Evaluating GML Support for Spatial Databases

Lakshmi N Sripada, Chang-Tien Lu
Department of Computer Science
Virginia Tech
7054 Haycock Road
Falls Church, VA 22043
{lsripada, ctlu}@vt.edu

Weili Wu
Department of Computer Science
University of Taxes at Dallas
2601 North Floyd Road
Ricahrdson, TX 75083
weiliwu@utdallas.edu

Abstract

This paper presents a study on Geography Markup Language (GML), the issues that arise from using GML for spatial databases and solutions that have been proposed. We concentrate on three aspects of GML, including storage, parsing, and querying. GML is an XML encoding for storing geographic data. It has been developed by the OpenGIS as a medium for uniform geographic data storage and exchange among diverse applications. The underlying concepts of XML therefore can also be applied to GML data. This results in both advantages and disadvantages, which are discussed in the paper. GML is a comparatively new language in the field of geographic information systems and still in its developmental stage. Most of the data processing techniques need to be further developed in order for GML to be an efficient medium for geographic data storage and processing.

1. Introduction

With the increase in the use of Internet for the exchange of information, there is a need for developing languages that can be used by the diverse systems to understand each other. Markup languages were developed to provide a common medium of data description and display for such systems. XML or Extensive Markup Language was an attempt in this direction. XML is a text based meta-language. Being a meta-language, other languages can be developed by extending XML for use in specific areas of application. Geography Markup Language (GML) has been defined as an XML encoding for geographic information. This encoding helps in the storage, exchange, and modeling of geographic information systems. GML uses the concepts provided in the abstract specification (simple feature data model) by OpenGIS consortium for modeling geographic objects, such as geometry, topology, time, and features. It is a data descriptive language, which means that the data is stored in a self-descriptive manner. It is not a presentation markup language like HTML. In other words, GML stores the data but does not indicate how the data is to be displayed. GML has been designed to be used as a mechanism for information discovery, retrieval and exchange [11]. Since spatial databases store geographical information, the concepts of GML can be used for the storage and exchange of spatial data sets.

A spatial database system has been defined as a database system supporting spatial data types. Spatial data type objects not only have a non-spatial description, such as name and population, but also contain spatial attributes associated with them, such as location, geometry, and neighborhood properties. A spatial database system has to provide various functionalities, including input, storage, retrieval, selection, analysis, and display of the information [12]. Although these features are provided by traditional databases, such databases do not store information in a uniform format. This leads to the difficulties of exchanging information among databases. GML proves to be a common language in which these databases can exchange information with each other. Thus, although the representation of information is unique, the use of information can be different and meaning can vary depending on the context. This makes the data very flexible and portable.

However, using GML for spatial databases has its advantages and disadvantages. The data types required for spatial databases have spatial and geometric attributes in addition to the one-dimensional attribute of traditional data type. GML documents are suitable for storing such data types due to their nested structure, which involves storing similar data together, thus making it effective in representing spatial features and attributes. Spatial databases involve a large amount of data, which has to be stored in such a way as to allow efficient query processing. However, extracting information from GML documents is rather tedious because the XML/GML parser parses the entire data before looking for a specific piece of information. This might lead to very inefficient query processing, especially for large databases. Since GML is an XML encoding for geographic data, the query languages and other data processing capabilities available for XML can also be used for GML. A number of query languages have been developed for XML documents. Most of these languages provide support for only alphanumeric data. However, GML documents also

contain spatial and temporal attributes in addition to the alphanumeric data. These XML query languages have to be extended to provide support for GML documents. Thus even GML promises to provide many capabilities, mechanisms to implement these capabilities and use them effectively have not yet been fully developed.

The rest of the paper is organized as follows. Section 2 gives a description of GML document schema. Section 3 summarizes studies conducted on GML document storage. Section 4 provides an overview of GML parsers. Section 5 introduces query languages available for GML. Section 6 discusses the advantages of GML. Finally, we summarize our work in Section 7.

2. GML Schema

GML is a markup language, which means that GML documents have to follow certain rules in order to be termed as valid GML documents. These set of rules are defined in a schema document. GML specification states that the method of generating the GML document is irrelevant, that is, the documents can be hand-generated or through any available tools. All that is required is that the documents should conform to the requirements in the GML specification. GML version 1.0 uses the Document Type Descriptors (DTDs) for defining the elements and their associated attributes. However, DTDs have many disadvantages. For example, they are not written in XML, which makes it inconvenient to be interpreted. GML version 2.0 and 3.0 use GML schemas instead of DTDs. A GML schema is written in XML, which means that one interpreter can be used for both the DTDs and GML documents. The elements and their attributes used in a GML document must be defined in the related GML schema for the document to be valid. A GML schema provides a set of type definitions and element declarations that can be used to check the validity of well-formed GML documents [15]. GML defines various entities such as features, geometries and topologies through a hierarchy of GML objects. GML specification provides a series of schema for describing geographic data. The GML schemas defined in the OpenGIS specification include feature, geometry, topology, value, coordinate reference system, and style-descriptor. There are other schemas that are subclasses of the feature schema, such as observation, coverage and definition. Depending on the requirements of the application domain, designers can create different types. These schemas can be created by adding or restricting the features of GML base schema [8]. This provides the flexibility to use GML to represent diverse types of spatial objects. Using some basic features, all these objects can be described. Most applications make use of only a subset of the schemas that have been defined in the GML specification.

3. GML Document Storage

Efficient storage of GML documents is also an important issue. One of the important considerations regarding the storage of geographic data is that the data does not have to be stored in GML format for transportation. The data can be stored in one of the existing formats and converted into GML format for exchange whenever required. GML documents are much larger in size than other documents containing the same information due to their descriptive nature. The techniques used for the storage of XML documents cannot be directly used for the storage of GML documents due to the differences in XML and GML data, in the sense that GML data have larger number of dimensions than the XML datasets, and the spatial attributes of GML data also have to be preserved while storing these documents.

Most of the approaches to storing GML data are based on relational model or object-oriented model. One approach proposed for the storage of GML documents is to use the concepts of relational databases for the storage of GML documents [7]. In this way, a complete set of data management services would be available. Such database schemas have been divided into two categories: structure mapping approach and model mapping approach [19]. The design of the database schema is based on the understanding of the DTD (Document Type Descriptor) or GML schema that describes the structure of the GML documents in case of structure mapping. Under the model mapping approach, a fixed database schema is used to store any GML documents without the assistance of GML schema or DTD. In a study conducted to compare the performance of different approaches of storing GML documents [7], three types of document storing techniques were compared, including LegoDB (structure mapping), Monet, and Xparent (model mapping). The study concludes that LegoDB is the best approach to store GML documents as it works well for both queries involving large number of attributes and documents that have large amounts of data. Any approach for mapping GML documents to databases should take advantage of the features provided by the relational databases like concurrency control and at the same time allow for querying large amounts of data efficiently with queries involving a large number of attributes.

4. GML Parser

A GML parser reads the GML document and creates a representation of the document. This representation can be accessed by other parts of the application. Software applications interpret the output from the parser into their own local meaningful contexts. Not many parsers are available for reading and interpreting the GML files. However, XML parsers can be used for parsing GML files, as GML is based on XML specifications [10]. Some of the XML parsers available are Xerces2 [3], XSV [16],

and MSXML4.0 [1]. A software application must know what each element in the GML dataset means - whether the element refers to a feature, a property of a feature, or a feature collection, so that the data can be used in a meaningful context. The software has to meet two requirements for processing GML information - it has to use the GML parser to validate the data so that it conforms to the GML schema, and also it should understand how the data has been defined in GML according to the specification. This knowledge will help the application in correct interpretation of the data. Not only the document has to be read by the application, it also has to be interpreted by the application in terms of the GML specification. This interpretation would result in extraction of data embedded in the GML document for use by the application in its local context.

There are two standard APIs that are used by software applications to parse GML documents: DOM (Document Object Model) and SAX (Simple API for XML). The choice of DOM or SAX parser for GML documents depends on the resource usage of each of these approaches and their efficiencies. DOM constructs a tree structure for the GML data as it processes the data. This structure would involve a large amount of memory for spatial databases. Therefore, DOM seems to be unsuitable for GML documents. A SAX parser parses the documents sequentially, treating the document as a data stream. This consumes much less resources and hence can be used for large data sets. However, SAX parser has to process the entire document before the processed data is available to the application. Moreover, it does not support random access of data. This might cause it to be inefficient for spatial databases due to their large data size. Various studies have been performed to compare the performance of GML parsers [14, 18]. Due to the differences in their approaches to store the parsed documents, DOM and SAX have their own limitations. While SAX model excels in point and range queries, DOM is better for join operations. SAX is suitable for storing and processing data in web server; in contrast, DOM is suitable for storing documents at the client side since it requires more resources. The use of DOM parsers for GML documents is thus not feasible, because of its memory usage. SAX parser, on the other hand, is inefficient in case of queries involving large number of attributes since the data involved in spatial databases is large. Thus a parser has to be developed combining the features or advantages of these two types of parsers. This is a major area in the field of GML that has to be explored in order to make GML a universal language. Table 1 provides a summary of the features provided by DOM and SAX.

5. GML Query Languages

Even the well-known approaches of querying that work well with the XML files could not guarantee good results when applied to GML documents that contain both alphanumeric and spatial data [7]. The storage of information in GML documents leads to new problems of extracting data from those documents. Many query languages have been proposed for querying GML documents [6, 17]. Being derived from XML, GML has the advantage of readily available query languages that have been developed for XML. The query languages available for XML documents are of general use. However, these languages have to be extended with spatial operators if the language is to be used for GML. A specification of query language for GML by extending the concept of XML-QL [13] has been proposed [6] and compared based on the features defined by [5].

All query languages have an underlying data model that abstracts away from the physical representation of the data. For example, relational query languages operate on relations and object-oriented query languages on objects. Therefore, it is necessary to define precisely a data model for GML. The objects represented in GML are much more complex compared to the ones in XML, since geographic objects have spatial attributes in addition to the non-spatial attributes that are used to describe spatial objects. The data model for GML query language has to reflect this complexity. It has to be more efficient than the data model for XML in the sense that the data represented in GML would be more complicated and will have more dimensions associated with it. The queries for GML data are of two types- spatial and non-spatial. Non-spatial queries are similar to the XML queries, since XML queries and documents involve alphanumeric data only. Therefore, XML query language models can be extended so that the spatial query attributes of GML can be included. This takes advantage of the existing XML query processing capabilities available and at the same time provides the capabilities required for GML data processing. Xquery (XML query language) is one of the proposals for query languages for XML. Xquery has been designed to meet the requirements of an XML query language as identified by W3C XML query working group [2]. Vatsavai claims that Xquery is the most comprehensive of all languages and chooses Xquery instead of any other language for extending to GML query language [17]. Different approaches for developing query languages have been proposed [4, 6, 17]. However, an important consideration while developing such languages is to decide whether to extend the already existing XML query languages or to develop a new query language for GML.

6. Summary

GML has been developed to help in the interchange of geographic data over the Internet and across diverse systems. The sharing of such data involves support for various services for the interpretation and storage of data.

Although GML specification has the guidelines for developing GML documents, much work has still to be done in the area of parsing, querying, and storage of these documents. The storage of GML documents and the querying of those documents are inter-related. Efficient query processing requires that the documents be stored in a logical manner preserving the neighborhood characteristics of spatial data. Also the storage techniques can take advantage of already existing techniques such as relational model. Parsing of GML documents is another area of active research. Parsing of these documents presents a difficult situation due to the size of these documents. However, despite of these drawbacks, GML would emerge as a major form of geographic data exchange in future due to its non-proprietary and portable nature.

7. References

[1] "MSXML 4.0 SDK," http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk30/htm/xmmscxmloverview.asp.

[2] "XQuery 1.0: An XML Query Language," *W3C Working Draft* http://www.w3.org/TR/xquery/, 2003.

[3] Apache XML project, "Xerces2 Java Parser," http://xml.apache.org/xerces2-j/index.html.

[4] D. Beech, A. Malhotra, and M. Rys, "A formal data model and algebra for XML," *University of California, Berkeley, CS 298-13: Digital Library Seminar* http://elib.cs.berkeley.edu/seminar/2000/20000207.pdf, 2000.

[5] A. Bonifati and S. Ceri, "Comparative analysis of five XML query languages," *ACM SIGMOS Record*, vol. 29, pp. 68-79, 2000.

[6] J. E. Corcoles and P. Gonzalez, "A specification of a spatial query language over GML," *Geographic information systems. Proceedings of the ninth ACM international symposium on Advances in geographic information systems*, pp. 112-117, 2001.

[7] J. E. Corcoles and P. Gonzalez, "Analysis of different approaches for storing GML documents," *Geographic information systems. Proceedings of the tenth ACM international symposium on Advances in geographic information systems*, pp. 11-16, 2002.

[8] S. Cox, P. Daisey, R. Lake, C. Portele, and A. Whiteside, "OpenGIS Geography Markup Language (GML 3.0) Implementation Specification," *OpenGIS Specifications* http://www.opengis.org/specs/?page=specs, 2003.

[9] Galdos Systems Inc, "Top 10 benefits of using GML," *Wireless developer network*, http://www.wirelessdevnet.com/channels/lbs/features/top10gml/, 2001.

[10] D. Murray and J. C. Chow, "An XML-Driven data translation engine for GML 2," *For proceedings of the urban and regional information systems association* http://www.safe.com/company/media_archive/GML_User_Perspectives.pdf.

[11] Z.-R. Peng and M.-H. Tsou, *Internet GIS Distributed Geographic Information Services for the Internet and Wireless Networks*: John Wiley & Sons, Inc., 2003.

[12] P. Rigaux, M. Scholl, and A. Voisard, *Spatial Databases with Application to GIS*: Morgan Kaufmann Publishers, 2002.

[13] J. Robie, J. Lapp, and D. Schach, "XML Query Language (XQL)," http://www.w3.org/TandS/QL/QL98/pp/xql.html.

[14] S. Shekhar, R. R. Vatsavai, N. Sahay, T. E. Burk, and S. Lime, "WMS and GML based interoperable web mapping system," *Geographic Information Systems. Proceedings of the ninth ACM international symposium on Advances in geographic information systems*, 2001.

[15] H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, "XML Schema Part 1: Structures, W3C Recommendation 2 May 2001," http://www.w3.org/TR/xmlschema-1/, 2001.

[16] H. S. Thompson and R. Tobin, "Current Status of XSV: Coverage, known bugs, etc," http://www.ltg.ed.ac.uk/~ht/xsv-status.html, 2004.

[17] R. R. Vatsavai, "GML-QL: A Spatial Query Language Specification for GML," *Department of Computer Science and Engineering, University of Minnesota* http://www.cobblestoneconcepts.com/ucgis2summer2002/vatsavai/vatsavai.htm.

[18] W3C DOM WG, "Document Object Model FAQ," *W3C Architecture domain* http://www.w3.org/DOM/faq.html, 2003.

[19] M. Yoshikawa and T. Amagasa, "XRel: a path-based approach to storage and retrieval of XML documents using relational databases," *ACM transactions on Internet technology*, vol. 1, pp. 110-141, 2001.