The Journal of GIS Association of Korea, Vol. 12, No. 4, pp. 307-319, December 2004

A GML Data Storage Method for Spatial Databases

Ho-young Jeung* · Soo-hong Park**

ABSTRACT

Managing GML data in traditional database systems is not efficient since it has not only characteristics of spatial data but also features of (semi) structured XML documents. XML enabled database systems can manage XML data efficiently, however they cannot handle spatial data. Spatial database systems are good at spatial data handling but those are inefficient for XML data. This paper proposes a storage method of GML data for spatial database systems in order to solve the problems. The proposed method generates spatial database schemas from GML application schemas and store GML data into SDBMS through the generated schemas. A prototype of the storage method has been implemented on the PostgreSQL/SPE system to show the proposed method is appropriate for storing GML data. As a result, the implemented system was able to store various GML data which had diverse XML structures and different size. Stored data size was smaller than GML files. Furthermore, spatial, non-spatial, and mixed content queries could be performed over the stored GML data as quickly.

Keywords : GML, Spatial Databases

요 약

GML 데이터는 일반적인 문자, 숫자 형태의 데이터와는 다르게 공간데이터의 특징과 (준) 구조적인XML 데이터의 성격을 동시에 지니고 있어 표준 데이터베이스에서 관리되기 힘들다. XML 저장이 가능한 데이터베이스는 GML 데이터를 효율적으로 저장할 수 있지만, 공간데이터 처리 능력이 부족하고, 공간데이터베이스는 XML 데이터를 저장하기 어렵다. 본 논문에서는 GML 데이터를 공간데이터베이스에 저장하여 기존의 문제점들을 해결하고자 한다. 이를 위하여 GML 응용스키마로부터 OGC에서 제시한 공간데이터베이스의 스키마로 변환할 수 있는 방법을 제안하고 PostgreSQL/SPE 시스템을 기반으로 프로토타입 시스템을 구현하였다. 그 결과 다양한 기하 모델과 XML 문서의 구조 정보를 포함하고 있는 GML 데이터들이 제안된 기법을 통하여 공간

* Master Studnet, Department of Geoinformatic Engineering, College of Engineering, Inha University, jeung.hy@gmail.com
** Assistant Professor, Department of Geoinformatic Engineering, College of Engineering, Inha University, shpark@inha.ac.kr

데이터베이스에 저장될 수 있었고 저장된 데이터의 크기는 GML 파일로 존재할 때보다 현격하게 적은 공간을 차지하였다. 또한 저장된 데이터에 대하여 공간, 비 공간 및 혼합 질의를 수행하여 저장된 GML 데이터들이 빠르게 검색되고 복잡한 질의가 손쉽게 수행될 수 있음을 보였다.

주요어 : GML, 공간데이터베이스

1. INTRODUCTION

1.1 Background and Research Objectives

Extensible Markup Language (XML)[1] is emerging as the standard of data exchange among web applications and Geography Markup Language (GML)[2][3] is getting more popular to web-based Geographic Information System (GIS) applications likewise. Most of the GIS applications store geographical data into database systems. Hence, storing and querying GML data from database systems are very important to support the spatial data exchange trends through the web.

Spatial database management systems (SDBMS) designed to handle very large amounts of spatial data stored, using specialized indices and query-processing techniques[4][5]. Therefore, stored GML data in SDBMS could be very efficient to handle spatial information. However, storing GML into SDBMS is a very hard work since SDBMS are not designed to handle (semi) structured data like XML. Recently, many mechanisms which XML documents are able to be managed by databases have been studied. Despite the many researches, those studies are not suitable for GML documents because GML data has several specific characteristics unlike XML documents such as geometric attributes of features.

Therefore, there are a couple of possible solutions to manage GML data in database systems. One gives spatial functionalities to XML storable DBMS and the other makes SDBMS XML-enabled. This study selects the latter and describes how GML can be stored to spatial databases and performed both of spatial and non-spatial queries.

The primary goal of this study is to present a storage technique for using spatial databases to store, query, and manage GML data. In order to achieve this goal, this research brings strategies of XML storage into spatial database domain and develops a prototype system which integrates the special needs of GML data into spatial databases. In addition to proving that the proposed technique is possible, this study experiments various spatial and non-spatial queries over the stored GML data. Specifically, the objectives of this research are :

- ▸ to select suitable XML storage strategies for GML;

- to define mapping rules from GML to spatial databases;
- to implement the above strategies on a SDBMS;
- to explore the possibility of managing GML in the prototype system with the result of query evaluating;

1.2 Related Work

Corcoles and Gonzalez evaluated several XML query languages and identified key features required for specifying spatial queries over GML documents[6]. They showed that non-spatial queries could be directly answered with XQuery[7] but spatial queries required a rich set of topological predicates and spatial analysis functions. They proposed a GML query language called GML-QL which extended the XQuery to support spatial query operations.

They also studied the behavior of different alternatives over XML documents applied to store and query GML documents[8]. The alternatives selected use relational schemas to store GML data and three approaches (one structure-mapping, two simple model mappings) have been used. The experimental results showed that the structure-mapping approach was efficient for their previous study.

Shrestha had large volume of research in XML technologies for GML storage[9]. The study experimented many approaches of XML storage using both of structured storage and unstructured storage in XML-enabled systems, as well as native-XML databases like hybrid storages. He executed non-spatial queries over GML and tested XML-specific functions like transformation, construction of new elements, document retrieval and so on. However, he found spatial queries can not be executed and it needed to develop GML query processors in XML database systems.

Most of the previous researches in this section have been based on XML storable database systems to manage GML data. The key difference between this research and the previous researches is that this study proposes a new approach using a spatial database system.

2. Selecting GML Storage Strategies

2.1 XML Document Management Models

There have been many techniques developed for XML document management. These techniques are able to be divided into four primarily management models which are file systems, general database systems, native XML database systems, and XML extensions of traditional databases.

XML documents are often stored in file systems and it is the easiest way to store but it lacks fast and efficient support for large XML files. This model is hard to have indexing or external searching even though XML document access tools are a lot. Traditional database technology brings many benefits to XML document management (e.g., recovery, concurrency control, and high level query capabilities). However, it has been designed to handle semi-structured XML data.

Native XML databases are databases designed especially to store XML documents. Like other databases, they support features traditional databases have. The only difference from other databases is that their internal model is based on XML and not some others, such as the relational model. XML-enabled systems are databases with extensions for transferring data between XML documents and themselves. Some of these also support native XML storage.

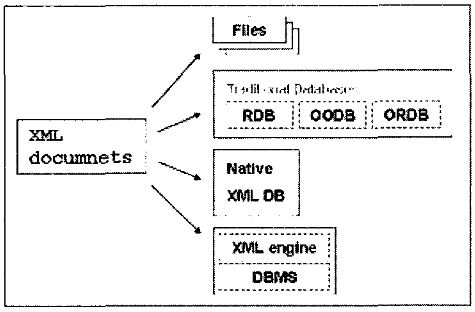

Figure 2. XML management models

Since spatial databases are built on relational/object-relational database technologies, this study focuses on the traditional database model as an XML management model.

2.2 XML Storage Approaches for GML

The performance of database systems is highly affected by storage approaches when XML data is stored. These storage approaches can be categorized into three parts, as follows:

- A view of XML : data-centric and document-centric;
- Designing database schema : structure-mapping and model-mapping;
- Database models : relational, object-relational, and object-oriented;

XML documents can be either document-centric or data-centric. Document-centric involves a liberal use of free-form text that is "marked up" with elements. On the other hand, data-centric documents are typically easier to process with computer programs because the data is better organized. In common, since GML data is created by generating existed GIS data sources, not by hand writings, GML documents should typically be data-centric XML.

In the view of designing database schema, structure-mapping approach generates database schema from the logical structures of target XML documents[10]. Basically, a relation or class is created for each element type in the XML documents[11]. Furthermore, database schemas are designed based on detailed analysis of DTD[12] or XML Schema[13][14] through mapping rules[15][16][17]. In the contrast to the structure-mapping, model-mapping approach uses a fixed database schema for all kind of XML documents[10][16][18][19]. The fact that GML has predefined GML schemas allows application based on GML to estimate what structural data is coming in. This is remarkably distinguishable from generic XML data. Even though structure-mapping approaches are difficult to be implemented, those can be a more efficient storage model[20][21] especially if databases already know the structures of target. We can analyze application schemas where has full information of instance GML data. Hence, structure-mapping strategy is the better approach to spatial databases.

Relational database model may produce database schema with many relations in order to store XML documents. It may also potentially have many joins, which would make the queries expensive to evaluate[5][20]. In the case of object-relational model, it allows the use of structured or nested attributes in relations. These characteristics can be used to map XML documents onto databases in a more natural way[16][22][23]. Moreover, this model can reduce table join that involves much cost as mentioned and improve overall performance of database systems. Spatial DBMS standard[24] is not to specify database schema for the object-oriented database model. Therefore, object-relational model is the best choice for this study.

3. Deciding Spatial Database Schemas

3.1 Conceptual Schema Mapping

An entity, in the Entity-Relationship (E-R) model, is a "thing" or "object" in the real world that is distinguishable from all other objects[26]. It is represented by a set of attributes that have domain, or value set to permit values. An entity set is a set of entities of the same type that share the same properties, or attributes. A relationship is an association among several entities. Likewise E-R concepts, some of vocabularies are defined in the GML specification[3]. An object is defined as an entity with a well defined boundary and identity that encapsulates state and behavior. Association is a structural relationship that describes a set of links, in which a link is a connection among objects.

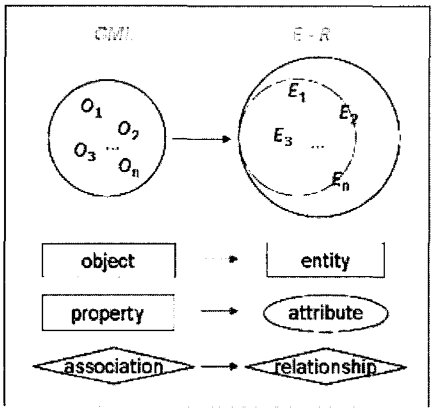

Figure 3. Conceptual schema mapping

In two conceptual models, since each definition has similar concepts and models world phenomena, some assumption can be defined between them:

- A collection of objects is mapped to an entity set;
- An object is mapped to an entity;
- A property is mapped to an attribute;
- An association is mapped to a relationship;

On the other hand, all of theses assumptions may not be realized at logical level, or physical level since XML model and relational/object-relational model are very different logical and physical structures. Hence, some special mapping mechanism is needed.

3.2 Mapping Rules

Followings are the summary of the mapping rules from the previous study[23] :

- Simple elements can be mapped directly into database attributes;
- Sequences of elements are translated into a tuple;
- Elements with attributes are mapped into a database composite attribute;
- XML elements that are optional result in database attributes may have NULL values;
- Set-valued attributes are transformed into a database set-valued attribute;
- Repeatable elements which can occur zero, one, or several times are transformed into a set-valued database attribute;

This research proposes GMLType as an UDT and GMLValue as an UDF. Those allow that object-relational databases can store XML elements with complex content model.

- Complex combinations of elements are transformed into a GMLType attribute;
- Mixed-content type model are transformed into a database attribute with GMLType;
- Alternatives are transformed into a database attribute with GMLType;

Since GML properties do not have other properties as children, one GML property is transformed into one database attribute. However, set-valued database attribute should be allowed in the case of a GML property has multi values. Table 1 describes the rules of mapping from GML simple types which are defined in GML schema file, basicTypes.xsd, to database schemas.

- τ describes XML type defined by GML 3 schema.
- $\mu()$ is a function which evaluates the mapping rules.
- {} represents a set constructor or the list constructor.

Complex GML objects/properties can be mapped to another GML object/property created by evaluating mapping rules. It means that mapping rules can be evaluated for many times recursively and finally the initial GML object/property shall meet the GML simple content model mapping.

Table 1. GML simple type mapping

GML Simple Types	Database Attribute Domain
τ = NullEnumeration	$\mu(\tau) \rightarrow$ NULL
τ = NullType	$\mu(\tau) \rightarrow$ NULL
τ = booleanOrNull	$\mu(\tau) \rightarrow$ BOOL
τ = booleanOrNullList	$\mu(\tau) \rightarrow$ {BOOL}
τ = booleanList	$\mu(\tau) \rightarrow$ {BOOL NOT NULL}
τ = stringOrNull	$\mu(\tau) \rightarrow$ STRING
τ = NameOrNull	$\mu(\tau) \rightarrow$ STRING
τ = NameOrNullList	$\mu(\tau) \rightarrow$ {STRING}
τ = NameList	$\mu(\tau) \rightarrow$ {STRING NOT NULL}
τ = doubleOrNull	$\mu(\tau) \rightarrow$ DOUBLE
τ = doubleOrNullList	$\mu(\tau) \rightarrow$ {DOUBLE}
τ = doubleList	$\mu(\tau) \rightarrow$ {DOUBLE NOT NULL}
τ = integerOrNull	$\mu(\tau) \rightarrow$ INTEGER
τ = integerOrNullList	$\mu(\tau) \rightarrow$ {INTEGER}
τ = integerList	$\mu(\tau) \rightarrow$ {INTEGER NOT NULL}
τ = CodeType	$\mu(\tau) \rightarrow$ STRING
τ = CodeListType	$\mu(\tau) \rightarrow$ {STRING NOT NULL}
τ = CodeOrNullListType	$\mu(\tau) \rightarrow$ {STRING}
τ = MeasureType	$\mu(\tau) \rightarrow$ DOUBLE
τ = MeasureListType	$\mu(\tau) \rightarrow$ {DOUBLE}
τ = MeasureOrNullListType	$\mu(\tau) \rightarrow$ {DOUBLE}
τ = CoordinatesType	$\mu(\tau) \rightarrow$ GEOMETRY
τ = SignType	$\mu(\tau) \rightarrow$ CHAR(1)

In case of geometry type, there are some differences between GML 3 Geometry and SQL Geometry which has only linear simple types. Thus, other GML 3 Geometry types except the simple types are transformed to GMLType where stores the values into XML elements.

4. Prototype System

The SPE (Spatial Processing Engine) has been developed by department of Geomatics, Inha University for years to research spatial database technology. That is a kind of a server extension that manages geospatial data in the PostgreSQL server[25]. The SPE system is a suitable back-end to apply this study since it has implemented most of the components defined by OGC standard [24]. The libXML parser as an XML/GML parser has been chosen in order to extend the SPE/PostgreSQL server to be XML-enabled. The GML extension of the prototype system is organized into 3 major software components, (GMLAnalyzer, MapGen, and Loader), which are further broken down into smaller logical sub-modules. They work in sequence as following 6 steps :

Step 1. UDF, UDT, and entire system components are set up as soon as the prototype system is installed to PostgreSQL/SPE.

Step 2. GMLAnalyzer which contains a GMLParser as a sub-module, starts to work automatically for building pre-organized mapping information from GML schema files.

Step 3. GMLAnalyzer parses GML application schemas and creates an Feature_ MAPPING table where includes mapping information.

Step 4. MapGen analyzes both of the Feature_MAPPING table and the

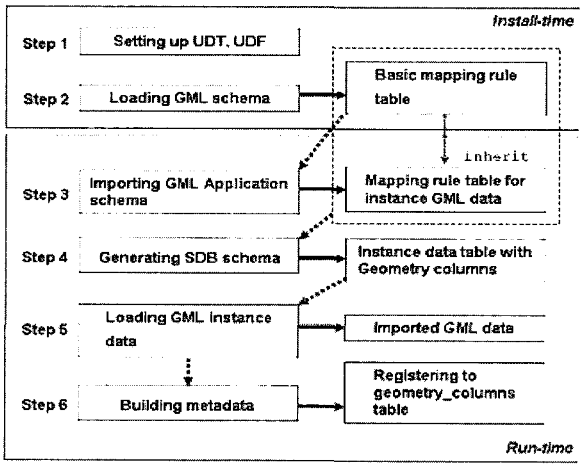

Figure 3. Procedure of the prototype processing

MAPPING_RULES table and generates table schemas for instance GML data.

Step 5. Loader parses instance documents and store the values of GML objects/properties into the Feature table. It references to the mapping information MapGen produced.

Step 6. Loader produces metadata such as envelope, name, and description which are GML properties, and register the information to the GEOMETRY_COLUMNS table.

Figure 4 shows extended spatial database schemas of the prototype system. Bold characters represent extension of the schemas in the OGC specification[24]. MAPPING_ RULES table is created by installing SPE to PostgreSQL. It contains mapping rules that XML types have in GML schema files. At the install time, a component module of GML processor parses all information GML schema has and builds the table with basic mapping information. The Feature_ MAPPING table is constructed at the running time. It sets up all mapping information by paring GML application schema files, as well as the table brings the pre-organized mapping information into the Feature_MAPPING table by inheriting the MAPPING_RULES table.

5. Test Study

This study used 12 various GML instance files (D1 to D12) and 2 application schema files (S1, S2) in order to evaluate queries. Some were from GML 3 specification (cambridge.xml and city.xsd) and the others came from a conference, named "GML Days 2004"[27] where

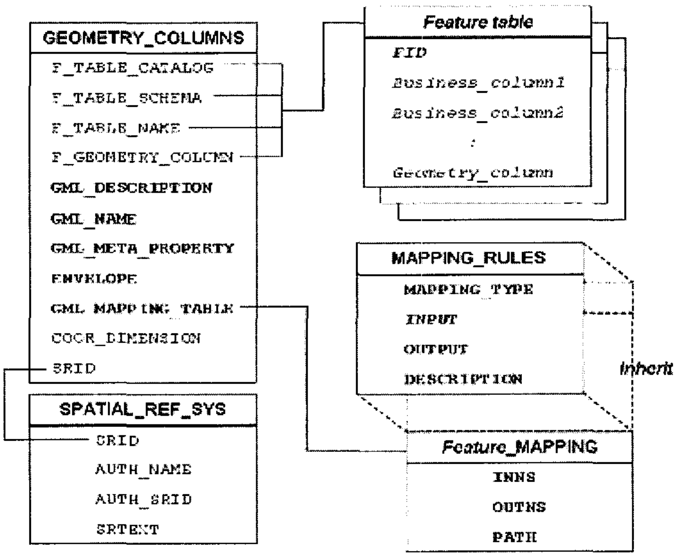

Figure 4. Extended spatial database schema

many GML experts presented. cambridge.xml is the GML instance document of city.xsd. vancouver.xsd is a GML application schema of the other data files (D2 to D12).

5.1 Produced Database Schemas

Figure 5 illustrates a database schema which generated from vancouver.xsd, building.xml, airport.xml, and roads.xml. This database schema has a key feature which three GML data share an application schema and it represents that the application schema was written for public purposes. Three tables (BUILDINGS, AIRPORTS, and ROADS) have same table schema for each other but each table stores geometric information into different Geometry columns (POSITION, CENTERLINEOF, EXTENTOF). Thus, each table has two NULL attributes respectively. In the contrast of the previous mapping, this case can be often happened because mostly GML data has similar structures. Since this is appeared to traditional database systems, too, this does not be thought as a problematic case and the performance and storage size is not affected by it as mentioned above.

There is another example to represent a special case. CITY_MAPPING table contains the structural information of GML instance, cambridge.xml. The structural information comes from GML application schema, city.xsd, as well as inherits the MAPPING_RULES table where includes mapping rules of GML

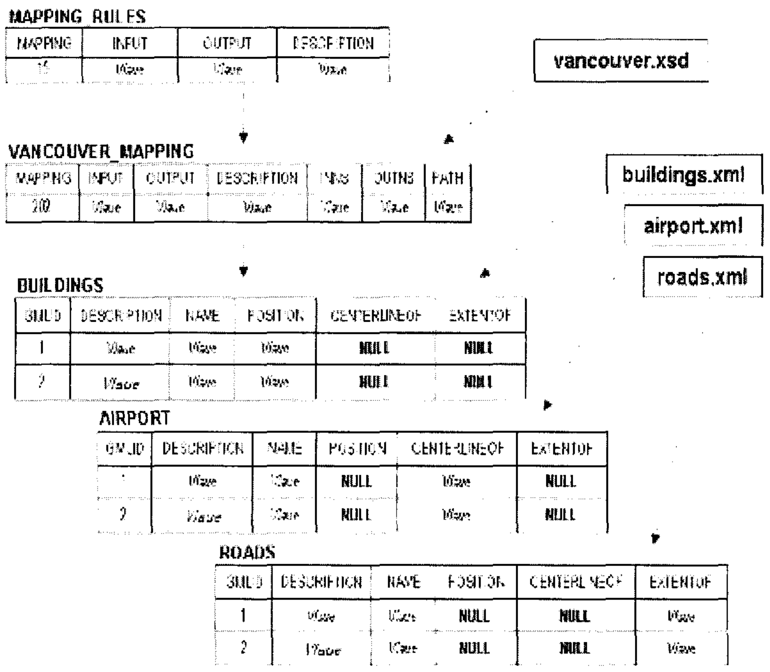

Figure 5. Produced table schema from vancouver.xsd

schemas. The loader loads GML instance data to CAMBRIDGE table using the information of CITY_MAPPING table. ambridge.xml has only two feature members, as GML Properties, however, the feature members have totally different XML structures although those share a application schema file. This case was one of the worst scenarios GML data may have. Cases like this can be appeared when document-centric (see 2.2) data comes into a table through the mapping rules. However, since GML data is obviously data-centric XML, rarely could these cases be showed in practice.

This study has described only a couple of cases but all test data has been stored to the prototype systems without reporting any problems. Without considering for performance, like elapsed storing time, the proposed storage method can be thought as a proper strategy to store GML data.

5.2 Storage Size

This study has also experimented with storage size. Table 2 shows the comparison between two storage models and Figure 13 illustrates the tendency as growing the raw data size.

In general, relation size is much smaller than GML size, even smaller than half. In particular, a couple of difference tendency are shown, one is that GML size is smaller than relation and the other is the opposite.

Case 1 : Relation size > File size - It is when GML data is smaller than 8 kilo bytes. Since the PostgreSQL/SPE system uses 8K block(page) size, the prototype system has at least 8K size for any other data. GML data usually has much larger data size, otherwise databases technologies are useless for GML.

Case 2 : Relation size < File size - The tendency of this case is more remarkable when GML data contains much of positional data like coordinates. The prototype system/SPE stores the coordinates to LOB types and it allows such case data to be much smaller. Moreover, the prototype system does not store the additional information such as XML comments, and empty spaces the stored size can be smaller. In addition, DBMS commonly use data compression, especially for textual data.

Table 2. Storage usage [bytes]

Data	GML	Relation	Ratio G : R
D1	2,105	8,192	389 %
D2	854	8,192	959 %
D3	4,088,526	1,736,704	42 %
D4	4,086	8,192	200 %
D5	104,291	40,960	39 %
D6	138,327	8,192	39 %
D7	10,850	8,192	75 %
D8	497,087	245,760	49 %
D9	192,515	98,304	51 %
D10	15,918,470	7,774,208	48 %
D11	277,526	131,072	47 %
D12	44,909	24,576	55 %

5.3 Query Evaluation

Table 3 shows the descriptions and features of the query templates. At the "Query" column, N1, N2, and N3 describe that those are non-spatial queries. S1, S2, S3, and S4 are spatial queries. M1 denotes mixed queries. The words written in italics and bolds represent spatial functions/operations that spatial databases have.

In general, S1 through M1 were very complex queries needed many geometric calculations. Despite the complexities, query response times were under one second except only 4 responses. Furthermore, the most remarkable thing is that all queries were performed against all of the test data without any problems. This implies highly important meaning to overall this study, since this research has great interest of spatial abilities over stored GML data.

6. Conclusion

This paper has proposed a storage method which extends a spatial database to store and query GML data. For this proposal, many mapping rules from GML application schema to spatial database schema were defined as well as suitable XML storage approaches for GML data ware selected. A prototype system was implemented to verify the method. In addition, spatial, non-spatial, and mixed queries are experimented over the stored GML data which had various structural information, many geometry types, and diverse data size. As a result, various GML data could be stored to the prototype system. Also, complex queries including spatial

Table 3. Query templates

Query	Description	Feature
N1	Select all features	plain query
N2	Obtain idenfiers where the legnth of description is longer than 20	simple aggregate
N3	Return all columns, after self join with description	performance test purpose
S1	Select *geometries(extentOf)* where *bounding area* of the place are *greater* than 50,000,000	geometric operator
S2	return both *binary values* and *texture value* of the Geometry columns where description contain 'builtUp' string	WKB, WKT constructing
S3	Count the geometries where *envelope* of the geometry cross the *boundary* of those	spatial relationship
S4	search line features which *bounding boxes* spatially does not *equal* to convexhull geometry of the *bounding boxes* and return the value as *binary*	spatial analysis with spatial relation
M1	After self join, count the number of features where a geometry *spatially equals* to the other and one *intersects* to the other	mixed query

asking were simply and quickly executable.

This study has shown that it is possible to handle most queries on GML data using a spatial database, barring XML queries based on path. The potential advantage of this approach is that spatial databases demonstrate great possibilities to manage GML data. In contrast, this was not possible with XML enabled databases[9].

There are several avenues for future works. This research has not defined operations for GML since we have thought that spatial databases have already "good" operations. Therefore, how to store GML has been the primary issue. However, GML 3 does not include only simple object model but also many unexplored domains such as topology, coverage, dynamic features, units of measure, and so on. This study is only one initial step towards finding a good way to manage GML data. One answer against how to manage various kinds of GML data efficiently, is using advanced databases technologies. When we try to find out the methodology that GML data is stored to any database systems this study will be a good case study.

References

[1] T. Bray, J. Paoli, and C. Sperberg-McQueen, 1998, "Extensible Markup Language(XML) 1.0", Technical report, W3C Recommendation.

[2] S. Cox, A. Cuthbert, R. Lake, and R. Martell, "Geography Markup Language (GML) Implementation Specification, version 2.1.2", OpenGIS Consortium, 2002.

[3] S. Cox, P. Daisey, R. Lake, C. Portele, and A. Whiteside, "Geography Markup Language (GML) Implementation Specification version 3.0", OpenGIS Consortium, 2003.

[4] P. Rigaux, M. Scholl and A. Voisard, "Spatial Databases With Application to GIS", Academic Press. 2002, pages 21-26.

[5] Shashi Shekhar, Sanjay Chawla, "Spatial Databases a Tour", Prentice Hall, 2003, pages 10-11.

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

[7] S. Boag, D. Chamberlin, F. Fernandez, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu, "XQuery 1.0: An XML Query Language", W3C Working Draft, 2002.

[8] J. E. Corcoles, P. Gonzalez, "Analysis of Different Approaches for Storing GML Documents", In Proceedings of the Tenth ACM International Symposium on Advances in Geographic Information Systems GIS'02, 2002, pages 11-16.

[9] Bikram Bahadu Shrestha, "XML Database Technology and its use for GML", www.itc.nl/library, 2004, pages 14-15, 41-63, 93-96.

[10] M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura, "XREL: A Path-Based Approach to Storage and Retrieval of XML documents using Relational Databases," Proceeding ACM Transactions on Internet Technology, Volume 5. 2001.

[11] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl., "From Structured Documents to Novel Query Facilities". In Proceedings of the ACM SIGMOD International Conference

on the Management of Data. 1994, pages 313-324.

[12] J. Bosak, T. Bray, D. Connolly, E. Maler, G. Nicol, C. M. Sperberg-McQueen, L. Wood, and J. Clark, "W3C XML Specification DTD", Technical report, W3C Recommendation, 1998.

[13] H. S. Thompson, David Beech, Murray Maloney and Noah Mendelsohn, "XML Schema Part1: Structures", W3C Recommendation, 2001, http://www.w3.org/TR/xmlschema-1.

[14] P. Biron and A. Malhotra, "XML Schema Part2: Datatypes", W3C Recommendation, 2001, http://www.w3.org/TR/xmlschema-2/.

[15] J. Shanmugasundaram, H. Gang, Kristin Tufte, Chen Zang, David DeWitt, and Jeffrey F. aughton, "Realtional Databases for Querying XML Documents: Limitaions and Opportunities", In Proceedings of the Conference on Very Large Databases. 1999, pages 302-314.

[16] M. Klettke and H. Meyer, "Managing XML documents in object- relational databases" Rostocker Informatik Fachberichte, 1999.

[17] C. Kanne and G. Moerkotte, "Efficient storage of XML data", In Proceedings of the International Conference on Data Engineering. 2000.

[18] D. Florescu and D. Kossmann, "Storing and querying XML data using an RDBMS", IEEE Data Engineering Bulletin. 1999, pages 27-34.

[19] T. Shimura, M. Yoshikawa, and S. Uemura, "Storage and retrieval of XML documents using object-relational databases", In Proceedings of DEXA, 1999.

[20] D. Florescu and D. Kossmann, "A Performance Evaluation of Alternative Mapping Schemas for Storing XML Data in a Relational Database", Technical Report 3680, INRIA, 1999.

[21] F. Tian, D. DeWitt, J. Chen, and C. zhung, "The design and performance evaluation of various XML storage strategies". Submitted for publication, Computer Science, University of Wisconsin, Madison, 1999.

[22] M. Klettke and H. Meyer, "XML and Object-Relational Database Systems-Enhancing Structural Mappings Based on Statistics". WebDB, 2000.

[23] P. Bohannon, J. Freire, P. Roy, and J. Simeon, "From XML Schema to Relations: A Cost-Based Apporach to XML Storage". 18th International Conference on Data engineering(ICDE2002), 2002.

[24] David Beddoe, Paul Cotton, Robert Uleman, Sandra Johnson, Dr. John R. and Herring, "OpenGIS Simple Features Specification for SQL Revision 1.1", OpenGIS Consortium, 1999.

[25] The PostgreSQL Global Development Group, "The PostgreSQL 7.2 Administrator's Guide", www.postgergl.org, 2002, pages x,xi,xii.

[26] A. Silberchatz, H. Korth, and S. Sudarshn, "Database System Concepts : Third edition", The McGraw-Hill Companies, Inc. 1996, pages 23-28.

[27] GML And Geo-Spatial Web Services Conference 2004, Vancouver, British Columbia http://www.gmldays.com/workshops.html, 2004 July.