STUDY ON NATIVE XML DATABASE BASED GML STORAGE MODEL

Shuliang Zhang[a, *], Jiayan Gan[a], Jiehui Xu[a],Guonian Lv[a]

[a]Key lab of Virtual geographic environment, Ministry of education, Nanjing Normal University ,210046,Nanjing，China - zhangshuliang@njnu.edu.cn,ganjiayan@126.com

KEY WORDS： GML, XML, Native XML Database, GML Storage, JTS, XQuery

ABSTRACT：

As a mature spatial interoperability specification of OGC, GML rapidly plays an important role in a large number of GIS research and application domains, such as spatial data modelling, transmission and exchanging, integration and sharing, because of its advantages of opening and self-description format system, rich spatial data expression technology, flexible application schema and so on. However, the complexity of GML specification leads to bulkiness of GML data files, which limits extent of GML data sharing. Furthermore, its non-structural data model goes against its storage by commercial relational database, which ultimately result in low performance of GML application and destroy abundant semantic of GML data. Thus, appropriate GML storage approach would improve extend and depth of GML application.By analogy approach, this paper study on storage and manage model of XML data, analyze current status of GML storage method. Combining XML database with characteristic of GML spatial data, create GML storage mapping mechanism based on feature and storage granularity, form GML to Native XML Database Storage Model (G2NXDBM) consisted of GML file segment, GML schema file and GML storage log file. On the basics of JAXP (Java API for XML Processing) program API and open source code JTS (Java Topology Suite), the author developed G2NXDBMs prototype system which include schema mapping constructor and file storage tools depending on Ipedo native database by IPedo Inc.USA. To test the efficiency of GML query, the author selects 5 groups of test data with the same size. The experiment result demonstrates that although the storage space of G2NXDBMs is larger than GML file, it has more advantages of query efficiency, support of XML technology and data management than GML file and relational database storage, it also testify the validity and practicability of G2NXDBM.

1. INTRODUCTION

As a mature spatial interoperability specification of OGC, GML rapidly plays an important role in a large number of GIS research and application domains, such as spatial data modelling, transmission and exchanging, integration and sharing, because of its advantages of opening and self-description format system, rich spatial data expression technology, flexible application schema and so on. However, the complexity of GML specification leads to bulkiness of GML data files, which limits extent of GML data sharing. Furthermore, its non-structural data model goes against its storage by commercial relational database, which ultimately result in low performance of GML application and destroy abundant semantic of GML data. Thus, appropriate GML storage approach would improve extend and depth of GML application. Although GML application increasingly manifold, current research on GML database and related technologies are on the germination status. Neither clear concepts nor architecture nor mature special product exists.

Earlier XML/GML data is stored as document (text file), is queried by information access methods such as key word query which is simple and fit for frequently document updating context. Since such storage approach lacks of systematic storage and query mechanism which leads to low query ability and can neither satisfy complex condition query nor optimize query.

Traditional relational database system is more mature on aspects of query, concurrency, security and other technologies, and extends function of processing XML/GML data. Due to the big difference between semi-structural XML data model and traditional relational model would bring problems on data storage and query. The problems are as follows: ①when store XML/GML data, traditional RDBMS doesn't support hierarchy and semi-structural data format, nested XML/GML data should be converted by mapping mechanism before stored in simple relational tables. Such approach not only reduces storage efficiency but also leads to information losing during data conversion; ②when query XML/GML data, XML/GML query requirement should be firstly converted to database query expression, such as SQL; Then query engine of database optimize query expression and generate query execution plan; finally convert query result to XML/GML data. Such approach resolves problems of query complexity in a certain extent, but too many conversions bring new problems of low efficiency and confusion of query semantic.

As increasingly mature of XML technology, XML databases have been broadly studied and applied, databases used for storing and managing XML data have come forth, namely native XML database [www.rpbourret.com, 2005]. Advantages of native XML databases are as follows[Liugang,2005]: ①effectively support XML character of self-description, semi-structure and sequence; ②provide "out and home access" for XML document, system store XML data directly, but not convert XML data to relational model or object orientated model, and store XML document in relational database or object orientated database. The native XML database can store XML document in native XML database and has capability to access the "same" document; ③directly support XML query

* Corresponding author.

language such as XQquery, XPath but not conversion to SQL or OQL (Object Query language); ④provide rich development interfaces. Most native XML database support XML:DB API of XML:DB.org development.

GML database as a tool for managing XML data with special format, its research could be advanced on the basis of current XML database technology combined with characters of GML. Combined with characters of native XML database and GML spatial data, this paper designed a more effective native XML database based GML spatial data storage model-G2NXDBM, develop and implement GML data storage prototype system, directly access GML data in native XML database. This paper also evaluates this storage model on aspects of storage space and query efficiency.

2. NATIVE XML STORAGE BASED GML STORAGE MODEL- G2NXDBM

2.1 Architecture of G2NXDBM

2.1.1 GML spatial data storage mechanism

The main approach of native XML database based GML data storage is: firstly analyses structure relationships between features according to GML application schema, then map structure relationship to collection of native XML database, and establish collection in database to reflect document structure.

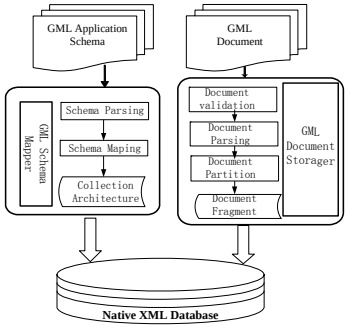

Figure 1. Storage Mechanism of GML Spatial Data

Assume that every GML document corresponds to at least one specific GML schema, this schema file has existed in system or provided by the owner of the GML document. When import a specific schema, system would generate a one to one mapping rule between schema and storage structure, and establish corresponding collection. Thus, when read a GML document, system would classify document by schema. And partition this document by unit of feature, then store the partitioned document in corresponding data collection according to mapping rule.

2.1.2 GML Spatial data storage structure

In this storage model, when GML data is stored in native XML database, the storage is implemented by three types of files, （1）GML fragment file; （2）GML schema file; （3）GML storage log file. GML fragment file is the fragments after partition imported GML document according to certain granularity. GML schema file is used to define content, structure and constrain rule of GML document. It could be used to conjecture storage mapping rule between GML document and native XML database and testify validity of GML document. GML storage log file mainly save partitioned GML document name, GML fragment storage position, fragment order and metadata about stored GML document such as storage time, storage person and so on.

Two root collections are defined in database, one is data collection, and another is management collection. Data collection is used for store GML schema and log file. In data collection, document belong to different schema correspond to different document collection. In the same document collection, the GML fragment file with the same type would be stored in the same feature collection. Also, the management collection is also stored according to classified schema. Its storage structure is shown as figure 2.

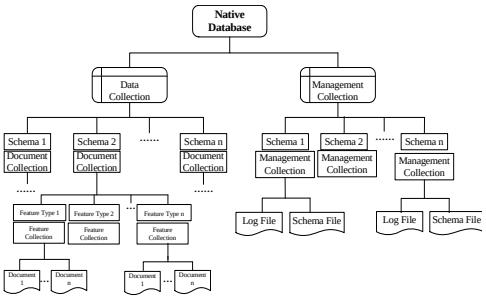

Figure 2. Storage Structure of GML Spatial Data GML

Native XML database based GML storage model has characters as follows:

(1)Partitioning storage granularity of GML document according to feature type not only has certain semantic repetition but also enhance storage efficiency.

(2)Record nested relationship between features by hierarchy of collections in native XML database.

(3) One of the most important characters of Native XML database is providing “out and home access” function, thus, storing GML document in native XML database can retake the “same” document.

(4) Store document with the same feature type in the same collection can accelerate query speed. Because querying on one collection would be faster than multi-collections.

(5) saving partition information and storage information in log file can facilitate later operations such as information access and document restore.

2.2 Storage granularity of G2NXDBM

Current native XML database could be classified to three types according to its storage granularity: element based, sub-tree based and document based. Lore[J. McHugh ,1997] and TIMBER[H. V. Jagadish,2002] adopt element based storage strategy, each element is the minimal storage unit; Natix[T. Fiebig,2002] adopt sub-tree based storage strategy, partition XML document into a series of sub-trees according to physic storage pages size, take each sub-tree as storage record; Xindice adopt document based storage strategy, take a XML document as storage record.

Two types of data stored in GML database system: one is GML schema file; another is GML document instance. GML schema determines storage structure of data, and document instance contain the true data content in system. GML schema defines content and structure of GML document, which could be adopted as the standard of GML documents classify and basis of storage granularity selecting. GML adopt feature to describe geographic entries and geographic phenomenon, data in GML document is organized by units of feature. Thus determine storage granularity of GML document model according to feature type is appropriate.

```
<schema targetNamespace="http://www.ukusa.org"
xmlns:app="http://www.ukusa.org"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ex="http://www.ukusa.org">
    <element name="City"
type="gml:AbstractFeatureCollectionType"
substitutionGroup="gml:_FeatureCollection"/>
    <element name="Road" type="ex:RoadType"
substitutionGroup="gml:_Feature"/>
    <element name="Bridge" type="ex:BridgeType"
substitutionGroup="gml:_Feature"/>
    <element name="Building" type="ex:BuildingType"
substitutionGroup="gml:_Feature"/>
    <complexType name="RoadType">
        <complexContent>
            <extension base="gml:AbstractFeatureType">
                <sequence>
                 <element name="roadName"
type="string"/>
                  <element name="roadCode"
type="string"/>
                  <element ref="gml:curveProperty"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="BuildingType">
        <complexContent>
            <extension base="gml:AbstractFeatureType">
                <sequence>
                 <element name="BuildingCode"
type="string"/>
                  <element name="BuildingHeight"
type="string"/>
                  <element ref="gml:Polygon"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="BridgeType">
        <complexContent>
            <extension base="gml:AbstractFeatureType">
                <sequence>
                 <element name="span"/>
                 <element name="height" type="integer"/>
                 <element ref="gml:centerLineOf"/>
                 <element ref="app:spans"
minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <element name="spans"
type="app:CurvePropertyType"/>
    <complexType name="CurvePropertyType">
        <sequence>
            <element ref="app:Gorge"/>
        </sequence>
    </complexType>
    <element name="Gorge" type="app:GorgeType"
substitutionGroup="gml:_Feature"/>
    <complexType name="GorgeType">
        <complexContent>
            <extension base="gml:AbstractFeatureType">
                <sequence>
                  <element name="width" type="integer"
minOccurs="0"/>
                  <element name="depth" type="integer"
minOccurs="0"/>
                  <element ref="gml:centerLineOf"
minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</schema>
```

Table 1. City.xsd (GML application schema instance)

When import GML schema, one of the key points is to confirm storage granularity of document instance according to schema. Granularity is the degree of data induction and aggregation, storage granularity determines size of fragments partitioned from each GML document instance in database. Fragment could be the whole GML document or single element. Table 1 shows GML application schema City.xsd that describes city, in this instance, city is the root element of GML document, city contains Road, Building, Bridge, Gorge sub-elements. For example, Road is a independent road, it's a integrated description of road, the sub-element of Road, such as roadName 、 roadCode 、 gml:curvePropert is specific description on a certain aspect of road. If take City as root element of partitioned fragment, the whole GML document would be encapsulated as an object which would be operated as a whole. And this object contains many integrated objects. To query or reuse GML document, such granularity is too large, which would increase cost on calling and operation. If take roadName、roadCode、gml:curveProperty as root element of partitioned fragment, they are only describe one aspect of object, it lacks of integrated meaning and reusing character. Moreover, query result returned by user could not be generally single element. Thus, appropriate granularity should be partitioned by feature type. The feature with the same type should be stored in the same disk file, for example, take Road, Bridge, Building, Gorge as root element of partitioned GML document instance fragment, the element content and its sub-element, as an integrated object , is stored in corresponding database.

When read a GML document instance, system read would firstly check the validity of GML document instance, after affirm that GML document is good format GML document, system would establish corresponding relationship between GML document and storage structure according to mapping rules, and decompose, save document instance. The instance document City.xml generated according to City.xsd would not be given in details due to the limitation of page. As mentioned above, features (Bridge, Gorge, Building, and Road) have been confirmed as root elements; so when it comes to Bridge, Gorge, Building or Road element, system would store element content and sub-element it contained into corresponding data collection as a whole fragment without parsing sub-element. Thus this approach does not change inner structure of element. It's still a legal GML document fragment. In other words, it's still XML format document that stored in database, in the database, a XML file is used for managing these document fragments. This instance includes two Bridge objects, one Gorge object, one Road object, one Building object. The feature object with the same type would be merged into one file, so this document would be partitioned into one Bridge fragment, one Gorge

fragment, one Road fragment and one Building fragment. These fragments would be enduringly saved in database as information resource of system to be used by other applications.

2.3 Storage Strategy of G2NXDBM

In the native XML database based GML storage model, when store GML data, four steps should be implemented:

(1)Import GML schema. Parse this schema, map feature object and document structure, and establish corresponding collection in native XML database.

(2)Import GML document. First, validate if corresponding GML schema document exists, if not, import schema file. Then verify GML document structure according to this schema, check if each element in GML document instance correspond to relevant schema structure and give checking result.

(3)Partition GML document and store it into corresponding document collection. GML document is partitioned by unit of feature after parsing, store partitioned GML fragment according to feature type, and the feature with the same type is stored in the same disk file.

(4) Store log file recorded document partition information and GML application schema file into corresponding management collection of database.

2.3.1 GML Schema Mapping

GML schema mainly describes structure of GML document and relationships between elements. In this paper, GML document storage granularity the author recommended is feature. Then, when GML document is partitioned into several GML fragments by unit of feature and stored in native XML database, how to express relationships between elements? In fact, native XML database use collection to manage document, its function corresponds to table in database and file in file system. One document collection aggregates one type of document which facilitates user operation. Many collections could be created and managed at the same time. Collections could be deployed by levels. One collection could contain several sub-collections, or contain several XML documents. Thus, user can create collection according to document structure defined by GML schema, feature with different types belong to different collection, and hierarchy of collection could be used to express subordinative and nesting relationship between features. Mapping GML application schema, first identify feature object and its document structure in this schema, then find mapping relationship with database and establish corresponding data collection. By mapping, document collection directory path of instance City.xml includes City \ Bridge \ Gorge, City \ Road, City \ Building.

2.3.2 Storage of GML Document

When store GML data, first need to check if corresponding GML application schema exists, if not exist then import schema file and verify document according to schema. Then partition document corresponding to schema requirement and cluster features with the same type into the same physical page, finally store them in corresponding collection in native XML database. GML document is partitioned by unit of feature after parsing, store partitioned GML fragment according to feature type, and the feature with the same type is stored in the same disk file. Then how to quickly recognise features with the same type and store them in the same disk file? This paper adopts approach of clustering and classifying.

The basic idea of clustering and classifying is to partition semantic block according to schema structural graph, each semantic block describe an integrated logical unit. Then cluster all instances of the same semantic block. The advantage of this approach is mainly embodied on three aspects: ①increase storage efficiency. When store GML document, if only purely partition GML document according to feature, it may leads to largeness of number of partitioned files and largely decrease storage efficiency. Therefore, cluster feature with same type in the same file would decrease number of files and enhance storage efficiency; ② Easy to query. By aggregating features with same type in the same disk file, feature could be accessed directly and decrease I/O times the disk need, when user need to query content of certain feature; ③ Easy to retrieve document later. Since features with same type cluster in the same file, unnecessary conversion and parsing would be avoided when reorganize document.

In this paper, storage granularity of GML document is on basis of feature type, certain feature describes an integrated logical unit which could be expressed as a semantic block. In GML schema, elements in GML document could be clearly expressed by nested model, thus it's reasonable to construct semantic block for GML document according to GML schema. Aiming at City.xsd in this paper, we can conclude that Bridge and its sub-element, Road and its sub-element, Building and its sub-element form its respective semantic block: Bridge (span,height,gml:centerLineOf,spans),Road(roadName,roadCode,gml:curveProperty),Building(BuildingCode,BuildingHeight,gml:Polygon),Gorge(width,depth, gml:centerLineOf).

One record of such storage strategy is an instance of semantic block. In figure 3, record R1, R2 are two instances of Bridge(span, height, gml:centerLineOf, spans), record R3 is instance of Road(roadName, roadCode, gml:curveProperty), record R4 is instance of semantic block Road(roadName, roadCode, gml:curveProperty), record R5 is instance of Gorge(width, depth, gml:centerLineOf). In this way, when store GML practically R1 and R2 are stored in the same physical page, R3, R4, R5 might be stored in respective physical page. Thereby, four fragments-Bridge,Gorge,Building and Road would be generated, extraction of fragment is on basis of application schema and data file. Since extraction method is relatively simple, it would not be listed in details.

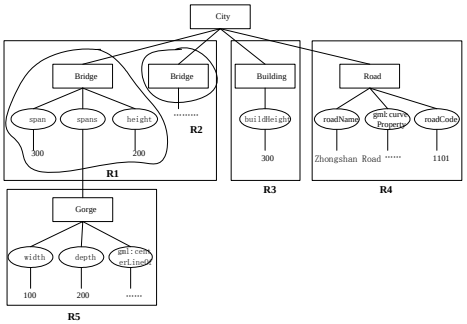

Figure 3. Record of City.xml

2.3.3 Storage of log file

When store GML document, documents correspond to different schema is stored in different document collection, GML document with same schema is stored in the same document collection. In other words, it's possible that several data files with the same schema could be partitioned into several GML fragments and stored in the same document collection, and then GML fragments from multi-original data file could be stored in the same document collection, consequently leads to failure of

find the corresponding GML fragment when retrieve original data file. In addition, structure of document such as relationship and order between elements might be disarranged. Thus, these factors certainly will bring difficulties to management and retrieve of document data.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="root_LogFiles"
type="FileCollectionType"/>
    <xs:element name="WholeFile"
type="WholeFileType"/>
    <xs:element name="SegmentFile"
type="SegmentFileType"/>
    <xs:complexType name="FileCollectionType">
        <xs:sequence>
        <xs:element ref="WholeFile" minOccurs="0"
  maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="WholeFileType">
        <xs:sequence>
        <xs:element ref="SegmentFile" minOccurs="0"
 maxOccurs="unbounded"/>
            <xs:element name="Store_Time"
type="xs:string"/>
            <xs:element name="Store_Member"
type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="SegmentFileType">
        <xs:sequence>
            <xs:element name="SegmentFileName"
type="xs:string"/>
            <xs:element name="ColletionPath"
type="xs:string"/>
            <xs:element name="ChildSegmentFileName"
type="xs:string" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>
</xs:schema>
```

Table 2. log schema file

Then, how to manage these GML documents stored in database? The approach this paper adopted is to define a storage log file for each partitioned GML document after stored in database. Data file with the same schema corresponds to one storage log file, this log file save relevant storage information for each GML document stored in database, includes: GML fragment name, GML fragment storage position, fragment order and metadata about GML document storage such as saving time, saving person and so on.

Since information of each log file is basically fixed. Thus, it's necessary to define a standard log schema file. In this way, log file could be customized according to schema file. We use XML format to describe log file, in the same way, we also use XML format to describe standard schema file.

Definition of storage information for fragment document as follows, element SegmentFileName indicates fragment file name, element ColletionPath indicates path the fragment file stores. Element ChildSegmentFileName indicates file name of features that would be nested by feature corresponds to this fragment.

Definition of storage information for the whole GML document as follows, element SegmentFile indicates relevant information of partitioned fragment file, element Store_Time indicates store time of this document, element Store_Member indicates the store person of this document. In addition, extend relevant information according to requirement.

FileCollectionType indicates storage information of different data file with the same schema.

According to definition of above nodes, log schema file could be defined as following table. Thereby define log file by this schema file.

Thus, when store instance document City.xml, user can define a log file to store information of document partition according to this log schema file, consequently facilitate later operation and retrieve. Log file of City.xml generated on the basis of log schema file is omitted in this paper. On this basis, log file of city.xml and city.xsd schema file could be stored in corresponding management collection. In this way, GML instance document, GML schema file and storage log file could be stored in native XML database and implement native based storage of GML data.

3. DESIGN AND IMPLEMENT OF G2NXDBM PROTOTYPE SYSTEM

3.1 Design of G2NXDBMs Framework

This constructed G2NXDBMs prototype system, Figure 4 shows frame sketch map of this system architecture.

G2NXDBMs prototype system mainly implements storage management on GML data, GML schema and metadata by two modules: GML storage and GML query.

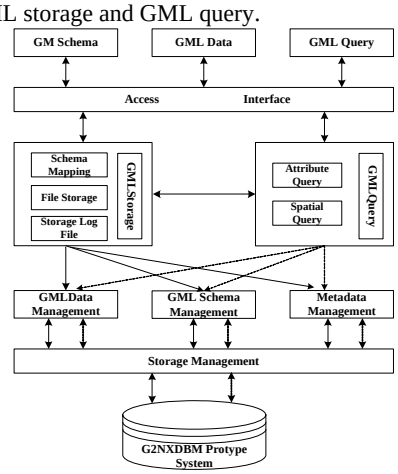

Figure 4. Architecture of G2NXDBMs

Design idea and function of GML storage module could be described as follows: ①analyze GML application schema document, generate one to one mapping rule between schema and storage structure; ②establish relevant collection in

database according to mapping rules; ③analyze GML data document, partition document according to storage granularity; ④store partitioned GML fragment document in native database.

GML query module mainly contain two sub-modules: attribute query and spatial query. Attribute query implement query by directly using query language XQuery, XPath carried by native XML database; combined with spatial data type and spatial operation operator, spatial query implement spatial analysis and query function by extending XQuery language in native XML database.

Currently, native prototype system mainly has functions of GML storage and GML query, and has certain GML data interface and relevant management interface which would make a stable basis for future function extension

3.2 Implement of G2NXDBMs

3.2.1 Choosing of development technology and platform

This prototype system chooses Java as system development language, and adopts JBuildinger2005 produced by Borland Company as system integration platform. As well as we know, many characters of java make it fit for XML programming language. In addition, JAXP (Java API for XML Processing) provides programming interfaces for Java to process XML document, this make it convenient for us to process XML document by using Java language.

This system use JTS(Java Topology Suite) open source code when implement spatial analysis and query function, this is a Java API about spatial prediction and function, it also comply with Simple Feature Specification For SQL1.0 published by Open Geospatial consortium (OGC), and provide an integrated, consistent and strong implement of basic 2D spatial algorithm. Its main characters includes: quick and exact spatial operation; strong algorithm of spatial operation; using clear and exact model in operation process; implement spatial operation of all basic geometries[http://www.vividsolutions.com,2006][Harrie, 2006]. Besides standard query syntax and strong query capability, JTS would decrease development load of prototype system and make author transfer research point to GML storage method and extraction strategy of native XML database.

At present, there are 30 types of mature commercial and open source products aiming at native XML database, such as Xindice[http://xml.apache.org,2006],IPedo[http://www.ipedo.com.cn,2006],eXist[http://exist.sf.net ,2006],OrientX[http://idke.ruc.edu.cn,2006],Tamino[http://www.softwareag.com,2006],X-Hive/DB[http://www.x-hive.com, 2006] and so on. This paper selects Ipedo database produced by American Company-IPedo as GML document storage platform. Choosing of Ipedo mainly depends on its three supporting advantages of Java and XML: It's a mature commercial native XML database which provides complete database management function and certain assurance of performance; it is implemented by pure Java and provide rich Java interfaces which is propitious to second development; it support XML standard of W3C, such as XSL, DOM, XQuery and so on.

3.2.2 Experimentation result and analysis

Experimentation platform of G2NXDBMs prototype system is P4 2.4G, main memory is 512MB, Hard disk is 80G, WindowsXP operation system.

Using different scale of GML data document to study storage and query efficiency of this system, this experimentation basically compared two types of storage method, one is GML document file, and another is G2NXDBM prototype system. Experimentation compared space required by these two storage methods, select different size of testing data, experimentation result is shown as Table 3. This table indicates that G2NXDBMs prototype system take larger storage space than GML document.

Storage Method	File1 (kb)	File2 (kb)	File3 (kb)	File4 (kb)	File5 (kb)
GML Document	330	566	1,157	2,331	4,727
G2NXDBM Prototype System	452	758	1,509	3,046	5,431

Table 3. Comparison of storage space

Experimentation also compared query efficiency of these two storage methods. On the same machine, aiming at the same query objective, query experimentation is carried out by two storage methods. Figure 5 indicates that when file is small, time required by these two storage methods is almost equal, however as the document size increase, query time required by GML file obviously increase and longer than G2NXDBMs prototype system need. Because when querying GML document, the whole XML file is parsed to a tree in main memory. Generally speaking, parsing time determine query computing time, so larger size of file longer time query needs. On the contrast, G2NXDBMs prototype system store GML document by fragments, and use mature query technologies such as XQuery in native XML database, so it has certain advantage on query aspect.

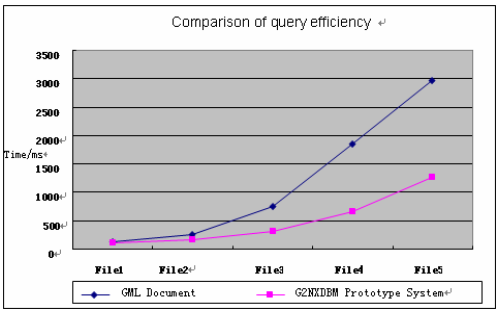

Figure 5. Comparison of query time

Experimentation result indicates that storage space of G2NXDBMs prototype system is larger than GML document. But G2NXDBMs prototype system has obvious advantage on aspects of query efficiency, support to XML technology and data management. Comparison of overall evaluation on these two storage methods is listed in Table 4.

Evaluation Index	Storage Method	
	GML File System	G2NXDBMs Prototype System
Query Speed	slow	quick
Storage space	small	larger
Support to XML technology	weak	strong
Data management function	weak	strong

Table 4. Comparison of two storage methods

4. CONCLUSION

Combined with characters of native XML database and GML spatial data, this paper proposed native XML database based GML storage model G2NXDBM, on the basis of this model, this paper designs and develops G2NXDBMs prototype system consisted of three parts: schema mapper, document storager, and spatial analyser. The experimentation use GML data describe city to testify rationality of prototype system, the conclusion indicates that this system can effectively store and query GML data.

Limited to content and time, the storage model G2NXDBM proposed by this paper does not refer to a series of operation, such as GML data insert, delete, and update, these problems are needed to be studied further and resolved. In addition, native XML database based GML native database should contain content on aspects of storage, index, retrieve, concurrency, security, integrality and so on. Thus, this paper would consider factors of storage, index, retrieve, concurrency, security, integrality in future, and gradually implement native GML database.

ACKNOWLEDGEMENTS

This work was sponsored by the National Natural Science Foundation of China (Research on the Storage and Index Mechanism of GML Spatial Data, No. 040401045) and 863 Project of National Science and Technology ministry of China (Research on GML-GIS and its implementation schema of application, No.2006AA12Z221)

REFERENCES

H. V. Jagadish ,2002. H. V. Jagadish, Shurug AL-Khalifa, et al.2002.TIMBER: A Native XML Database. Technical Report, University of Michigan

Harrie, 2006. Using Java Topology Suite For Real-Time Data Generalisation And Integration. http://www.ikg.uni-hannvoer.de/isprs/workshop/Johansson_Harrie.pdf

http://exist.sf.net ,2006. eXist: http://exist.sf.net

http://idke.ruc.edu.cn,2006. OrientX: http://idke.ruc.edu.cn/OrientX/

http://www.ipedo.com.cn,2006. IPedo:http://www.ipedo.com.cn

http://www.softwareag.com,2006. Tamino: http://www.softwareag.com/tamino/architecture.htm

http://www.vividsolutions.com,2006.http://www.vividsolutions.com/jts/jtshome.htm

http://www.x-hive.com, 2006. X-Hive/DB:http://www.x-hive.com/products/

http://xml.apache.org, 2006. Xindice: http://xml.apache.org/xindice/

J. McHugh, 1997. J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A Database Management System for Semistructured Data. *SIGMOD Record*, Vol.26（3）:54-66, September 1997.

Liugang,2005. Naitve XML database research and application, *Microsystems Development*, (8)

T. Fiebig, 2002. T. Fiebig et al. Anatomy of a native XML base management system. http://www.csd.uch.gr/~hy561/Papers/natix02.pdf

www.rpbourret.com,2005.XML Database Products: Native XMLDatabases.http://www.rpbourret.com/xml/ProdsNative.htm