American Journal of Applied Sciences 5 (10): 1358-1368, 2008
ISSN 1546-9239

Publishing Geospatial Data through Geospatial Web Service and XML Database System

Pouria Amirian and Ali A. Alesheikh
Faculty of Geomatics Engineering, K.N. Toosi University of Technology,
Vali-e-asr St., Mirdamad Cross, 19967-15433 Tehran, Iran

Abstract: Technically the spatial non-interoperability problem associated with current geospatial processing systems can be categorized as data and access non-interoperability. In GIS community, Open GIS Consortium (OGC) geospatial Web services have been introduced to overcome spatial non-interoperability problem associated with most geospatial processing systems. At the same time, in Information Technology (IT) world, the best solution for providing interoperability among heterogeneous systems is Web services technologies. Geospatial Web services and Web services technologies differ in the way that latter are composed of particular set of technologies and protocols but the former are comprised of defined set of interface implementation specifications which can be implemented with diverse technologies. This research describes and discusses that geospatial Web services which are developed using Web services technologies can provide access interoperability among various geospatial and non-geospatial processing systems. In addition to access interoperability, making use of open and platform independent data standards like Geography Markup Language (GML), data interoperability can be achieved. Meanwhile, proper management of geospatial data necessitates use of efficient and optimized data management systems. In this respect, the study also illustrates the practical evaluation of existing solution for storing and publishing geospatial data as GML. Based on the practical evaluation of this research, coupling native-XML database systems with Web services technologies proved to be an open, interoperable and efficient solution for developing geospatial Web services.

Key words: Geospatial web services, XML database, geography markup language, interoperability

INTRODUCTION

Majority of geospatial processing systems require some level of interoperability as a fundamental capability. Based on OGC Reference Model[1], spatial interoperability refers to capability to communicate, execute programs, or transfer geospatial data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units. Therefore, non-interoperability of geospatial processing systems prevents sharing of geospatial data and services among software applications. Spatial interoperability faces two main challenges; syntactic heterogeneity and semantic heterogeneity[2]. Syntactic heterogeneity which is the result of differences in storage formats and software incompatibility is a technical issue and can be addressed by technical means. Semantic heterogeneity arises as a result of incompatibility in meanings of data. Addressing syntactic heterogeneity is the main concern of this research.

Syntactic heterogeneity of geospatial information systems can be categorized in data and access heterogeneity. In data heterogeneity geospatial processing systems use various internal proprietary data formats. To share geospatial data, converters and/or transfer formats must be developed, which is a resource and time consuming task. In addition, there are so many different standards for geospatial data that converting various data formats can itself become a barrier to interoperability.

Access heterogeneity restricts inter-process communication among various geospatial processing systems, since different vendors' geospatial processing systems use proprietary software access methods with proprietary software interfaces. In other words, interface definition languages, communication protocols, communication ports and even object transfer mechanisms, varies in each software development platform. So the software platform which has been used to develop the geospatial processing system imposes the use of specific and proprietary

Corresponding Author: Pouria Amirian, Faculty of Geomatics Engineering, K.N. Toosi University of Technology, Vali-e-asr St., Mirdamad Cross, 19967-15433 Tehran, Iran. Pouria.Amirian@gmail.com

communication methods among various parts of the system. For this reason, different geospatial processing systems that have been developed by different software development platforms, cannot communicate and share services automatically and in an interoperable manner.

In GIS community, OGC has introduced specific kind of online services, to overcome spatial non-interoperability problem. These services which are called OGC geospatial Web services (or geospatial Web services for short) have been developed with the goal of sharing geospatial data and services among heterogeneous geospatial processing systems. Web Feature Service (WFS) and Web Map Service (WMS) are the most fundamental geospatial Web services which are introduced by OGC. At the same time, in Information Technology (IT) world, the best solution for providing interoperability among heterogeneous software systems in distributed and decentralized environments are Web services technologies[3].

Geospatial Web services and Web services differ in a way that Web services are composed of particular set of technologies and protocols but Geospatial Web services are comprised of defined set of interface implementation specifications which can be implemented with diverse technologies[4].

With respect to above description, it is suggested that making use of Web services technologies as enabling infrastructure for implementing geospatial Web services would significantly facilitate sharing geospatial data as well as access to processing services from multiple resources in and out of GIS community. In other words, geospatial Web services which are developed using Web services technologies can provide access interoperability among various geospatial and non-geospatial processing systems. Furthermore, using open and platform independent data standards like GML, data interoperability can also be achieved. Meanwhile, proper management of geospatial data necessitates use of efficient and optimized data management systems. In this context, considering the nature of GML as an XML-based standard, using XML database systems is suggested for facilitating and improving geospatial data management. This research describes development of a Geospatial Web service using Web Services Technologies and XML database systems to achieve spatial interoperability, while having a proper management on spatial data over the web. Since there are two technologies for XML data management, in the context of developed geospatial Web service, practical evaluation of two technologies illustrated as well. The study first explains Web services technologies, geospatial Web services and XML database systems. Afterwards design and development of a geospatial Web service is discussed. Finally practical test and its outcomes illustrated.

WEB SERVICES PLATFORM

Web services are self-contained, self-describing, modular applications that can be published, located and invoked across the Web and perform functions that can be anything from simple requests to complicated business processes[5]. Web Services are the basic components of distributed service-oriented systems. The World Wide Web Consortium (W3C) defines Web Services as a software system designed to support machine-to-machine interaction over the Internet[6,7,8].

Any Web service has an interface described in a machine-processable format. Other systems and services interact with the Web service in a manner described by its description using messages. Messages are conveyed typically using Hyper Text Transfer Protocol (HTTP) with an XML serialization, in conjunction with other Web-related standards, but any other communication protocol can be used as well[6].

Web services are based on open standards, so they provide interoperability in decentralized and distributed environments like Web. These new technologies can be developed by using any software platform, operating system, programming language and object model. The basic Web services architecture consists of specifications of Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description Discovery and Integration (UDDI). The mentioned core specifications support the interaction of a Web service requester with a Web service provider and the potential discovery of the Web service description.

SOAP is a lightweight, XML-based protocol for exchanging information in decentralized, distributed environments. SOAP is used for messaging among various components in a Web services platform. SOAP is platform independent and also it can be used with virtually any Network Transport protocols such as File Transfer Protocol (FTP) and HTTP.

WSDL is XML-based specification for describing capabilities of a service in a standard and extensible manner. Technically WSDL defines the software interface of Web service in platform independent approach.

UDDI is a set of specifications and Application Programming Interfaces (APIs) for registering, finding and discovering services.

From middleware point of view, Web service technologies are one of distributed component technologies. But the goal of Web services goes beyond

those of classical distributed component technologies such as Java RMI, .NET Remoting and CORBA: Web services aim at standardized support for higher level interactions such as service and process flow orchestration, enterprise application integration and provision of middleware of middleware[9]. Instead of building applications that result in collections of objects or components that are firmly integrated and understood just in development time, the service approach of Web services platform is much more dynamic and is able to find, retrieve and invoke a distributed service dynamically[4]. Another key difference is that with Web Services the industry is solving problems using technologies and specifications that are being developed in an open way, via partnerships and consortia such as the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS) and using standards and technologies that are the basis of the Internet.

From a technical standpoint, each Web service has three main parts: Service description, Executable agent and the mapping layer between the two (Fig. 1).

The machine-readable service description (that is a WSDL document) contains network address for the service, the operation it supports and other necessary information for consuming the service. The executable agent is responsible for implementing the functionality of the service. The description is separated from the executable agent using a mapping layer. The mapping layer is implemented using proxies and skeleton in service requester and service provider respectively[10]. This layer is responsible for accepting the message, transforming the XML data to and from the native format of executable agent and finally dispatching the data to the executable agent. On account of separation between executable agent and description of service or separation between semantic and functionality of services in the Web services world, each service can be developed by using any software development platform, operating system, programming language and object model.

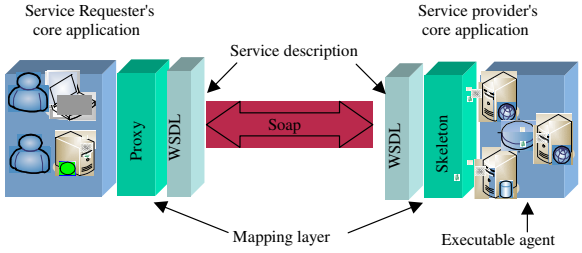

Fig. 1: Major parts of a web service

GEOSPATIAL WEB SERVICES

Nowadays, geospatial Web services have been considered as the promising technology to overcome the non-interoperability problem associated with current geospatial processing systems. They are particular kind of online services which deal with geospatial information and can provide access to geospatial information stored in a database, perform simple and complex geospatial analysis and return messages that contain geospatial information[11].

In this context, OGC has defined a comprehensive framework of geospatial Web services which is known as OGC Web services framework (OWS). OWS allows distributed spatial processing systems to interact with the HTTP technique and provides a framework of interoperability for the many web-based services, such as accessing spatial data services, spatial processing services and data locating services[12]. OWS framework consists of interface implementation specification and encodings which are openly available to be implemented by developers. The interface implementation specifications are software technology neutral details about various operations of each geospatial Web service. The encodings provide the standard glue among different parts of geospatial Web services. Each service of this framework can be implemented using various software technologies and systems. The most fundamental services and encodings of the OGC Web service framework are Web Map Service (WMS), Web Feature Service (WFS), Geography Markup Language (GML) and Common Query Language (CQL).

In short, WMS is an OGC Web service that provides maps on request. A map is a graphical visualization of geospatial data. WFS is the main geospatial Web service for publishing and requesting vector geospatial data in GML format. When a client sends a request to an OGC WFS, the service sends a response message that provides geospatial feature data in GML. In this case, requests for geospatial data contain CQL expressions. Using CQL, spatial and non-spatial query expressions can be created to be sent to WFS. WMS and WFS and other geospatial Web services supply standard access to geospatial data thus provide access interoperability in GIS community.

According to GML specification[13], GML is an XML grammar written in XML Schema for modeling, transporting and storing geospatial information. GML playing a major role in OGC Web service framework. Next section briefly introduces GML.

Geospatial Web service differs from the Web services. The most important distinction between these

two kinds of services is the fact that Web services are particular set of technologies and protocols but geospatial Web services are composed of defined set of interface implementation specifications which can be implemented with diverse technologies. Following items explain the mentioned difference in detail[4]:

- In the OGC Web service framework HTTP is defined as the sole distributed computing environment. In contrast, Web services can be implemented virtually with any standard protocols such as HTTP, FTP and TCP to name a few
- OGC Web services do not necessarily use the usual Web services core standards, including SOAP and WSDL. In other words, in Web services platform, the main messaging protocol is SOAP and this protocol can be considered as the main cause of achieving interoperability among various applications which are running on heterogeneous platforms. In OGC Web service framework SOAP is not the main messaging protocol. In addition in most geospatial Web services, creation and publication of WSDL document has not defined yet
- OGC Web services have particular interface for binding that might leads to interface coupling problem. In accordance with OGC Web service framework specifications, each geospatial Web services have to publish its own capabilities through a so called capabilities document. This document (which is an XML document) provides human and machine-readable information about the geospatial data and operation supported by a specific instance of a geospatial Web service. But this document is not comprehensive enough to play a same role as WSDL document. In other words, capabilities document cannot offer enough information to enable developers and thus software applications to consume a geospatial Web service programmatically and automatically, while according to Newcomer and Lomow[10], ideally the service requester should be able to use a service exclusively based on the published service contract
- In OGC Web service framework, CQL is used to specify which geospatial data have to be sent back to requester. This language can cause interoperability problems when considering the various scenarios in which there is a need for geospatial Web services to communicate with other Web services outside geospatial community. This language has unconventional structure when compared with other standard query languages such as Structure Query Language (SQL) and Xml Query Language (XQuery). The structure of CQL has the potential of causing interface binding problem which is the main barrier in front of making truly loosely coupled services

In summary, OGC's Web services and Web services are compatible with each other, but they are not technically implemented in the same way.

As mentioned before, geospatial Web services can be implemented using existing software development technologies. It is suggested that using Web services technologies as enabling infrastructure for implementing geospatial Web services can significantly facilitate sharing geospatial data as well as access to processing services from multiple resources in and out of GIS community. In other words, geospatial Web services which are developed using Web services technologies can provide access interoperability among various geospatial and non-geospatial processing systems.

GML

GML is rapidly emerging as a world standard for the encoding, transport and storage of all forms of geospatial information. GML provides data interoperability among heterogeneous geospatial processing systems. GML is an XML-based markup language that is used to encode information about real world objects. In GML these real world objects are called features and they have geometry and non-geometry properties.

GML has three main roles with respect to geospatial information. First as an encoding for the transport of geospatial information from one system to another; second as a modeling language for describing geospatial information types; and third as storage format for geospatial information[14].

GML is well suited for encoding the geospatial information sent to and from geospatial Web services. GML is used in both the request and response messages of the WFS, which is a standard service for accessing geospatial feature data.

As a modeling language, GML provides basic types, construct, units of measure, reference systems and so on to model all aspects of real world objects and relationships among them.

As storage format GML is a plain textual file format. Since GML is based on XML, the same technology for managing XML data can be used to manage geospatial data stored in GML. Management of geospatial data has a considerable impact on the way the geospatial Web services (and in particular WFS)

retrieve and publish geospatial data in GML format. In general there are two technologies for management of XML data. In this research both XML management technologies have been evaluated in the context of a geospatial Web service. This evaluation was intended to indicate the strengths and weaknesses of each technology in retrieving and publishing geospatial data.

XML DATABASES

XML, as a rich set of technologies, is playing an important and increasing role in share and exchange of data over the Web. The more XML has been used in share and exchange of data, the more XML data management issues have to be considered. So, database researchers have actively participated in developing technologies centered on XML data management, in particular data models and query languages for XML. As a result of these researches, many XML data management systems have been implemented. In general, XML data management systems can be categorized as XML-enabled databases and native-XML databases[15].

Typically, an XML-enabled database is a relational database which provides storage of hierarchical XML documents in relational model and provides proprietary methods for relational to XML data mapping (or conversion) for retrieving stored data as XML. The mentioned proprietary methods vary in each software package from extension to standard SQL language to implementation of a full featured XML query language.

On the other hand, a native-XML database has an XML document as its fundamental unit of logical storage, just as a relational database has a row in a relation as its fundamental unit of logical storage. A native-XML database defines a logical model for its fundamental unit of storage and stores and retrieves XML documents according to that model[15]. The advantage of this native approach is that XML data can be stored and retrieved in their original formats and no additional mappings or translations are needed. Furthermore, most native-XML databases have the ability to perform sophisticated full-text searches including full thesaurus support, word stubbing (to match all forms of a word: run, ran, running) and proximity searches[16].

Since there are two technologies for XML data management, comparative performance analysis have to be performed to indicate which technology is more appropriate to manage geospatial data stored as GML. In order to compare native-XML with XML-enabled databases, two well known commercial database systems were selected. Microsoft SQL Server 2000 as a XML-enabled and Software AG's Tamino 4.4 as native-XML database employed to store geospatial data as GML.

Microsoft SQL Server 2000 takes advantage of an embedded engine capable of returning data as XML. This feature is built as an extension to the standard SQL SELECT command and data is rendered as XML before being sent back to the client[17]. At the other hand, Tamino is a software system for storing, managing, publishing and exchanging XML documents in their native format. At the heart of Tamino is a powerful XML engine providing all functionality necessary to dynamically process, generate and exchange XML documents[18].

Although various benchmarks had been developed for studying the efficiency of XML databases[19,20,21,22,23,24] most of them concentrate on defining set of queries and specifications for evaluation of XML data management technologies. Other related works consist of evaluation of using various methods for extracting XML data from relational databases[16,25,26] and evaluation of XML query languages[27,28,29]. In this context, no work has been done on the type of data which should be stored in each kind of XML management technology and what kind of queries perform best on each one. In addition geospatial data which are encoded as GML are huge in volume, rich in data types and complex in semantics. So a dedicated evaluation should be performed to indicate which kind of XML management technologies should be employed to store GML data. In this research performance evaluation has been made between the mentioned two database systems in the context of a geospatial Web service.

EXPERIMENTAL EVALUATION

Implementing geospatial web service using XML databases: In order to evaluate native-XML databases and XML-enabled databases for storing and retrieving GML data, an OGC WFS service was designed and developed. In the context of the WFS, evaluations were made on GML data of various sizes which was stored in native-XML and XML-enabled database systems. In this evaluation, set of standard queries for feature data retrieval was run on both systems and running time of queries was used as the measure of performance. Besides, size of GML data and ability to load huge amount of GML data into databases measured as well. This section describes the sample data, architecture and result of the evaluation of the implemented system.

Sample GML data: As sample data, three different sizes GML documents were created. Three layers of

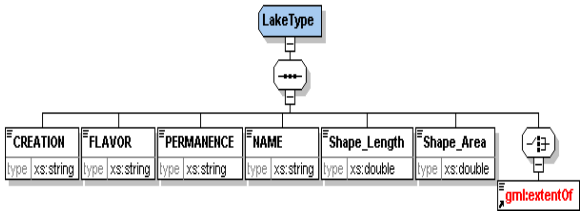

Fig. 2: GML application schema of Lakes Layer. Creation, Flavor, Permanence, Name, Shape_Length and Shape_Area are non-spatial properties of each lake feature. Geometry of each lake feature is described using gml:extentOf element

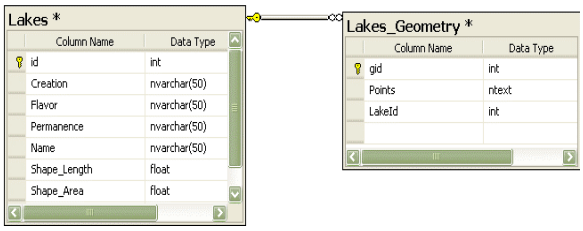

Fig. 3: Relational schema of Lakes Layer. Each record in Geometry_Lake table has a LakeId field that points to a corresponding lake feature in Lakes table. Points field store space delimited coordinates of composing points of each lake feature

1:25000 data of Lakes, Provinces and Cities of Iran were selected and converted to GML 3.1 documents and corresponding GML application schemas were produced. Afterwards corresponding relational schemas of data tables created. Figure 2 show the GML application schema and Fig. 3, shows relational schema of Lakes layer. As illustrated in relational schema of Lake layer, each record in Geometry_Lake table has a LakeId field that points to a corresponding lake feature in Lakes table. Also it was possible to create a single table for each layer and store geometry of each feature in the same table, using two or more separate tables is more general solution. Since there might be some features which have complex geometries (interior, exterior boundaries) the single table solution is only applicable for simple features but it becomes infeasible if complex features are included. While the mentioned layers just contain simple features, the solution used in this research is extensible to include complex features as well.

These layers (Lakes, Provinces and Cities) contain 31, 30 and 936 features respectively. For realistic testing of performance some randomly generated features have been added to GML documents to produce large GML datasets. As a result Provinces, Lakes and Cities layers, contain 1000, 10000 and 100000 features respectively.

Design and implementation of geospatial web service: On account of integrating XML database systems and Web services technologies for developing a Geospatial web service and determining the best technology for managing geospatial data as GML, an OGC WFS was designed and developed. According to WFS Implementation Specification[30], there are three operations for basic WFS:

GetCapabilities: The purpose of this operation is to obtain service metadata, which is a machine readable (and also human-readable) description of the required technical information for consuming WFS. The most important part of the service metadata indicates which feature types the WFS can provide and what operations are supported on each feature type.

DescribeFeatureType: WFS describes the structure or schema of any feature type it can provide using DescribeFeatureType operation. This structure will be used for retrieving geospatial data.

GetFeature: This operation is used for retrieving geospatial data. Making use of CQL, spatial and non-spatial query expressions can be introduced to WFS to retrieve the appropriate GML data. This operation was used to compare performance of native-XML and XML-enabled Database.

These operations provide the software interface of the WFS system. In other words, internal details of the functional units and software components as well as communications are transparent to consumer applications; the consumer application just can communicate with the WFS system through the operations and defined set of parameters which are specified in WFS implementation specification. Software components, communication among them and physical location of each component are specified in logical and physical architecture of the WFS system.

Physical architecture is quite different from a logical architecture. The physical architecture is about the number of machines or network hops involved in running the application. Rather, a logical architecture is all about separating different types of functionality in software components[31].

Traditionally logical architecture of software applications consists of three tiers; presentation and user interface tier, business logic tier and data

management tier. With the advent of new technologies and software design patterns the traditional logical architecture is rarely efficient for the modern software applications. Today, the business logic tier is often physically splits among a client, Web server and application server. In addition, with new software design patterns (such as façade, flyweight, adapter and composite) the business logic breaks up into multiple parts and components.

In this research the WFS designed in four logical tiers: presentation and user interface tier, business logic tier, data access tier and data management tier.

As the name implies, the presentation and user interface tier provides the end user an appropriate and friendly user interface which hides details of local and remote computational tasks from user. This tier is responsible for gathering the user inputs, validating the user inputs, composing CQL statements based on the user inputs to make requests, sending requests to WFS server and displaying the returned geospatial data.

The business logic tier includes all business rules for the WFS system. For the implemented WFS theses rules consist of translating requests to DBMS specific query language statements and dispatching them to the next tier.

Data access tier interacts with the data management tier to retrieve, update and remove information. The data access tier doesn't actually manage or store the data; it merely provides an interface between the business logic and the database. Logically, defining data access as a separate tier enforces a separation between the business logic and how application interacts with a database (or any other data source). This separation provides the flexibility to choose later whether to run the data access code on the same machine as the business logic, or on a separate machine. It also makes it much easier to change data sources technologies without affecting the application. In addition by isolating the data access code into a specific layer, the impact of changes in data access technologies is limited to a smaller part of the application. This is important because in this research two distinct database products and access technologies were utilized to evaluate which solution provide best performance for storing and retrieving GML data.

The forth tier handles the physical retrieval, update and deletion of data. This is different from the data access tier, which requests the retrieval, update and deletion of data. These operations are implemented within the context of a full fledged database management system.

The first three tiers of the mentioned logical architecture have been developed using Microsoft. NET 2.0 framework. In order to implement the client side application (user interface and presentation tier) a windows-based application was developed. Web services infrastructure was utilized in all interactions between client side application and WFS server. In other words, WSDL was used to create proxy and skeleton in client side application and business objects of WFS server respectively. SOAP was used to transport every interaction (request and response) between the proxy and skeleton.

In client side application CQL statements which are created by client side application (using various logical and comparison operators) specify which feature types and attributes are required. The created CQL statements then are sent to the WFS server and the requested geospatial data is sent back to client side application.

In the developed system, objects and components in business logic and data access tiers work together to prepare an appropriate response message. More accurately, user supplied parameters are parsed by business objects to determine which methods have to be executed. In the case of GetFeature operation, user supplied CQL statements are translated to DBMS specific query language (SQL in case of SQL Server 2000 or XQuery in case of Tamino). Then SQL or XQuery statements are delivered to objects and components in data access tier to be sent to DBMS.

In the last tier of the architecture, geospatial data were stored as GML 3.1 in the back-end XML databases. For retrieving geospatial data, SQL or XQuery statements which were sent by data access components are executed and result are sent back to the data access component. Data access components dispatch retrieved geospatial data to business logic components. Afterwards geospatial data are prepared to be valid against WFS specifications. Finally, prepared GML data are sent back to the client using Web services infrastructure (using SOAP).

In implemented physical architecture of WFS, two identical computer machines were employed to evaluate XML databases, with Intel Pentium IV CPUs clocks at 3.2 GHz, 1024 MB of main memory and 120 GB of hard disk. The operating system was Windows XP professional and Microsoft Internet Information Services 6.0 (IIS 6.0) was utilized as Web server application. Data access, business logic components and XML databases (native-XML and XML-enabled databases) deployed and installed on both machines separately.

The cost metrics used in this research is processing or running time of query (GetFeature Operation) to measure the overall response time. To choose an

appropriate set of queries for this research, most commonly types of queries used in retrieving GML data through WFS were considered.

It should be noted that, in an environment where servers and clients communicate through network infrastructure, processing time would be a measure of not solely of the database, but of all the software and hardware comprising the system including the operating system and the network. This, in turn, makes it hard to isolate the query processing time of database from other parameters involved in the whole process. Therefore in this research all tiers of application were physically deployed on single machines to get optimal performance. Performance is the speed at which an application responds to a user. To get optimal performance which is the fastest possible response time for a given user, the ideal solution is to put all tiers of an application on one computer machine. This means no network hops, no network latency and no contention with other users.

In addition to processing time, disk space required by each XML database to store same GML data and ability to load huge amount of GML data to database were evaluated. Processing times are measured using performance monitor utility classes which was developed using core classes in Microsoft .NET framework. Next sections present outcomes of the evaluation.

DISK SPACE AND INDEX SIZE EVALUATION

As shown in Fig. 4, the native-XML database needs more disk space to store both GML data and indexes than the XML-enabled database. The result is more serious as the number of features increases. The reason for this large size is that the native-XML database must store verbose GML structure which contains both data and descriptive tags. In other words, GML has the same shortcoming in disk space efficiency as its predecessor (XML). The tradeoff of XML versus other exchange formats is that the verbosity introduced by the descriptive tags of data elements improves readability and semantic exchange at the sacrifice of increased file size. Between 20 to 60% of an XML document consists of tag names[32].

In general, indexing increases the insertion and update time and decreases the response time in query processing. Tamino allows two types of indexing on text fields; Text indexing and standard indexing. Both can be built on the same field/element. The text indexing is utilized in pattern matching queries (the Like operator in SQL). The standard indexing is used for textual and numeric fields. Indices are created on all

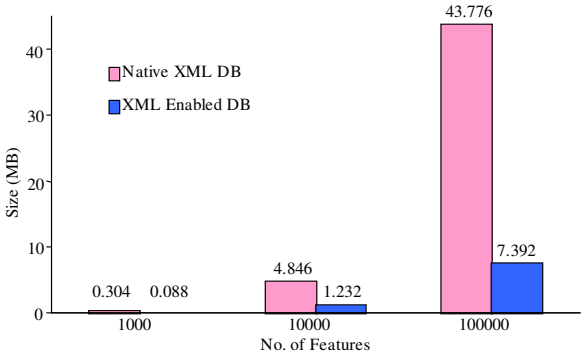

Fig. 4: Database size (both GML data and Indexes)

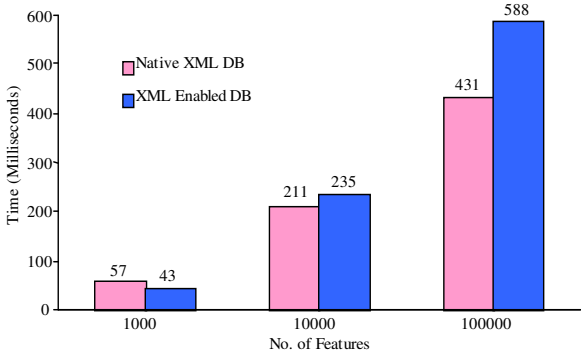

Fig. 5: Retrieval of single feature using gml:id attribute

numeric and textual elements in Tamino, since those are the only fields used with comparison operators. The only field that has a text index is Name field of all GML documents (Name of Lakes, Provinces and Cities). The primary key fields have default indexes in SQL Server 2000 (unique indexes). In addition to primary keys, foreign keys and other fields were indexed as well.

RETRIEVAL OF SINGLE FEATURE USING GML

ID evaluation: The GetFeature operation for a single feature using gml:id attribute measures the time to search for a feature using an index key (id field is the primary key of Provinces, Lakes and Cities Table). In this case for 1000 features, the results for both products are very similar but for 10000 records onward, the native-XML database outperforms the XML-enabled database (Fig. 5). The results shows that the native storage strategy and indexing approach used in native-XML database are more efficient solution for retrieving geospatial data. In this test the following CQL statement were used.

```
<GetFeature service = "WFS" version = "1.1.0"
outputFormat = "text/xml; subtype = gml/3.1.1">
   <wfs:Query typeName = "PAM:Lake">
     <ogc:Filter>
       <ogc:GmlObjectId gml:id = "L5000" />
     </ogc:Filter>
   </wfs:Query>
</GetFeature>
```

Code 1: CQL statements for retrieving Lake feature which has the L5000 attribute as gml:id

Pattern matching query for retrieving features evaluation: For pattern matching the following CQL statement was evaluated:

```
<GetFeature service = "WFS" version = "1.1.0"
outputFormat = "text/xml; subtype = gml/3.1.1">
   <wfs:Query typeName = "PAM:Lake">
     <ogc:Filter>
        <PropertyIsLike wildcard = "*" singleChar =
        "#" escapeChar = "!">
            <PropertyName>Name</PropertyName>
            <Literal>*ab*</Literal>
        </PropertyIsLike>
     </ogc:Filter>
   </wfs:Query>
</GetFeature>
```

Code 2: CQL statements for retrieving lake features whose names contain the substring "ab"

The above CQL expression returns all Lake features whose names contain the substring "ab". As shown in Fig. 6, native-XML database outperforms XML-enabled Database in all cases. It is concluded that using text index which was created for Name fields, provide this performance improvement.

Retrieval of whole features of a layer evaluation: In order to retrieve whole features of all layers (111000 features of Cities, Lakes and Provinces layers) the following CQL expression was utilized.

```
<GetFeature service = "WFS" version = "1.1.0"
outputFormat = "text/xml; subtype = gml/3.1.1">
   <wfs:Query typeName = "PAM:Cities">
   <wfs:Query typeName = "PAM:Lakes" />
   <wfs:Query typeName = "PAM:Provinces" />
</GetFeature>
```

Code 3: CQL statements for retrieving all features of Provinces, Lakes and Cities Layers

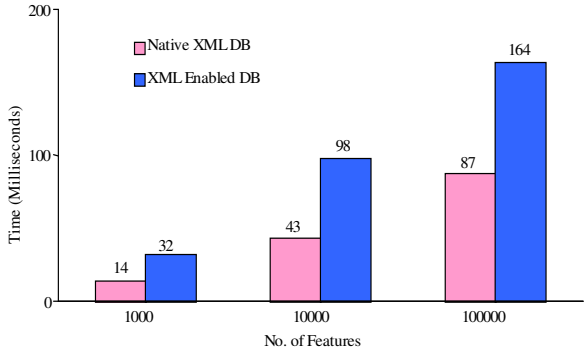

Fig. 6: Retrieval of features using pattern matching

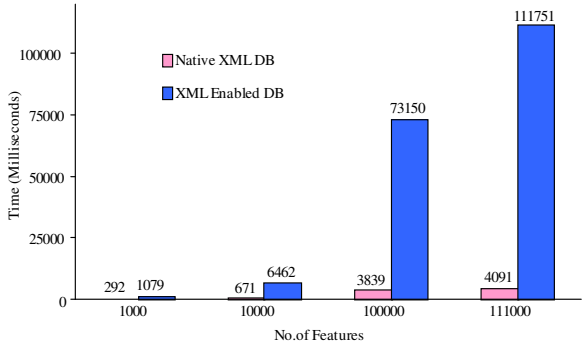

Fig. 7: Retrieval of whole features

For retrieving whole features of layers the native-XML database has better performance than the XML-enabled database as data size becomes larger (Fig. 7). Since XML-enabled database uses two separate tables for each layer, retrieving whole features of a layer has the overhead of joining two large tables. Also this may be due to the native-XML database's storage strategy which does not need to reshape data to provide data as GML format.

GML BULK LOADING EVALUATION

XML bulk loading technique is optimized to load huge amount of XML data into database. XML bulk loading provides necessary tools and utilities for reading, caching and inserting massive amount of XML data into database.

In SQL Server 2000 XML bulk loading component reads the XML data and identified the database tables and fields involved. It then executed SQL statements against SQL Server 2000 to insert whole document into pertinent tables and fields. At the other hand, Tamino Mass Loader Facility has been developed for efficient loading of large size XML documents.

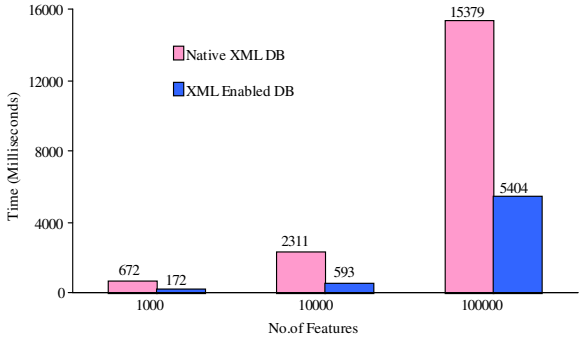

Fig. 8: Bulk loading of GML features into databasess

As the Fig. 8, indicates, bulk loading component of SQL Server 2000 is as an efficient tool for data loading. It provides higher performance when loading large amount of data into database than Tamino's Mass Loader Facility.

RESULT OF THE EVALUATIONS

After analyzing the practical results, it is concluded that the native-XML database has better performance than the XML-enabled database for handling GML documents with larger data sizes. In general XML-enabled database cannot handle large-sized GML documents as efficiently due to conversion overhead. In contrast, the native-XML database engine directly accesses GML data without conversion. In other words, native-XML databases enables direct operation on GML documents, features in document and attributes of each features as opposed to complicated joins of relational tables in XML-enabled database. This saves time on programming, execution and retrieval, especially for complex features types.

Although the native-XML database provides high performance in handling GML documents, data and index size consumed by the native-XML database is much larger than in the XML-enabled database. With this in mind, using native-XML databases for storing geospatial data (as GML), provides an efficient solution for storing and accessing high volume geospatial data in multi-user enterprise environments. In addition, as more and more data is stored and exchanged using GML format, by using XML technologies which are easy to integrate with native-XML databases, more spatial capabilities can be added to native-XML databases.

CONCLUSION

In this research design, development and practical evaluation of a geospatial Web service (basic WFS) using Web Services platform and XML database technologies was described.

Developing geospatial Web service using Web services technologies provide interoperability among geospatial and non-geospatial processing systems. Since Web services technologies are foundation of direct and open application-to-application communication, functionality of the implemented WFS can be simply added to any geospatial or non-geospatial processing systems which are running on heterogeneous platforms. Furthermore, logical designing of WFS using four tier logical architecture, end in a software system which is flexible to be implemented in various physical architectures. So the WFS can be configured into an appropriate physical architecture that will depend on our performance, scalability, fault-tolerance and security requirements. Besides, by isolating the data access code into a specific tier (Data Access tier), the impact of changes in data access technologies was limited to a smaller part of the application. This is important because in this research two distinct database products and access technologies were utilized.

Since GML is based on XML, XML databases can be used to manage geospatial data. Based on practical tests of this research using native-XML databases provides an efficient solution for storing and accessing high volume geospatial data in multi-user enterprise environments. Considering outcomes of this research, coupling native-XML database systems with Web services technologies proved to be an open, interoperable and efficient solution for developing geospatial Web services.

REFERENCES

1. OGC., 2003. The OpenGIS Reference Model. http://portal.opengeospatial.org/files.
2. Worboys, M. and M. Duckham, 2004. GIS a Computing Perspective, Florida, USA, CRC Press.
3. Volter, M., M. Kricher and U. Zdun, 2005. Remoting Patterns: Fundamental of Enterprise, Internet and Realtime Distributed Object Middleware, New Jersey. John Wiley & Sons, Inc, USA.
4. Amirian, P. and A. Alesheikh, 2008. J. Applied Sci., 8 (5): 730-742.
5. Nance, K.L. and B. Hay, 2005. Automatic transformations between geoscience standards using XML. Comput. Geosci., 31 (9): 1165-1174.
6. Stal, M., 2002. Web services: Beyond component-based computing. J. Commun. ACM., 45 (10): 71-76.

7. Booth, D., H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris and D. Orchard, 2004. Web Services Architecture, W3C Working Group. http://www.w3.org/TR/ws-arch.
8. W3C., 2006. The World Wide Web Consortium. Web Services Activity Statement. http://www.w3.org/2002/ws/Activity.
9. Vinoski, S., 2003. Integration With Web Services, IEEE J. Internet Comput., 7 (6): 75-77.
10. Newcomer, E. and G. Lomow, 2005. Understanding SOA with Web Services, Maryland, Addison Wesley, Inc, USA.
11. Lake, R., D. Burggraf, M. Trinic and L. Rae, 2004. Geography Markup Language, Chichester, England, John Wiley and Sons.
12. Zhang, J., J. Gong, H. Lin, G. Wang, J. Huang, J. Zhu, B. Xu and J. Teng, 2007. Design and development of distributed virtual geographic environment system based on web services. Inform. Sci., 177 (19): 3968-3980.
13. Open GIS Consortium, 2005. Geography Markup Language Specification 3.1 http://portal. opengeospatial .org/files/?artifact_id=4700.
14. Lake, R., 2005. The application of geography markup language (GML) to the geological sciences. Comput. Geosci., 31 (9): 1081-1094.
15. Lu, E.J., B.C. Wu and P.Y. Chuang, 2006. An empirical study of XML data management in business information systems. J. Syst. Software, 79 (7): 984-1000.
16. Atay, M., A. Chebotko, D. Liu, S. Lu and F. Fotouhi, 2007. Efficient schema-based XML-to-Relational data mapping. Inform. Syst., 32 (3): 458-476.
17. Neilsen, P., 2003. Microsoft SQL Server 2000 Bible, Indianapolis, Indiana, John Wiley and Sons, USA.
18. Software AG's Tamino XML Server. http:// documentation.softwareag.com/crossvision/ins441/ overview.htm.
19. Runapongsa, K., J.M. Patel, H.V. Jagadish and S. Al-Khalifa, 2002. The Michigan benchmark. http://www.eecs.umich.edu/db/mbench.
20. Yao, B.B., M.T. Ozsu and J. Keenleyside, 2002. XBench-A family of benchmarks for XML DBMSs. In: Proceedings of Efficiency and Effectiveness of XML Tools (EEXTT) 2002, pp: 974-985.
21. Bohme, T. and E. Rahm, 2001. XMach-1: A benchmark for XML data management. In: Proceedings of German Database Conference BTW2001, pp: 264-273.
22. Schmidt, A., F. Waas, M.J. Carey, I. Manolescu and R. Busse, 2002. XMark: A benchmark for XML data management. In: Proceedings of the 28th Very Large Databases (VLDB) Conference, Hong Kong, China, pp: 974-985.
23. Bressan, S., M.L. Lee, Y.G. Li, Z. Lacroix and U. Nambiar, 2002. The XOO7 benchMark. In: Proceedings of the First VLDB Workshop on Efficiency and Effectiveness of XML Tools and Techniques (EEXTT), Hong Kong, China.
24. Nambiar, U., Z. LaDDDDcroix, S. Bressan, M.L. Lee and Y.G. Li, 2002. Efficient XML data management: An analysis. In: Proceedings of the 3rd International Conference on Electronic Commerce and Web Technologies (ECWeb), Aix en Provence, France, 2002.
25. Liu, J. and M. Vincent, 2004. Querying relational databases through XSLT. Data Knowledge Eng., 48 (1): 103-128.
26. Fong, J., H.K. Wong and Z. Cheng, 2003. Converting relational database into XML documents with DOM. Inform. Software Technol., 45 (6): 335-355.
27. Chung, T.S. and H.J. Kim, 2003. Techniques for the evaluation of XML queries: A survey. Data Knowledge Eng., 46 (2): 225-246.
28. Haustein, M. and T. Harder, 2007. An efficient infrastructure for native transactional XML processing. Data Knowledge Eng., 61 (3): 500-523.
29. Jea, K.F. and S.Y. Chen, 2006. A high concurrency XPath-based locking protocol for XML databases. Inform. Software Technol., 48 (8): 708-716.
30. Opern GIS Consortium, 2005. Open GIS Web Feature Service implementation specification 1.1. <https://portal.opengeospatial.org/files/?artifact_id =8339>
31. Lhotka, R., 2006. Expert C# 2005 Business Objects. 2nd Edn. California, USA, APress Publishing.
32. Lawrence, R., 2004. The space efficiency of XML. Inform. Software Technol., 46 (11): 753-759.