Independent Study CS5990.010

Second Report

A GML-Based Open Architecture for building Geographical Information Searching Engine Over Internet: Design and Preliminary Implementation

Submitted to:

Dr. Le Gruenwald

Submitted by:

Jianting Zhang

Mar 20th, 2001
Content

Introduction ................................................................................................................................. 3
Related Work ............................................................................................................................... 6
System Architecture .................................................................................................................. 10
Implementation .......................................................................................................................... 13
  Software agents for collecting geographical Information over Internet ......................... 13
  GML Wrapper for Relational Database (SQL Server for example) ................................. 16
  GML wrapper for Object-Relational Database (Oracle for Example) ............................ 19
  Building a mediator that support spatial query ................................................................. 21
  Client Query and visualization ............................................................................................. 24
Experiment and Results ........................................................................................................... 25
Discussion and Future Directions ............................................................................................. 29
References ................................................................................................................................. 31
Appendix: Source Code List .................................................................................................... 34
A GML-Based Open Architecture for building Geographical Information Searching Engine Over Internet: Design and Preliminary Implementation

Introduction

Today’s Internet contains a great deal of information that can be geo-referenced. The National Academy of Sciences estimates that 80 percents of the information on the Internet have a spatial component ([HREF 1]). These include coordination information, such as latitude/longitude and their various kinds of projections; mailing address which can be geocoded into coordination; and relative distance/direction information. These information play crucial roles in people’s everyday life as well as governmental administrations.

Internet mapping of geographical data, such as MapQuest.com and Excite.com has provided geographical information access through Internet and are now gaining more and more popularity. Major GIS vendors, such as ESRI, MapInfo, Intergraph, AutoDesk, etc., provide software components to publish geographical data online. However, data as well as software is proprietary and they are completely controlled by vendors. The generated maps usually either in image forms (GIF/JPEG) or embedded objects in browsers and it is very hard for users to make further use of the geographical information except visualization. This is a serious limitation for geographical information application since geographical information can only be meaningful when integrated with domain information and knowledge.
With the widely applications of personal portable communication and computation devices, such as cell phone, GPS, PDA, Palm, etc., geographical information and application are undergoing significant changes. The volume of personal geographical information, such as location and moving direction information, produced by GPS and cell phone is much more than traditional geographical information (such as street map, cadastral information which are generally for government administration purpose). It is highly unlike that these personal geographical information can be collected and managed by a centralized system. Instead, they are often embedded in various forms of documents, such as HTML pages in an autonomous and distributed fashion. We believe that a geographical searching engine over Internet, which is an extension of current text-based information searching engine such as Google and InfoSeek, is highly desirable. There are tremendous business opportunities by bridging personal needs and services that can be provided.

Currently, the idea of Location-based Services (LBS) has drawn much attention from wireless communication to GIS applications. The key idea of LBS is that the a portable device send its location information to a gateway, the gateway search through its database to find most relevant information near the location and send it back to client for further process. The problem with this solution is that the gateway must maintain a centralized geographical information database to support query. For example, OpenWave system, a product from Phone.com, requires every website to report its geographical location to its gateway in order to be able to be accessed by its mobile devices. This solution requires a well-established geographical information infrastructure while anything not registered in the central database will not be accessible to end users.
A more formal and open solution which specifically focus on geographical information infrastructure is proposed by SRI International ([HREF 1]). They propose a new Top-Level Domain Name (TLD) named “.geo”. According to their suggestions, the earth is divided to 1 by 1 degree cells and each cell has one or more GeoRegistries. All URLs register their geographical locations with these GeoRegistries and the GeoRegistries will provide services to any geographical information requests. However, this proposal is rejected by ICANN recently. From practice perspective, one major shortcomings of the “.geo” TLD proposal is that it doesn’t address anything about how to deal with text-based geographical information such as mailing address which is more relevant to and widely used in our everyday life. Again, anything not registered in the central database will not be accessible to end-user.

As XML becoming more and more popular, a solution is to use XML tags to mark geographical information and deliver it over Internet, such as OGC’s Geographical Markup Language (GML)([HREF 2]). However, just like geographical information infrastructure can not be built in a short time, it is highly unlike that all HTML pages can be changed to XML in a short time.

Our solution is within the scope of current Internet application practice. The key idea is that individual Internet users can put whatever geographical information they want in HTML pages and we’ll build software agents to roam over the Internet to retrieve geographical information and put them in databases. Wrappers for these heterogeneous databases are built and GML is used as the communication protocol. A mediator is also built to integrate geographical information from the distributed databases of regional
servers. The mediator, which severs as a geographical information search engine, also
supports client query and analysis on geographical information.

The reset of this paper is arranged as follows. Section 2 overview of related work.
Section 3 presents the architecture of the proposed solution, section 4 describes the
implementation details, Section 5 are some primary results and finally Section 6 presents
some discussion and future work directions.

**Related Work**

The geographical perspective of information on the Web doesn’t t receive much
attention in database research based on our investigation. Only Two Papers
(Buyukkokten, 1999; Ding, 2000) are available related to web-based geographical
location information in DBLP literature in recent three years. While the first paper
(Buyukkokten, 1999) is more intuitive and focusing on applications such as site mapper,
area mapper and zip-code mapper, the second one (Ding, 2000) is more formal and
presents algorithms to estimate geographical scope of web resources. However, both of
their work is focused on characterizing web resources from geographical perspective.
They didn’t show how individual user’s can make use of web-based geographical
information. On the contrary, our primary goal is to build searching engine for web-based
geographical information.

There are several topics involved in our work. The most relevant ones are:
Internet Mapping, Software Agents, HTML and text information retrieval, XML and its
extension GML, heterogeneous database wrapping and integration, spatial database and
its support for geographical query and analysis, visualization of geographical
information, etc. In this section we’ll give a brief overview.
An overview of Internet Mapping is presented (Hardie, 1998). For more information on Internet mapping please refer to our previous review (Zhang, 2000). The academic project best known in this field is Alexandria Digital Library Project ([HREF 3]) funded by NSF digital library projects. In this project, collections of geographically referenced materials and services for accessing those collections are developed. A Gazetteer Server with a GUI is also built to map a place name on to an image. For example, when user type in “Washington”, the server returns all the entries related to Washington and map them on an image and then user can Zoom In/Out to a specific geographic region. However, all these place information are static and do not associate with any web pages.

The emerging Location-based Services (LBS) has drawn much attention from wireless communication to GIS applications. A survey of the LBS products from major GIS vendors ([HREF 4]) can be found in http://www.jlocationservices.com which is jointly sponsored by SUN and ESRI [HREF 5]. OGC has announced the wireless location services initiative [HREF 6] from GIS application perspective; LIF, the Location Inter-operability forum [HREF 7] jointly sponsored by Motorola, Ericsson and Nokia is established recently to allow user appliances and Internet applications to access location information from the wireless networks. The different focuses as well as the relationship between LIF and OGC are well recognized ([HREF 8]). OpenWave has already released Mobile Location Server and Mobile Management Server for LBS purposes ([HREF 9]). These organizations and their efforts will greatly prompt geographical information utilization. It is highly like that the geographical information captured by wireless devices will be further incorporated into other formats and applications.
Google ([HREF 10]) and other search engines roam over Internet to collect information thus can be thought as a kind of simple software agent. However, they only retrieve text information for key words matching and do not support geographic information. The natural language nature of mailing address needs more intelligent software agent to collect geographical information over Internet autonomously.

The concept of intelligent agents has gained much popularity in both academic researches and real practices, such as telecommunication (Chorafas, 1998), virtual shop navigation (Bensaid, 1999) and even auction (Noriega 1999). However, different from these action-centered agents, searching geographical information over Internet is more data-centered. Given huge amount of web pages (1,346,966,000, currently reported by Google), the complexity of agent and the processing speed must be well balanced. Google itself has tried to provide geographical search but the performance is very poor since it merely try to match all the words in an address as it does for keywords based search ([HREF 11]). Google doesn’t provide any visualization function for the searched geographical information either.

To manage geographical information that is essentially a special type of spatial information, spatial data access methods and spatial data analysis methods are needed. Spatial index methods, such as Quad-tree, R-trees are well studied in academic field (Shekhar, 1999). However, few mainstream commercial database software supports spatial data access. Oracle Spatial Option is one of them that support Quad-Tree/R-tree indexing methods ([HREF 12]). Spatial analysis methods, such as distance/direction analysis, buffer analysis, topological analysis, network analysis and more complex analysis, such as terrain analysis, are well studied in GIS field and implemented in many
commercial GIS software. However, these software are usually standalone and usually do not provide good integration mechanism with mainstream database software. The major problem is that different GIS software vendors have different geographical data formats that results poor interoperability.

Geographical information is distributed in nature since geographical entities and phenomena are geographically distributed. Considering vast volume of geographical information, it is also natural to think that geographical information search engines should be regional, i.e., each region has one or more search engines. It is logical to think that search engines in different regions are provided by different vendors using different data storages and processing methods, and thus integration is needed when user want to search bigger regions. XML based data integration architecture (Baru, 1999) is becoming more and more popular since XML is a kind of text-based protocol and can be widely accepted and easily processed.

GML, which is an extension of XML, is proposed my OGC to solve the interoperability problem. Several spatial data types, such as point, polyline and polygon, as well as earth projection types are defined in GML DTD. Any software that supports GML can use geographical data in a GML document. An XML-based spatial data mediation infrastructure for global interoperability study is conducted in San Diego Supercomputer Center (Zaslavsky, 2000). Based on GeoXML, which is similar to GML but using a smaller DTD set, the author also proposed and implemented a wrapper/mediator architecture to visualize integrated geographical information (zhang, 2000). However, data used in both studies is pre-collected and doesn’t make use of rich geographical information over Internet.
System Architecture

The Overall architecture of the system is shown as Fig. 1.

The system adopts a three-tier architecture:

1. The first tier is a regional server. A regional server has three components that are a software agent, a local database and a wrapper. The software agent roams over all the websites in region assigned according to its schedule, retrieves all the geographical information and their associated text information in web pages and then stores them in the local database. Since the majority geographical information involved is of point type, the database could be
relational database, object-oriented database or object-relational database. If the
database supports geographical information accesses and analysis, it should
register this information in the mediator that we’ll describe below in detail. The
wrapper extracts relevant data from database and transforms it into GML format
when a valid request is received. Since intelligent software agents can better
understand the context of geographical information, the search results can be
expected to be more relevant and more accurate.

2. The second tier is the Mediator. Whenever the mediator receives a
request from a client, it first determines which region or regions are involved
using its spatial analysis methods and break the query into sub-queries. It then find
the corresponding regional servers in its meta-database. If the corresponding
regional server support spatial data access and analysis methods, it then just sends
a sub-query to the regional server. If not, the mediator check the most recently
data received from that regional server. If the time span is within tolerance, the
mediator queries the duplicated data on that regional server’s behalf in its own
database, otherwise it retrieves the updated data from regional server and update
the corresponding control information. The mediator sends the integrated
geographical information also in GML format back to requested client.

3. The third tier is the client side. It supports geographical data
visualization based on spatial queries. The client side communicates with the
mediator in real time and retrieve GML documents from mediator. It then
translates the GML into Java objects and renders these objects graphically in the
applet. User can use Zoom In and Zoom Out to focus on different levels of
details. Besides visualization functionalities, textual information associated with geographical locations is also displayed in client’s browser and hyperlink can lead user to corresponding web pages.

Note that we use GML as the common communication protocol between distributed regional servers and mediator as well as between mediator and client. The reasons are as follows:

1. Text-based information exchange protocol ensures platform independence and easy implementation.
2. Data provider can be any system with any data format as long as it can generate XML document, which is an industry standard.
3. The generated XML document is application independent, which can be used in any applications as long as the application can parse XML document.
4. It is more efficient if there are many attributes in the whole data set but few attributes are associated with each object, i.e., relational table is sparse which is usually the case for geo-referenced information.
5. The possible reason against XML is that a text based XML document is much bigger compare to a binary one. However, it can be compensated by applying compression methods between the sender and retriever.

We have implemented a GML wrapper for flat file in our previous study (Zhang, 2000). To make our work more general, we’ll implement GML wrappers for relational database and object-relational database. Microsoft SQL Server 2000 is used as a representative for relational database and Oracle 8.1.7 with Spatial Option is used as a representative for Object-Oriented database. In our prototype system, one regional server uses Microsoft SQL Server 2000 and the other regional server uses Oracle 8.1.7. The
mediator uses Oracle 8.1.7 to provide spatial query and analysis functions. Java is used to implement software agents and wrappers for regional servers as well as mediator and client applets/servlets for visualization of query result.

**Implementation**

**Software agents for collecting geographical Information over Internet**

Since web pages are supposed to be maintained by non-professional individuals and are more likely in free text format, we cannot assume the structure, content and or context of these pages. Parsing geographical information from HTML pages is somehow like natural language processing (NLP). However, due to the huge amount of pages we are supposed to process and the high cost of NLP algorithms, it is not realistic to solely depends on NLP which are still under studying (Dale, 2000). Our proposal has the following components:

1. First make full use of website information. We can locate the approximate location of the website by its IP address and area code of the phone number of the contact person of the website. Since most geographical information over Internet are local, we can expect that geographical information in the pages are around the approximate location. Exceptions might be national websites such as CNN and the White House; ISP provider such as AOL and MSN; free web page hosting sites such as geocities, etc. We can build a look up table for these ”special” sites. There is no implementation of this feature in this prototype and we’ll leave it to future study.
2. Make use of structural information of HTML a page, such as tables, lists, etc. If a piece of geographical information is found in an item, it is likely that the similar information will be found in the same nearby items at the same level.

3. Integrate traditional Information Retrieval (IR), Information Extraction (IE), Natural language Processing (NLP) techniques. First, some keywords are more likely to associate with geographical information, such as “take place at”, “visit”, “drive to”, etc. IR technologies can also help to build some scenarios. IE methods can be used to fill the template of a scenario (Embley, 1998). NLP is used only when necessary, such as combining two parts separated by other components to get a piece of complete geographical information.

Personal geographical information, although could be mostly longitude/latitude in the future when GPS are widely used, is mostly likely to be in mailing address format or relative distance/direction to an outstanding place. It is the software agent’s responsibility to decide the geographical information is exactly in which format. If it is in longitude/latitude format, then we can directly extract and store it in its database. If it is address information, we can map it to longitude/latitude by using geocoding (or address matching (Drummond, 1995) technologies. If it is relative location information, we can first retrieve the longitude/latitude information of the outstanding place and then calculate its longitude/latitude information by spatial computation using the relative distance/direction information.
Retrieving accurate and most relevant geographical and its associated text information using agent technology is challenging yet difficult. Although currently there are a few companies specialized in geocoding technologies ([HREF 13-15]), the implementation detail is not available. Recently, OGC has unofficially released Geocoder Service Specification ([HREF 16]) and is expecting implementation compliances. We believe that Geocoding should not be a problem in our future implementation. However, in this study for demonstration purpose, we artificially build ten web pages in two different regions that have yard sale information. Since we cannot have free Geocoding software, we directly use longitude/latitude instead. Although they are much simpler than real pages that we are supposed to deal with, it provides a basis for further research.

The procedure is as follows: the agent of the regional server first sends request to the server hosting the yard sale pages. Then it parses the returned HTML pages into trees using html parser classes in SWING provided in JDK Enterprises Edition v1.3 from SUN (HREF 17). The regional server also tries to find keywords related to yard sale and return the positions of the keywords. Then it determines whether there is location and/or time information around the keyword. If there is then the server parses the location information using class StringTokenizer to separate x and y coordination and store them in the regional server’s database. The general process is shown in Fig. 2.
**GML Wrapper for Relational Database (SQL Server for example)**

Generic relational database, such as Microsoft SQL Server, doesn’t support user defined data type. Since most of geographical data stored in the database of regional server is point data, we can use two columns to store x and y coordination each in a special table to store the geographical information. We also have a column in this table named ObjectID to link geometric information and their related attribute information in other tables.
Open an URLConnection for giving URL

Open an InputStreamReader to read the HTML page

Build an HTMLEditorKit to process the HTML page

Create a Document from the kit to represent the page

Bind InputStreamReader and Document using HTMLEditorKit

Go through the hierarchy of HTML document tree

Get tag name, tag values and textual data

Parse the textual data for specific information

Item Info  Time Info  Geographical Info

Address Info  Geocoding  Distance/Direction

longitude/latitude  Spatial Computation

Global knowledge

Validation

Database update

End

Fig. 2 Process of Retrieving geographical information from web pages
To build a wrapper that transfers geographical data from SQL Server to GML, a Java Servlet similar to our previous work in wrapping flat file to GeoXML is developed (Zhang, 2000). Since our data are mostly point data this time, we only take point data wrapping into consideration. The communication between Java wrapper and SQL Server is through JDBC-ODBC bridge. The general process of building a GML wrapper for SQL Server is shown as Fig. 3. Different from our previous study which takes layer number as parameters, this program takes start time and end time as parameters.

Fig. 3 Process of building GML wrapper for SQL Server
We use different virtual directories to simulate websites in one machine. The syntax to retrieve a GML document from our website in this prototype is like:

http://ada.cs.ou.edu/RegionalServer1/servlet/RegionalServer1

Where RegionalServer1 stands for the first regional server. Accordingly, RegionalServer2 stands for the second, so on and so forth.

The wrapper will return all the yard sale records which the regional server retrieved within a predefined time period. Note that since SQL server doesn’t support spatial query, the mediator will communicate with this wrapper constantly to retrieve the newly added yard sale records, storing in the mediator’s database (which support spatial query) and let the mediator do spatial query on the wrapper’s behalf. They use GML as the communication protocol that we’ll explain in detail later.

**GML wrapper for Object-Relational Database (Oracle for Example)**

Oracle is a well-known object-relational database. It allow user to define new data types and analytical functions. The latest version of Oracle has several object-oriented options or cartridges along with it, such as Spatial, Intermedia, etc. Although Oracle spatial option stores geographical data in its relational tables, it allows user to store and retrieve geographical objects such as point, polyline and polygon. It provides spatial access methods such as R-tree and thus is able to perform spatial query and analysis.

Comparing to the full-fledged GIS software, the functionality of Oracle Spatial is limited. The key point of using Oracle Spatial is that its Object-Oriented feature allows integrating spatial query and analysis with relational operations seamlessly. Although our
implementation is based on Oracle 8.1.7, we’re expecting future version can provide more powerful spatial functionalities that seem quite possible.

Building wrapper for Oracle is relatively simple. We can issue any SQL query to Oracle and transform the query result into GML. In the SQL query string, we not only can use standard SQL operators such as “equal “,”like”,”group by” etc., but also spatial operators, such as “filter” operator (SDO_FILTER), “nearest neighborhood” operator (SDO_NN), “relate operator” (SDO_RELATE) and “within distance” operator (SDO_WITHIN_DISTANCE), etc. SDO_FILTER and SDO_RELATE are more query oritented and SDO_NN and SDO_WITHIN_DISTANCE are more analysis oriented from traditional GIS perspective.

The returned result is a set of objects, each object has both geometric data and associated attribute data. By looping through the result set, we can transfer a geographical object into an GML element easily. Fig. 4 shows the process of GML wrapper for Oracle 8i with Spatial Option.

Again we use different virtual directories to simulate websites in one machine. The syntax to retrieve a GML document from Oracle Wrapper is exactly the same as retrieving a GML document from SQL Wrapper:

http://ada.cs.ou.edu/RegionalServer2/servlet/RegionalServer2.html

Note that since the wrapper can support spatial query, it is better for the mediator to send queries to the wrapper and let the wrapper do the query. The advantage is that there is no duplication of data between mediator and regional server and the regional server always sends back the most recent data. It is easy to see that the data transfer
overhead is also reduced since the resulted GML documents of specific queries are much smaller than retrieving all the data from a wrapper at once as the wrapper for SQL Server does. It is also clear that building GML wrapper for Oracle is much easier since it has built-in classes (so called adapter) to transfer geometric classes into GML elements.

**Building a mediator that support spatial query**

We build the mediator on top of Oracle DBMS. The key advantage is that Oracle Spaital provide a utility that map GML elements in a GML document into its spatial objects and we can easily insert, delete or query these spatial objects in Oracle.
The mediator stores the region of each of regional servers as a polygon objects in an region index table. The mediator also have tables to store geographical entities and their associated yard sale information from those regional servers that does not support spatial query and analysis. The mediator communicate with these servers and copy new data to the corresponding tables since last updates.

Once the mediator receive a request from client, it first decides which regional servers are involved by issuing a spatial query to its region index table. If a regional server doesn’t support spatial query then it check the last update time, if the time span between current time and last update time is out of tollerence, then it sends a request to corresponding regional server to update its table otherwise it query the table directly. The query result will be transferred into XML document nodes for further process. If a regional server does support spatial query, then it send the request to the corresponding regional server and transfer the returned GML document into XML document node. These XML document node and their metadata information will then be put together to form a GML document and send back to client. The Process is shown as Fig. 5.
Client Request

Spatial query on Region Index table

Break down the query into queries for each regional server involved

Spatial Query Support?

Yes

Send sub-request to regional server

Transfer the returned GML document into XML nodes

No

Check last update time

Is update needed?

No

Send request to regional server

Update database in corresponding table with returned GML document

Yes

Query on corresponding data table in the mediator

Transfer query result into XML nodes

Combine all the XML nodes and associated metadata information

Output the Generated XML Document

Fig. 5 Process of building GML mediator
Client Query and visualization

In our previous study, we have implemented a Java applet to visualize a GML document. It supports Zoom In/Zoom Out and Full extent. To focus our effort on the architecture, we make use of open source software GeoTools ([HREF 18]) for visualization.

GeoTools is a Java class library for developing interactive mapping applets and applications which is developed at Center for Computational Geography, University of Leeds, UK. Its architecture is shown as Fig 6. GeoTools has most features of commercial GIS software, such as ArcView and MapInfo although its functionalities are still under growing. A theme is a logical unit of geographical data, such as a base map showing road or a collection of points showing yard sale locations. The viewer can render several layers in its predefined order and interacts with user. GeoTools allows putting a scaler on a viewer to show the current scale of a map, it also can add legends to a theme and show tips for polygonal objects. The results of client visualization using GeoTools will be shown in the next section.

Fig. 6 Architecture of GeoTools
Experiment and Results

We download the census track data from ESRI in Shape file format and put it into Oracle using a translation adapter provided by Oracle Spatial. We also generate 10 random points to represent yard sale locations and put them into 10 HTML pages. Two regional servers, one uses SQL Server 2000 and the other uses Oracle 8.1.7, are assigned 5 of the yard sale locations each. The mediator server is built by using Oracle 8.1.7 and its Spatial Option. The client visualization functions are developed by using a Java Applet.

Fig. 7 shows a very simple HTML page with yard sale information includes its location and items to be sold.

Fig 7 A Simple HTML page with yard sale information
Fig. 8 shows a GML document generated by a regional wrapper.

```xml
<?xml version="1.0"?>
<featurecollection>
  <point>73.384651,35.25922</point>
  <point>73.382591,35.355526</point>
  <point>73.26249,35.330742</point>
  <point>97.256108,35.12929</point>
  <point>97.275436,35.047115</point>
</featurecollection>
```

Fig. 8 A GML document generated by a regional wrapper
Fig. 9 Shows the GML Document generated by the mediator server by combing the query result of two regional servers.

Fig. 9 A GML Document Generated by the Mediator Server
Fig. 10 shows the yard sale locations overlapped with its base map and related information in a client side. The left part shows the items to be sold in each yard sale location. The circles in the middle part show all the yard sale locations with different colors indicating the relevance to a user query as the legend in the right part shows. Currently, the relevance or the ranks are generated randomly, but it could be very useful in visualizing query results.

![Client Visualization of A Query Result of Yard Sale Locations and their associated items and ranks](image-url)
Discussion and Future Directions

The volume of personal geographical information will be dramatically increased in the near future due to the wide application of GPS and cell phone. It is highly like that these information will be used beyond the centralized systems where they are generated. They will be integrated into the main stream web HTML files in a loosely coupled and distributed fashion. Thus a geographical information searching engine over Internet similar to popular text-based information searching engine is highly desirable.

The prototype we have built so far is pretty simple comparing to the goals we are supposed to achieve. Nevertheless, it proves that our design is feasible and lays a good foundation for further research. Our work can be summarized as follows:

1. An open architecture is proposed to collect regional geographical information using software agent technology and GML as the common communication protocol. The negotiation mechanism is proposed and implemented between the mediator server and regional servers.
2. Building GML wrappers for both pure relational database like SQL Server 2000 and object-relational database like Oracle 8.1.7.
3. Building a mediator server with full spatial query capability by making full use of Oracle Spatial functionalities.

Future directions may include the followings:

1. Parsing real HTML pages and searching for geographical information in them. Semi-structured document processing techniques (including...
Information Extraction and Information Retrieval) as well as natural language processing technologies (NLP) are needed for accurate and efficient geographical information searching.

2. Geocoding methods, which are very important for geographical information transformation and query, need to be fully implemented. It is even more challenging when multiple languages are considered since the expression of address is highly related to language and culture in general.

3. Spatial and temporal representation of geographical information. Currently no commercial database system support spatial and temporal information of an object. We need to investigate more on Oracle as well as other database systems to find a way to represent both spatial and temporal aspects of geographical information to support advanced queries. A natural extension of this research direction is how to deal with semantics of geographical information which seems more and more important.

4. Scalability study. Considering the huge amount of web pages (which is more than 1 billion), we need further investigations on the scalability of the database systems we are using to achieve our goals.

5. More flexible client query and better visualization functions. There are always much more we can do on user interface.
References


10. S. Shekhar, S. Ravada, X Liu, Spatial-Databases-Accomplishments and Research Needs, IEEE transaction on Knowledge and Data Engineering, 11(1), 1999


## Appendix: Source Code List

### Core Programs:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
<th>LOC</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegionalAgent1.java</td>
<td>Software agent program to search through all the pages in region 1 and put the search results into SQL Server database.</td>
<td>150</td>
<td>4KB</td>
</tr>
<tr>
<td>RegionalAgent1.java</td>
<td>Software agent program to search through all the pages in region 1 and put the search results into Oracle database.</td>
<td>242</td>
<td>8KB</td>
</tr>
<tr>
<td>RegionalWrapper1.java</td>
<td>Wrapper program for SQL Server (Servlet)</td>
<td>69</td>
<td>2KB</td>
</tr>
<tr>
<td>RegionalWrapper2.java</td>
<td>Wrapper program for Oracle (Servlet)</td>
<td>229</td>
<td>5KB</td>
</tr>
<tr>
<td>BMServer.java</td>
<td>Servlet program to retrieve base map from Oracle and transfer it into GML document.</td>
<td>225</td>
<td>8KB</td>
</tr>
<tr>
<td>MDTServer.java</td>
<td>Servlet program to build the mediator server</td>
<td>419</td>
<td>15KB</td>
</tr>
<tr>
<td>TestYardSale.java</td>
<td>Applet program for client visualization</td>
<td>172</td>
<td>6KB</td>
</tr>
</tbody>
</table>

### Utility Programs

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
<th>LOC</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>InsertBM.java</td>
<td>Insert base map into Oracle database by extracting geometric objects from ESRI shape file.</td>
<td>128</td>
<td>6KB</td>
</tr>
<tr>
<td>InsertMeta.java</td>
<td>Insert metadata information into Oracle database</td>
<td>156</td>
<td>6KB</td>
</tr>
</tbody>
</table>