码值的引用,描述与节点相关的信息。结构数据的存储相对复杂,如表2所示。

表2 结构数据的存储结构

2B	2B	2B	4B	…	4B	4B	4B
标签名编码	属性个数	属性名编码	属性值编码	…	元素值编码	第一子节点前序序号	后继节点前序序号

在表2中,2字节的"标签名编码"用于存储节点标签名的编码值;2字节的"属性个数"存储节点最多可以包含的属性个数(通过对 GML 应用模式的解析,可以确定属性个数,包括必选属性与可选属性);2字节的"属性名编码"与4字节的"属性值编码"成对出现,其对数与属性个数一致(当属性个数字段值为0时,这两个字段不出现在存储结构中);4字节的"元素值编码"由标志位与实际编码组成(具体的编码规则为:首位,即标志位,取0时元素值为普通值,取1时元素值为坐标值;2-32位,用于代表元素值的编码值,GML 文档中复杂元素中不含元素值,此时元素值编码取-1);4字节的"第一子节点前序序号"存储当前元素的第一个子节点前序序号(当元素没有子节点时,值取-1);4字节的"后继节点前序序号"存储当前元素的直接后继节点的前序序号(当元素没有后继节点时,值取-1)。

3 NGDLST 模型的索引存储结构

为了满足原生 GML 数据库丰富的查询方式,如值查询、结构查询以及空间查询,设计了名索引、值索引、结构索引与空间索引四种索引结构。

3.1 名索引的存储

针对 GML 文档中的元素名与属性名,名索引主要用于对指定元素名或属性名的元素进行快速定位,其存储结构如表3所示。

在表3中,2字节的"元素名/属性名编码"存储需要建立名索引的元素名或属性名的编码;2字节的"节点个数"存储元素名索引中所有该名称对应的元素个数,或属性名索引中存储包含该属性的元素个数;12字节的"节点编码"存储与该名称对应的元素节点编码,或者属性名索引中存储包含该属性的节点编码。

表3 名索引的存储结构

2B	2B	12B	12B	…	12B
元素名/属性名编码	节点个数	节点编码1	节点编码2	…	节点编码 n

3.2 值和结构索引的存储

值索引的对象可以是文档中不包含空间信息的简单元素值或者是元素的属性值,主要用于支持值的比较查询。它的构建基于 GML 应用模式,支持整型、实型和字符串型。简单元素值索引,其索引项内容为具体值和值所属元素的元素编码。元素属性值索引,其索引项为具体属性值以及属性所属元素的元素编码。本文的值索引结构由12字节的"元素编码"和不定长的元素/属性值组成。在物理存储时,值索引使用 B+树实现,索引项即为 B+树的叶子节点。NGDLST 的结构索引实际是通过节点编码的计算来实现节点之间结构关系的判断,相对比较简单,在此不再进行具体描述。

3.3 空间索引的存储

NGDLST 模型的空间索引使用 R 树索引。考虑到针对 GML 的操作大多以要素为单位,因此 R 树的索引项除了索引几何对象的闭包范围,还保留包含该几何对象的要素的节点编码,以加速空间查询时对要素信息的获取。空间索引的存储结构由3部分组成:第1部分16字节的"矩形闭包",以4个字节为单位,分别存储几何对象的矩形闭包的左上和右下的 X 及 Y 坐标;第2部分12字节的"几何节点编码"存储闭包对应的几何对象的节点编码;第3部分12字节的"要素节点编码"存储闭包对应的几何对象其父 GML 要素的节点编码。

4 实验分析

4.1 实验环境及方法

为了验证本文的模型,在 Java 环境下开发了一个基于 NGDLST 的原型系统 NativeGMLDB。NativeGMLDB 使用 Java 拓扑套件(Java topology suite, JTS)[11]开源代码实现空间分析与查询功能,利用 JavaCC[12]作为扩展 XQuery 查询语言解析器的自动

生成工具,空间数据索引的实现则采用 Spatial Index Library[13]提供的 R 树索引开源代码。

实验使用 Shape file 格式的南京市基础地理数据中"居民地层"作为源文件,通过对其分别进行编辑、修改,再利用 FME2006 进行数据转换,最终形成实验所需要的 6 个 GML 文档,大小分别为 2MB,5MB,10MB,20MB,50MB,70MB。实验使用的 GML 文件大小及所包含的节点个数、几何对象个数如表 4 所示。为了验证 NativeGMLDB 在 GML 存储方面的优势,研究中选择基于内存的 XQuery 执行引擎(XQEngine)以及两个较为常用的开源原生 XML 数据库软件(dbXML 和 eXist)与 NativeGMLDB 进行存储时间与空间大小的测试对比实验。为了验证 NativeGMLDB 在 GML 查询方面的优势,通过在 W3C 的 XML 查询语言规范 XQuery 中扩展空间查询表达,我们设计了带谓词的非空间查询样例 Q1 和混合空间查询样例 Q2,如表 5 所示。实验采用的硬件环境为 CPU P4 3.06 GHZ、内存 1G、硬盘 120G,操作系统为 Windows XP。实验中对每个文档的存储与查询都运行 10 次,实验结果中的运行时间取 10 次的平均时间。

表 4 实验数据特征描述

Shp 文件大小(MB)	GML 文件大小(MB)	几何对象个数	节点个数
1.3	2.0	3251	26013
3.3	5.1	8128	64013
6.5	10.2	16255	127133
13.2	20.4	32510	368021
33.0	51.8	81138	650221
48.8	70.3	131554	896013

表 5 查询样例描述

编号	查询语句	查询描述
Q1	For MYMc in doc('GML 文档')// fme:jmd where MYMc/ fme:LEN > =1338.9 Return MYMc	祖先/后代关系路径查询 + 基于元素值的条件查询(查询文档中任意位置的 fme:jmd 元素,其子元素 fme:LEN 值大于 1338.9)
Q2	For MYMc in doc('GML 文档')// fme:jmd Where contains(*rect*, $c) and $c/ fme:LEN > =1338.9 Return $c	空间包含查询 + 基于元素值的条件查询,即混合空间查询(查询文档中包含在矩形 *rect* 中的 fme:jmd 元素,并且其子元素 fme:LEN 的值大于 1338.9)

4.2 查询实验结果与分析

针对表 5 中 Q1 的非空间查询样例,查询实验结果如表 6 示。

表 6 非空间查询(Q1)时间比较结果

	查询时间(s)					
	70	2	5	10	20	50
XQEngine	0.31	0.75	1.58			
dbXML	1.33	2.74				
eXist	0.56	1.11	1.66	3.14	4.68	6.50
NativeGMLDB	0.32	0.63	1.14	1.77	3.75	5.01
结果个数	2349	5722	10923	22643	54532	70483

由表 6 的实验结果可以看出,5MB 以下的数据使用基于内存的 XQEngine 查询效率最高,但其并不能支持较大数据的查询。dbXML 采用基于整个文档的索引存储模型,并不适合于文档节点的查询,查询效率低于 NativeGMLDB 与 eXist。而 NativeGMLDB 的查询效率整体优于 eXist。

由于 XQEngine、dbXML 和 eXist 不能支持 Q2 的查询方式,因此,无法开展对比实验。针对表 5 中 Q2 的查询样例,NativeGMLDB 实验结果如表 7 所示。

表 7 混合空间查询(Q2)时间比较结果

文件大小(MB)	2	5	10	20	50	70
总体查询时间(s)	0.15	0.21	0.52	1.30	3.44	4.43
查询结果个数	901	1041	3324	10127	26598	32876

由表 7 的实验结果可以看出,混合空间查询的时间与使用索引后产生的候选结果个数正相关。空间索引查询时间非常短,使用索引后对候选结果进行精确空间运算的时间占了空间查询时间的主要部分。借助空间索引,NativeGMLDB 执行空间查询的效率较高(对于有 1 万条查询结果的混合查询语句,NativeGMLDB 查询执行时间仅为 1.3s)。

5 结 论

从 GML 的存储技术需求出发,充分考虑 GML 的结构与语义特性和借鉴原生 XML 数据库的存储思想建立面向原生 GML 数据库的逻辑存储模型的方法,可完善和发展 GML 及其他 XML 形式文档数据的原生数据库存储。下面的工作是进一步研究建立在 NGDLST 之上的操作方法和接口,并通过完善 NativeGMLDB 来丰富原生 GML 数据库系统的理论与方法。

参考文献

[1] 张书亮,闾国年,苗立志等. GML 在中国的研究进展. 地球信息科学,2008,10(6):763-769

[2] 贾文珏,龚健雅,李斌. Web 要素服务的优化方法. 测绘学报,2005,34(2):168-174

[3] Chang-Tien Lu, Raimundo F, Dos Santos Jr, et al. Advances in GML for geospatial applications. *Geoinformatica*, 2007, (11):131-157

[4] 张爱国,邬群勇,王钦敏. 基于 PostgreSQL 数据库的 GML 数据存储. 测绘科学,2008, 33(1):194-196

[5] Corcoles J E, Gonzalez P. Analysis of different approaches for storing GML documents. In: Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems, Virginia, USA, 2002. 11-16

[6] Jeung H Y, Park S H. A GML data storage method for spatial databases. *The Journal of GIS Association of Korea*, 2004,12(4): 307-319

[7] 闫杰,方金云,韩承德等. GML 即时查询引擎研究与实现. 高技术通讯,2008,18(11):1154-1160

[8] Huang C H, Chuang T R, Deng D P, et al. Building GML-native web-based geographic information systems. *Computers & Geosciences*, 2009, (35):1802-1816

[9] 史婷婷,李岩,王鹏. 基于 GML 空间数据存储方法研究与实现. 计算机应用,2006,26(10):2408-2412

[10] 兰小机,闾国年,刘德儿. GML 空间数据查询与索引机制研究. 遥感学报,2006, 10(6):854-863

[11] JTS Topology Suite. http://www.vividsolutions.com/jts/jtshome.htm, 2009

[12] JavaCC. https://javacc.dev.java.net, 2009

[13] Java Spatial Index (RTree) Library. http://www.sourceforgecn.net/Projects/j/js/jsi/, 2009

Research on a storage model for native GML databases

Zhang Shuliang, Sun Yuting, Lv Guonian

(Virtual Geographical of MOE Key Laboratory, Nanjing Normal University, Nanjing 210046)

Abstract

In view of the technical difficulties in geography markup language (GML) data management, the high storage redundancy and the low efficient spatial data querying of GML object-relational databases, the paper proposes a logical storage model for native GML databases on the basis of the analysis of the existing native GML databases management systems' storage characteristics and methods. It is called the NGDLST model. The model's components of element name, attribute name, element value, attribute value, element node, element collection and document collection are defined based on a range encoding scheme and a tiny object storage mode which takes an element as the unit to store GML data, and on the basis of this, the model's storing structure for data and indexes is given. The results of the experiment on comparing the performance of the model's prototype system with that of three mainstream native XML database management systems show that the proposed model can perform better in GML storage and query.

Key words: geography markup language (GML), XML, native XML database, GML storage, native GML database