Web-based visualization of vector and raster data from spatial DBMS using a geo-browser

Hyung Woo Kim[1] and Yang-Won Lee[*2]
[1]Graduate student, Department of Spatial Information Engineering, Pukyong National University,
Daeyeon 3-dong, Nam-gu, Busan, Korea; Tel: (82)-51-629-6660; Fax: (82)-51-629-6653;
E-mail: kalituma@gmail.com

[2]Assistant Professor, Department of Spatial Information Engineering, Pukyong National University,
Daeyeon 3-dong, Nam-gu, Busan, Korea; Tel: (82)-51-629-6660; Fax: (82)-51-629-6653;
E-mail: modconfi@pknu.ac.kr

KEY WORDS: Web GIS, Spatial DBMS, Mashup, Geo-browser

Abstract : Through overall transition of paradigm of GIS, geo-visualization also extended traditional cartographic approaches by display of map that users interact with data dynamically. There is distribution of Web 2.0, including 'AJAX', at this dynamical function. GIS could develop revolutionarily and user's attention about GIS has increased due to such effect. In addition, as ordinary users who aren't experts in GIS could visualize and handle maps by using API for utilizing maps on web, distributed GIS entered into renaissance of mashups. Google API which have provided and updated steadily was a starting point in change of GIS by mashups. Implementation with such API becomes a typical form of development in recent.

In aspect of data, as past DBMS which had a limit in storable type have advanced on diverse data type, like spatial, media, and so on, distributed GIS which use heterogeneous data had enough and stable space for storing their data on web.

In this article, to integrate various components of distributed GIS, which are affected by advanced Web, and to implement distributed GIS which has geographical functionality, starting with geo-visualization, we implement visualizing system for spatial data (vector, raster) with mashups. Vector data in database is point data of observed numerical value and raster data is binary data by remote sensing. Overall structure can be divided three parts, database server, client and brokerage module which is the key of this system. Used DBMS is PostgreSQL/PostGIS for spatial database, and brokerage module is constructed by Java language. Google Earth, Chart API and Javascript are used for visualizing data on browser.

1. Introduction

All aspects of geographic information system have undergone paradigmatic shifts in architecture, approach way, data format, and platform. It's true that visualizing parts of GIS was affected by these shifts. A typical example of the effect is that geographic visualization extended traditional cartographic approaches by linking the visual map display with both the underlying geographic data structures and the system users dynamically (MacEachern et al, 1997)

A current trend of geo-visualization is mostly realized by distributed GIS based on web technology and can practice dynamically through interaction with users. The technology to permit such dynamic interaction is precisely web 2.0 which pivots on AJAX (Asynchronous Javascript and XML). Although AJAX isn't advanced technology but blend of techniques which also existed in the past, but it can implement dynamic UI (User Interface) and visualization by asynchronous communication. (Garrett, 2005) Various API (Application Programming Interfaces) which affected dissemination of recent GIS considerably for easier development and diverse function also is executed by Javascript, a core technique of AJAX.

Thanks to such foregoing development, distributed GIS have developed revolutionarily and gotten ever-increasing attention. (Chow, 2008) Circumstantial effects of web, indeed, let the group of none-expert users, even they don't know about GIS, to be able to create and manipulate maps. And it rearranges the influence of map on web by 'mashups'. (Batty et al, 2010 and Butler, 2006) A mashup refers to a web page or application that combines data or functionality from two or more external sources to create a new service using API. (Peenikal, 2009) Google Earth and Maps API are representative ways to realize mashups in aspect of geographical information. Especially Google Earth has more outstanding expressiveness than other geo-browsers and supports diverse computer languages, HTML, Javascript, Flash, etc. In addition, it allows more sophisticated use of the service by the release of the KML (Keyhole Mark-up Language) grammar specifications. (Henry, 2009) Not only Google Earth but also other API, like Google Chart and Maps API, can be used to specify information for spatial data.

On the other hand, DBMS (Database Management System) advanced as much as web technology did. It can store database as original form and apply to diverse storage method about spatial database, like spatial, media data, and so on. Expanded accessibility by compatibility with other computer language is also advantage of recent DBMS.

For such a diversification of resource of distributed GIS, demands for integrated systems which have geographical functionality and can handle dispersal and heterogeneous data have increased. So as to contribute to satisfy such demands, this research constructs visualization system for vector and raster data by mashups between Google API codes.

2. Specification of System

2.1 System Structure

Whole structure can be divided into three parts, database-server for managing diverse database, brokerage module for linking data with client, client for user interface and visualization.

At first, we choose PostgreSQL/PostGIS for DBMS, which is provided for free and can save and handle spatial data (raster, vector) by PostGIS, the application adds support for geographic objects to the PostgreSQL. Types of data used in this system are vector and raster. Vector is point data, consists of observed attribute data and location of observing station. Raster data is binary format data observed by satellite.

The brokerage module is a middleware located between web client and DBMS. It has charge of role for transforming rare data into image file, constructing KML for displaying on Google Earth plug-in and linking client with database to blend top and bottom layer's functionality. It can communicate with both database-server and client. Brokerage module is constructed by JAVA language and stored as library on the server, so developer can transplant to other environment as it stands.

Lastly, the client is user's interface tools to select and to transmit their terms to middleware. Besides it can display the data from server and brokerage module on geo-browser. Used language is Google Earth and Chart API, and Javascript, known for core technology of web 2.0, represented by asynchronous communication. (Garrett, 2005)

Operational order of the system is figure 1.

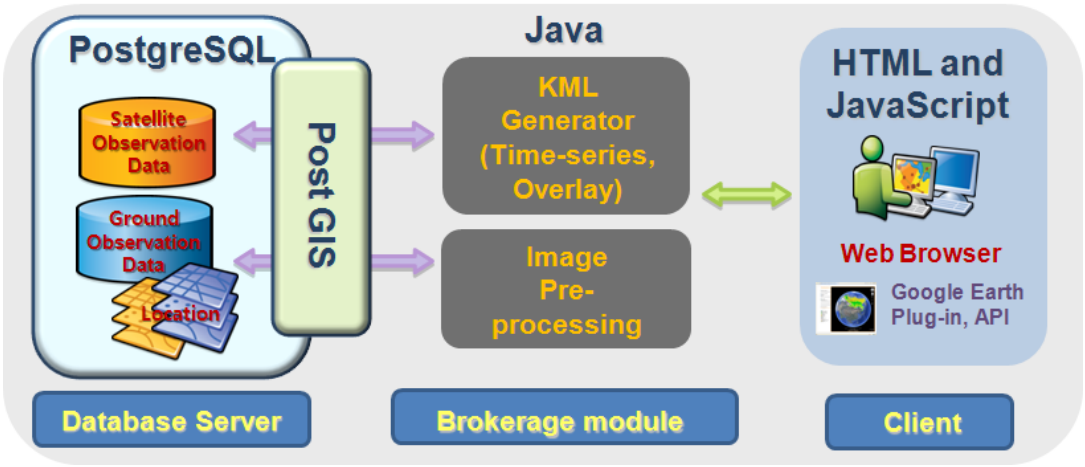


Figure 1 System Structure

System needs 'brokerage module' as bridge, the key of the system for communication among each components, for interchanging database and flags of terms because geo-browser cannot communicate with DBMS directly and get database in server. DBMS and client really intersect in brokerage module and could communicate each other.

Briefly explaining of figure 1, system operation starts at client by user's selection of terms and client interchanges the 'flag' with brokerage module, then brokerage module queries to DBMS, finally brokerage module transforms the dataset from DBMS into visual forms which are visualized by web client on geo-browser.

2.2 Database and Query

The system in this research has two types of data as mentioned above, vector and raster, and visualize through three-way methods : visualizing observation data, raster rata, and times-series animation.

Vector data is consists of location and attribute data based on ground observation, raster data is binary data based on remote sensing. Logical structure of database can be also classified along data type. Vector has two tables which have a same key, first table stores coordinates of observing point and another one has numerical data of attribute. Raster data also has tables, as many as number of columns, which is put binary in each row.

When the system queries for using data, the brokerage module configures statement after reflection of user's terms. Terms which should be considered for statement are date, location, and time range which users want to see. The dataset after query is sent to brokerage module and revised for using on geo-browser.

3. Implementation

In this chapter, case of implementation, we should mainly examine middleware and client's cooperation because almost functions of the system rely on cooperation, which could change by type of data, between middleware and client.

There are three implementation by brokerage module and client as briefly mentioned in chapter 2 :
First, which is a structural role that mainly connect and communicate logically, is linking between layers (DBMS and client) by referring to user's term and querying SQL.
Second, which is important for visualization, is generating KML and image file and overlaying for visualization on web. This is can be divided according to data type, whether users choose vector or raster.
Third is accepting user's term, like time, location, etc., for searching database.
In this chapter, we explain second and third implementation in detail except first.

3.1 Overlaying Placemark and Chart

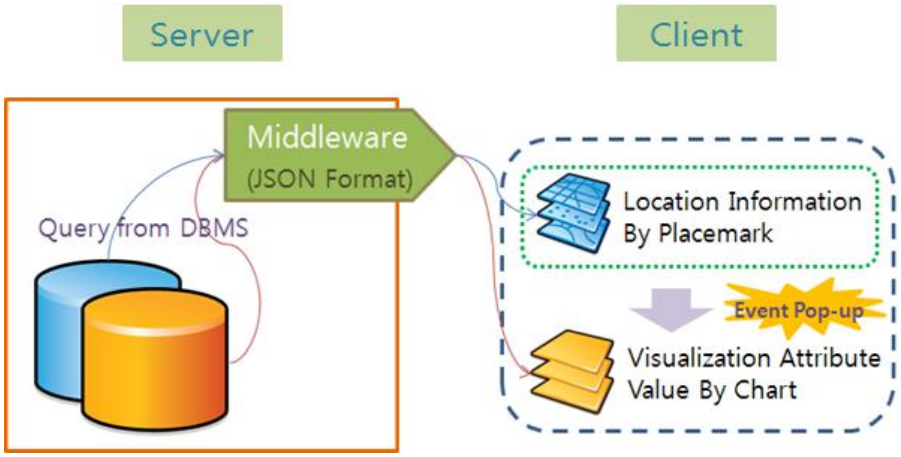
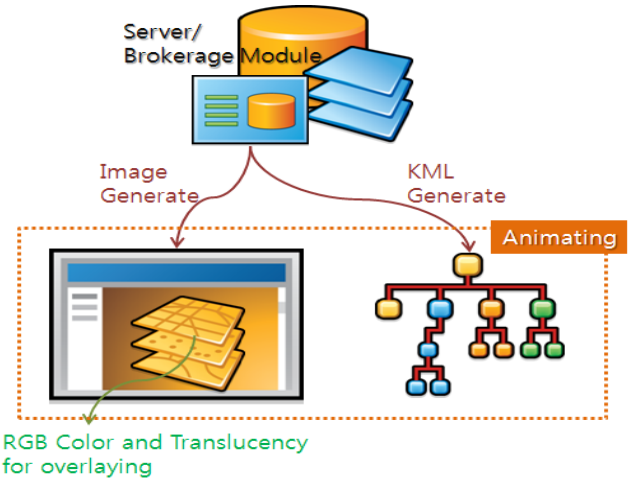


Figure 2 Displaying Method of Point data

Vector data is visualized through 'placemark with chart' by assembling location information with attribute data. After query to DBMS, brokerage module converts dataset to Json format for forwarding to client easily. Client which received data classifies data in category of location and attribute by tag. Above all, it's the point that client generates placemark on Google Earth by location data and wraps the placemark in event-listener so that user can check attribute information by pop-up chart. Chart type used in this system is line chart and x-axis is time-series and y-axis is attribute value. Figure 2 explain whole process of visualizing vector data shortly. The biggest advantage of this implementation is that placemark and chart could be realized by simple API code.

3.2 Generating Image and KML

Raster data, after going through several steps, is transformed into image and visualized on Google Earth by overlaying image and KML. Left of figure 3 is process of generating image and KML file for animation and raster overlay.
After query of binary database, dataset for raster will be sent to the brokerage module. Then brokerage module interprets binary dataset as float array and record the array to image format by library. This library performs to generate image file, by parameter and formula. The library finds maximum, minimum, and null value before calculation. Then method in library will compare float array from binary data set with maximum, minimum and null value and define the color's value of each cell considering each parameter. If value of each cell match the null value, color combination of RGB will be 255,255,255 showing white color. Otherwise it will set limit to value of each cell between maximum and minimum and calculate R, G, B value by using formula "255*(Value-minimum)/(maximum-minimum)". After that calculation, the library will configure image, reflecting each cell's color and translucency. So when we overlay this image on Google Earth, we can recognize both of geographical and attributional features.




```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
<!-- TimeSpan is recommended for GroundOverlays -->
  <Folder>
    <name>Name for Folder</name>
    <GroundOverlay>
      <name>Name for GroundOverlay</name>
      <TimeSpan>
        <begin>Start Day or Month(formed yyyy-mm-dd)</begin>
        <end>End Day or Month(formed yyyy-mm-dd)</end>
      </TimeSpan>
      <Icon>
        <href>Path of Image for Linking Each Other</href>
      </Icon>
      <LatLonBox>
        <north>Northermost Coordinates of Image</north>
        <south>Southernmost Coordinates of Image</south>
        <east>Easternmost Coordinates of Image</east>
        <west>Westernmost Coordinates of Image</west>
      </LatLonBox>
    </GroundOverlay>
    ......
```

Figure 3 Generate Image and KML for animation

To implement overlay or animation on Google Earth, we need not only image file but also KML file. Right of figure 3 is main part of generating KML format file for overlaying raster image and realizing animation. Almost of part is consists of tags like "<LatLonBox></LatLonBox>", "<TimeSpan></TimeSpan>", etc. and It can provide dynamic function in google earth.

Important parts of this research are functions of each tag. <LatLonBox> tag defines boundary of overlaying image and <TimeSpan> can connect number of sequence of linked image, which formed 'yyyy-mm-dd' or 'yyyy-mm' to configure animation data of Google Earth. And <href> tag define path of linked image. (Wilson, 2008)

3.3 User Interface

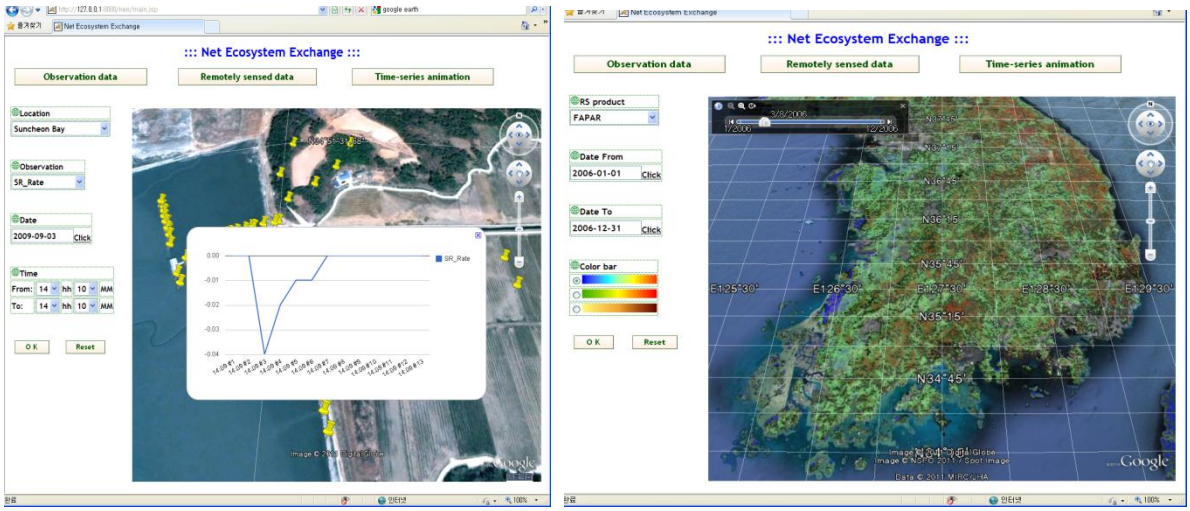


Figure 4 Implementation Result

We operated test for implementation and used data in test was 96 points vector data and binary image data of South Korea. Geographical range of data is, vector is in Suncheon and Busan, Korea, raster's range is between upper left, 39.9999, 124.481, and lower right, 32.504, 130.008.

Figure 4 are implementation results. Left of figure 4 is page for visualizing point data about observed temperature, soil respiration rate and etc., and right is page for displaying temporal raster image and animation about FPAR(fraction of absorbed photosynthetically active radiation), GPP(Gross Primary Production) and etc.

In case of user interface, user can use three tap buttons in top of page. Each of buttons can change menu for selecting user's term in left of screen. First button is for vector data (observing point data), second and third one is for raster data (remotely sensed data and time-series animation). In menu for vector data, user can select location, data column, date, time range for querying data and it's same as raster menu except location.

If user clicks 'OK' button after selection of all terms, Google Earth plug-in in right frame will be overlaid with placemarks or image data like figure 4. In case of vector data, user can check chart pop-up for expression of attribute value when they click the placemark. Browser provides drag-interface for switching the images when users select animation menu.

4. Conclusion

This research suggests that possibility of visualizing system by using disparate web API, and more generally, scalability of function of distributed GIS based on Internet. In particular, through using data of different type and implementation of GIS based on web, we can seek a way to visualize each database and check the potential of mixed utility of various API.

For visualizing database, Google Earth and Chart API were used on vector data, divided into geographical and attributional category, and raster data was transformed by middleware library. In case of vector data using mixed API to show themselves, it means that development of GIS system for visualization is easier than past era owing to advance of web technology. And in case of raster data, using library for production of image file and KML, it means possibility to actualize more advanced functions than a visualizing function. If the middleware for processing of data improves and expands gradually, this system can be closely to completion distributed GIS module.

Henceforward we should emphasize 3-dimensional feature of Google Earth and improve expressiveness by combining with 3D model for visualizing real world. Information delivery by virtual reality can be charming to curious and potential users. 3D can also provide the way to inform effectively. For example, if we use not only simple vector and raster data in 2D but also more complex data like polygon in 3D, the system can express diverse information at once by loading information on each dimension. And we should update this system for total GIS analysis system which can do geographical calculation because it is a direction of distributed GIS at this time. Lastly, our future work includes the concept of ‘Web 2.0’ and ‘semantic web’ by replacing of DBMS with database formed web document like XML, sharing data by applying LOD (Linking Open Data) and realizing ontology, and providing API by web services.

Acknowledgements

This work was researched by the supporting project to educate GIS experts.

References

Batty, M., Hudson-Smith, A., Milton, R., and Crooks, A., 2010, Map mashups, Web 2.0 and the GIS revolution, Annals of GIS, 16(1), pp. 1-13

Butler, D., 2006, Mashups mix data into global service, Nature, available at http://www.nature.com/nature/journal/ v439/n7072/full/439006a.html

Chow, T., E., 2008, The potential of maps APIs for Internet GIS applications, transactions in GIS, 12(2), pp. 179-191

Garrett, J., 2005, Ajax: a new approach to web application, available at http://www.adaptivepath.com/ ideas/essays /archives/000385.php

Henry, A., 2009, Using Google Earth for Internet GIS, MSc Dissertation, University of Edinburgh

MacEachren, A., and Kraak, M., 1997, Exploratory cartographic visualization: Advancing the agenda, Comp. & Geosc, 23(4), pp.335-343.

Peenikal, S., 2009, Mashups and the Enterprise, MPHASIS white paper, pp. 2-5

Wilson, T., 2008, OGC KML 2.2.0, OGC