DOI:10.13203/j.whugis2004.12.009
第29卷 第12期 武汉大学学报·信息科学版 Vol.29 No.12
2004年12月 Geomatics and Information Science of Wuhan University Dec. 2004

文章编号:1671-8860(2004)12-1071-04 文献标识码:A

GML空间数据存储映射模型研究

李 俊[1] 关佶红[1,2] 李玉珍[1]

(1 武汉大学计算机学院,武汉市珞喻路129号,430079)
(2 武汉大学空间信息与数字工程研究中心,武汉市珞喻路129号,430079)

摘 要:提出了一种将GML文档存储到空间数据库的存储映射模型G2SDB,同时通过定义边表,保留GML文档的部分结构信息,有利于处理GML查询中的路径表达式,提高查询操作的效率,也有利于GML文档的重构。

关键词:GML;空间数据库;G2SDB;内联模式树;空间数据库模式
中图法分类号:P208

1 GML数据存储技术的研究状况

GML是基于XML的,根据数据库管理系统的不同,XML数据库可以分为本源的[1]、基于面向对象的和基于关系的XML数据库系统。由于目前本源的XML数据库和面向对象数据库还不是十分成熟,因此,采用关系数据库系统来存储和管理XML数据是最切实可行的方法。在学术界,研究者们已经提出了XML到关系数据库的多种映射方法,具体见文献[2~6]。在工业界,各主要的关系数据库厂商都推出了支持XML的产品,如IBM DB2的XML Extender、SQL Sever 2000的XML等。

由于XML文档仅包含文字和数字,而GML文档还包含对空间对象的空间几何属性的描述,如点、线串、面等,GML3.0还增加了对复杂的几何实体、拓扑、空间参照系统、元数据、时间特征和动态数据等的支持,使其更加适合描述现实世界的问题,所以针对XML的处理方法不一定适合GML。有关具体研究见文献[7,8]。其不足在于,GML文档中的每个元素都必须对应地定义一个关系表,且不能自动处理父子元素之间隐含的参照关系。同时,GML文档的格式必须符合它们定义的应用模式,局限性很大。此外,文献[7,8]只是简单地将GML文档中的空间信息以坐标点的形式存储到关系表中,无法对其进行空间操作和分析。

针对现有GML数据存储技术存在的问题,本文设计的G2SDB能通过预定义的映射规则将GML文档存入空间数据库中。空间数据库是扩展了的关系数据库,它引入了面向对象模型,在此基础上扩展对空间数据的管理功能。可用的产品有Oracle的Oracle Spatial、Informix的DataBlade、IBM DB2的Spatial Extender等,它们已经实现了一些点、线、面等抽象数据类型(ADT),而且也实现了一部分空间操作功能。

2 G2SDB存储映射模型

G2SDB存储映射模型如图1所示。

2.1 模式简化器

GML应用模式的定义可能比较复杂,为了减少生成数据表的数量和多表关联对查询效率的影响,需要对其进行简化,由模式简化器完成。为叙述方便,将GML应用模式中部分标签或元素简化为表1所示的符号。

收稿日期:2004-09-05。
项目来源:国家自然科学基金资助项目(60373019);国家863计划资助项目(2002AA135340);软件工程国家重点实验室开放研究基金资助项目(SKL(4)003);测绘遥感信息工程国家重点实验室开放研究基金资助项目(WKL(01)0303);IBM软件奖研金资助项目。

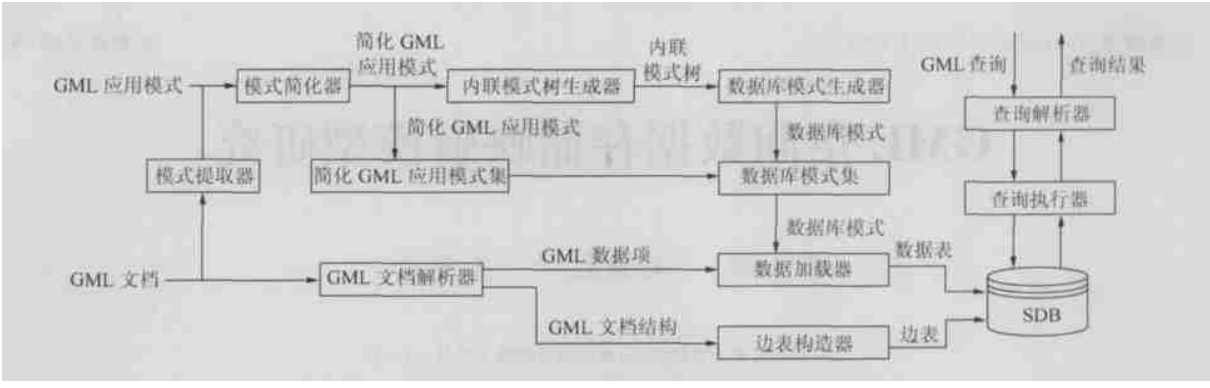

图 1 G2SDB 存储模型

Fig. 1 G2SDB Storage Model

表 1 GML 应用模式中标签或元素的简化符号

Tab. 1 Simplified Symbols of Tags or Elements in GML Application Schema

GML 应用模式中的标签或元素	符号
< choice>	|
< sequence>	,
< element type= S/>	S
< element type= S minOccurs= 0/>	S?
< element type= S minOccurs= 0 maxOccurs= unbounded/>	S*
< element type= S minOccurs= 1 maxOccurs= unbounded/>	S+

简化规则有：① S+ → S *；② S? → S；③ $(S_1 | \cdots | S_n) \rightarrow (S_1, \cdots, S_n)$；④ (a) $(S_1, \cdots, S_n)^* \rightarrow (S_1^*, \cdots, S_n^*)$，(b) $S^{**} \rightarrow S^*$；⑤ (a) …, S, …, S, …→…, S *, …, …, (b) …, S, …, S *, …→…, S *, …, …, (c) …, S *, …, S, …→…, S *, …, …; (d) …, S *, …, S *, …→…, S *, …, …。

运用规则①后，GML 应用模式中的元素< element type = S minOccurs = 1 maxOccurs = unbounded/> 被< el— ement type=S minOccurs=0 maxOccurs=unbounded/> 替代，它是对元素定义的扩充，便于定义映射规则。运用规则②后，元素< element type=S minOccurs =0/> 被< element type =S/> 替代。运用规则③后，标签< choice> 被标签< sequence> 替代。规则②、③通过引入冗余来简化数据库模式的定义。运用规则④后，元素统一地表示成< element type=S/> 或< element type =S minOccurs=0 maxOccurs = unbounde— d/> 的形式。运用规则⑤后，GML 应用模式中没有重复出现的元素。规则④、⑤会影响 GML 应用模式中部分元素的次序关系，可以通过边表弥补。简化后的 GML 应用模式在保证不丢失原模式信息的基础上有利于映射转换、减少生成的数据表的数量和提高查询效率。

由于篇幅有限，本文只给出一个简化后的 GML 应用模式实例，见图 2。

```
<element name="Map"type="ex:mapType"
substitutionGroup="gml:_FeatureCollection"/>
<element name="Layer" type="ex:layerType"/>
<element name="Feature"type="ex:featureType"
substiutionGroup="gml:_Feature"/>
<element name="SimpleProperty"
type="ex:SimplePropertyType"/>
<element name="GeometryProperty"
type="ex:GeometryPropertyType"/>
<element name="FloatProperty"
type="ex:FloatPropertyType"/>
<complexType name="mapType">
<complexType name="layerType">
<sequence>
    <element ref="gaml:featureMember"
    minOccurs="0" maxOccurs="unbounded"/>
</sequence>
</complexType>
<complexType name="featureType">
<complexType name="GeomtryPropertyType">
<complexType name="SimplePropertyType">
<complexType name="FloatPropertyType">
<complexType name="StringPropertyType">
</schema>
```

图 2 简化后的 GML 应用模式实例

Fig. 2 Sample of Simplified GML Application Schema

2.2 内联模式树生成器

可以用树型结构表示简化后的 GML 应用模式，称其为模式树。该树的根节点代表 GML 应用模式的根元素，叶节点代表 GML 应用模式中仅包含文本的元素。若元素包含的是非空间数据，叶节点代表的是元素；若元素包含的是空间数据，此时将该元素与一个空间几何对象对应(如 Oracle Spatial 的 MDSYS.SDO -GEOMETRY)，则模式树的叶节点代表对象。GML 中的几何体与空间数据库中(以 Oracle Spatial 为例)的几何对象的对应关系如表 2 所示。模式树的中间节点代表 GML 应用模式中的其他元素及其所包含的元素。包含其他元素的节点是被包含元素的父节点，被包含元

素是包含它的元素的子节点。

表 2　GML 中几何体与 Oracle Spatial 中几何对象的对应关系

Tab. 2　Correspondence Between Geometry in GML and Geometry in Oracle Spatial

GML	SDO-GEOMETRY
PointType	POINT
LineStringType	LINE 或 CURVE
PolygonType	POLYGON
MultiPointType	MULTIPOINT
MultiLineStringType	MULTILINE 或 MULTICURVE
MultiPolygonType 或 LinearRingType	MULTIPOLYGON

模式树中的各节点有一个惟一的标识符, 并用它所代表的元素或对象的标签名来标记。节点间的边表示它们之间的父子关系。如果父子节点间是一对一的关系, 用边"|"将它们联系起来; 如果父子节点间是一对多的关系, 只列出其中一个子节点, 并用边"| *"将它们联系起来。用模式树表示图 2 的 GML 应用模式, 如图 3 所示。

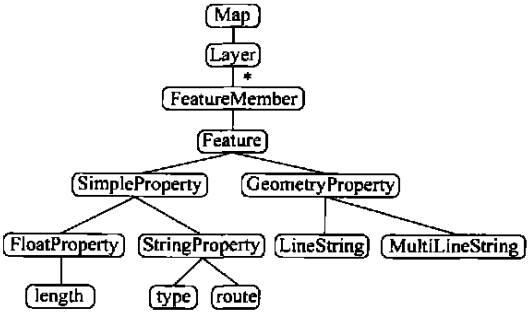

图 3　模式树

Fig. 3　Schema Tree

为了构造最优的数据库模式, 需对模式树进行内联, 即尽可能地将树中的子孙节点合并到父节点。完成内联后, 得到的树型结构称之为内联模式树。内联规则如下。

规则 1　如果模式树中的节点 a 通过边"|"连接到节点 b, 将节点 b 内联到节点 a, 并将节点 b 从树中删除。

规则 2　如果模式树中的节点 a 通过边"| *"连接到节点 b, 即节点 a 包含多个节点 b, 不内联节点 b。

内联产生的节点以它代表的所有元素的标签名标记, 其他节点以及边的表示方法与模式树相同。内联模式树采用深度优先搜索算法判断模式树中的每个节点可否内联。

根据内联规则对图 3 的模式树进行内联, 得到内联模式树, 如图 4 所示。

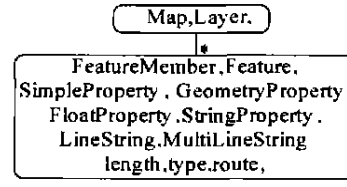

图 4　内联模式树

Fig. 4　Inlined Schema Tree

2.3　数据库模式生成器

根据内联模式树可确定最终的数据库模式。为内联模式树中的每个节点 e 创建一个对应的数据表, 规则如下。

规则 3　以整型字段 ID 作为数据表的主码, 它是对存入数据表的元组的编号。必须保证每个表的 ID 号互相不重复。

规则 4　如果| e.inlinedSet |=1, 表示没有其他节点内联到该节点, 将该节点代表的 GML 元素的名称作为数据表的属性, 也为该节点对应的 GML 元素的属性创建同名的数据表的属性。

规则 5　如果| e.inlinedSet | ≥ 2, 表示有其他节点内联到该节点, 用同一个数据表存储这些节点所对应的 GML 元素。记录该节点代表的所有元素的包含关系, 以"起始元素。中间元素。终端元素"作为数据表的属性。如果有元素包含属性, 则根据属性的归属以"起始元素。属性"、"起始元素。中间元素。属性"或"起始元素。中间元素。终端元素。属性"作为数据表的属性。

规则 3 确定了生成的数据表的主码, 规则 4 和 5 分别对内联模式树中节点可能出现的情况进行相应的处理, 确定了生成的数据表的属性。以上规则完整地将内联模式树中的节点和数据表对应起来, 也就是将 GML 文档中的元素或属性与数据库中的表或表的列对应起来, 从而完成了从 GML 应用模式到空间数据库模式的映射转换。

根据以上规则为图 4 的内联模式树创建数据库模式如下(由于部分属性名称过长, 为简便起见, 仅以各元素的首字母表示, 如 F. F. G. L 表示 FeatureMem-ber.Feature.GeometryProperty. LineString):

Map(ID, M ap. Layer)

FeatureMember (ID, F. F. S. F. l, F. F. S. S. t, F. F. S. S. r, F. F. G. L, F. F. G. M)

其中, 字段 ID、Map. Layer 的类型为 integer; F. F. S. F. l 的类型为 float; F. F. S. S. t、F. F. S. S. r 的类型为 string; 而字段 F. F. G. L、F. F. G. M 的类型为 MDSYS. S— DO-GEOMETRY(以 Oracle Spatial 为例)。

2.4 边表构造器

笔者还为每个 GML 文档创建了另一个表 Edge (parentID, childID, parentType, childType), 称作边表。边表用来记录 GML 文档中已存入数据表的元素间的父子关系。边表中的 parentID、childID 字段是对 GML 文档中父子元素的编号, 必须保证内联模式树中各节点包含的起始元素的编号与数据表中对应元组的 ID 号相同。parentType、childType 字段记录父子元素的名称。对于文档中的空间元素, 还是以空间几何对象的形式表示。

边表的作用如下。

1)边表将存入空间数据库中的各数据表关联起来。数据表存储了 GML 文档中的数据, 而没有记录下这些数据之间的关系。边表记录了 GML 文档中元素间的父子关系, 也就是保存了数据表数据间的关系, 通过 parentID、childID 与数据表的 ID 号相同的规定, 将边表与数据表关联起来, 满足了数据库的完整性约束条件。

2)边表是提高 GML 查询效率的核心。在 GML 查询处理中, 路径表达式扮演着关键角色。边表记录了 GML 文档中元素间的父子关系, 笔者分别在字段 parentID、childID 上创建索引, 为查询处理提供支持。查找出目标元素后, 通过 parentID、childID 与数据表 ID 号的关联, 在数据表中提取相应的数据。

3)边表为 GML 文档重构提供支持。边表记录了 GML 文档中部分元素间的父子关系, 当需要将存储的数据重新转换为 GML 文档时, 边表起到指导作用。

参 考 文 献

1 Bourret R. XML and Database. http://www.rpbourret.com/xml/XMLAndDatabases.htm, 2004

2 Shanmugasundaram J, Tufte K, He G, et al. Relational Databases for Querying XML Documents: Limitations and Opportunities. The 25th VLDB Conference, Edinburgh, 1999

3 Bohannon P, Freire J R J. LegoDB: Customizing Relational Storage for XML Documents. The 28th VLDB Conference, Hong Kong, 2002

4 Florescu D, Kossmann D. Storing and Querying XML Data Using an RDBMS. IEEE Data Engineering Bulletin, 1999, 22(3): 27~34

5 Schmidt A R, Kersten M L, Windhouwer M A, et al. Efficient Relational Storage and Retrieval of XML Documents. International Workshop on the Web and Databases (WebDB) (in Conjunction with ACM SIGMOD), 2000. 47~52

6 Yoshikawa M, Amagasa T. XRel: A Path-based Approach to Storage and Retrieval of XML Documents Using Relational Databases. ACM Transactions on Internet Technology, 2001, 1(1): 110~141

7 Córcoles J E, González P. Analysis of Different Approaches for Storing GML Documents. ACM-GIS, 2002. 11~16

8 Reinehr I, Dias E. A Mechanism to Import GML Data into a Relational Database. http://www.cpqd.com.br, 2003

第一作者简介: 李俊, 硕士生。研究方向: GML 空间数据存储和索引技术。

E-mail: moring-li@163.com

G2SDB: A New Storage Model for GML Documents

LI Jun[1] *GUAN Jihong*[1,2] *LI Yuzhen*[1]

(1 School of Computer Science, Wuhan University, 129 Luoyu Road, Wuhan 430079, China)

(2 Research Center of Spatial Information and Digital Engineering, Wuhan University, 129 Luoyu Road, Wuhan 430079, China)

Abstract: This paper proposes a new storage model called G2SDB to store GML data into a spatial database. The authors converts GML application schema to a tree named schema tree. Then the schema tree is inlined according to the predefined inlining rules. On the basis of the predefined mapping rules, the mapping between the inlined schema tree and a spatial database schema is determined.

Key words: GML; spatial database; G2SDB; inlined schema tree; spatial database schema

About the first author: LI Jun, postgraduate, majors in storage and index techniques of GML documents.

E-mail: moring-li@163.com

(责任编辑: 晓平)