A Case Study on Geospatial Data Management with Blackbook

Technical Report UTDCS-28-09

Department of Computer Science
The University of Texas at Dallas

September 2009

Sonia Chib, Jyothsna Rachapalli, Bhavani Thuraisingham, Latifur Khan and Murat Kantarcioglu

A Case Study on Geospatial Data Management with Blackbook

Sonia Chib, Jyothsna Rachapalli, Bhavani Thuraisingham, Latifur Khan, Murat Kantarcioglu

Department of Computer Science, The University of Texas at Dallas
800 West Campbell Road
Richardson, TX, 75083-0688, USA
{scc01000, jxr061100, Bhavani.thuraisingham, lkhan, muratk }@utdallas.edu

Abstract. Geospatial Data Management involves creation of a Web based GIS application which queries datasets stored in a spatially Mysql Relational DBMS. However, before the data can be queried from the database it needs to be translated from a heterogeneous source that follows different open-source file formats. Data heterogeneity and migration to a homogenous open source file format presents a major challenge. The geodatabase thus created stores large amounts of geographic data that conform to a single file format. The spatial queries produces an efficient result, as they utilize the index created on geometry of the geographic objects stored in the database. This project is now integrated with BLACKBOOK. BLACKBOOK provides infrastructure for semantic web and contains data which is a collection of disparate data sources. This serves as an entry point for this project. The purpose of this project was to enable Blackbook to find points within the specified proximity of a given point using the geospatial data source provided by HSIP.Further it was required to display this data in a user friendly manner which was done essentially using Google Maps API. This project was conceptualized to capture this essence.

Keywords: Geometry, Blackbook, Google map, Spatial operator, Spatial database

1 Introduction

Spatial data is becoming increasingly important in various sectors, public as well as private and with the advent of mechanisms to store and manage large amounts of spatial data; it has resulted in a need to efficiently execute queries on spatial data. This paper attempts to outline the steps involved in creating an enterprise application that processes large amounts of spatial data.

Spatial objects are representations of real world objects stored in a spatial database. Spatial data also known as geospatial data or geographic data is used to identify the location of geographic features on Earth such as rivers, streets, schools, states etc. A spatial database provides abstraction to represent these objects in the form of points, lines and polygon geometry types and additionally provides several operators that can be used to establish topological relationships between these objects. Topological relationships like overlap or intersects, contains, touches and disjoint are used to describe the position of spatial objects relative to each other.

Interoperability has become a chief concern these days as there needs to be cooperation among various heterogeneous proprietary formats being followed by different organizations. The data given to us is in several GIS formats, which gives rise to the problem of heterogeneity. In order to efficiently utilize the data, it needs to be homogenized. Translation/Migration of spatial data are particularly challenging as they involve complex geometries that need to be processed. Several tools are available, commercial as well as open source, which can be used to accomplish this. However the task of migration still remains time consuming and demanding because of the large size of the datasets and nature of the data stored in them.

Additionally the process of translation from one format to another can occasionally become prone to loss and error.

Once the data has been migrated to a spatial database, it is used as backend data source for the Geospatial data management (or Geospatial Proximity) project. Finally, the Geospatial Proximity project is integrated with the Blackbook to improve the usability for future research.

The rest of this paper is organized as follows. Section 2 briefly states the problem to be solved. In section 3, an overview of several Geo spatial data concepts and techniques is discussed. Section 4 addresses steps taken to integrate Google Map with BlackBook. Section 5 describes a high level architecture to utilize the homogenized geospatial dataset for web applications. The final section, section 6 concludes the paper.

2 Problem Statement

Given a location on map which is represented by latitude/longitude, find all points of interest within the proximate distance. The location is provided by the Blackbooks' internal data sources. The objective of BLACKBOOK is to provide analysts with an easy-to-use tool to access valuable data that can be used to make logical inferences across data sources. The points of interests are located from a huge geographic data. The geographic data that is provided is stored in several different GIS file formats. For example: ESRI Personal Geodatabase, ESRI Shapefile, ESRI SDC are among the formats of the given data. First three can be categorized under vector data and last one is raster data.

A GIS web application requires such disjoint set of data to be interoperable. GIS interoperability implies the ability to exchange information between different geospatial data storage system, even though it has been developed by different organizations on different platforms. Once the data has been translated/migrated using the available spatial data translation tools, a spatial index needs to be built on the geometry column in order to enable queries to efficiently retrieve them. Subsequently it is required to write an efficient SQL query that utilizes the spatial index. Further it is required to assess the performance of the query and make necessary alterations if not sufficiently fast.

3 Geospatial Database Management

Spatial data is the data that indicates the location of geographic features on Earth. The geographic locations like states, roads, political boundaries that exist on the surface of the Earth are projected onto a two dimensional display. The geometry associated with a geographic feature is an ordered sequence of vertices that are connected by circular arcs or straight line segments. The vertices are an ordered pair of coordinates and have an associated Spatial Reference Identifier (SRID).

Spatial Reference Identifier: It is a combination of an ellipsoid, a datum using that ellipsoid, and geographic coordinate system. The one used by the most of the given data is WGS84 (EPSG:4326).

GIS broadly uses two kinds of data structures to represent data:

Vector data: The objects are represented by points described by their co-ordinates in the associated spatial reference system. Vector representations are very compact resulting in low disk space consumption. However, it is considered to be more complex than raster data. Vector data can be of type point, line or polygon geometry. Physical storage of geometry for features is managed using standard data types. The geometry storage types available to use depend on database management system (DBMS) being used. A few databases have spatial data types while others provide binary large object (BLOB) storage types.

Raster data: It provides one of the simplest representations for GIS.The objects are based on the elements of a matrix. The geometry of such an element or pixel is given by row and column indices of that element.

· *Spatial Index:* The minimum bounding rectangle (MBR), also known as bounding box, is used to represent the maximum coverage of a 2-dimensional spatial object (e.g. point, line, polygon) within its two-dimensional coordinate system. To make the spatial query using buffer operator faster an index is built on the geometry attribute of the spatial object. A common data-structure used for creating index on spatial geometries is R-Tree. It is tree data structure that is similar to B-Trees, but is used for spatial access methods. It splits space with hierarchically nested, and possibly overlapping, minimum bounding rectangles MBRs.

To perform a search operation a search rectangle is given as input. The search starts from the root. Each internal node contains pointers to the corresponding child node and each leaf node contains MBRs of spatial objects. For every rectangle in the node it is checked to see if it overlaps the search rectangle or not. If it overlaps the child node has to be searched further and similarly search proceeds through the overlapping nodes till leaf node is reached. Then contained MBRs are tested against the search rectangle and the objects that lie in the search rectangle are returned in the result set.

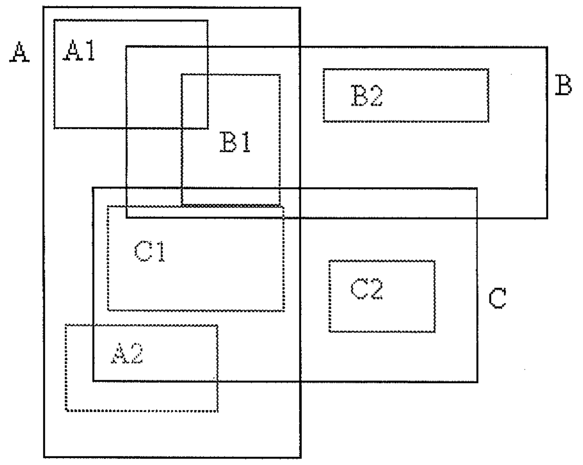


Figure 1: MBRs of geographic regions

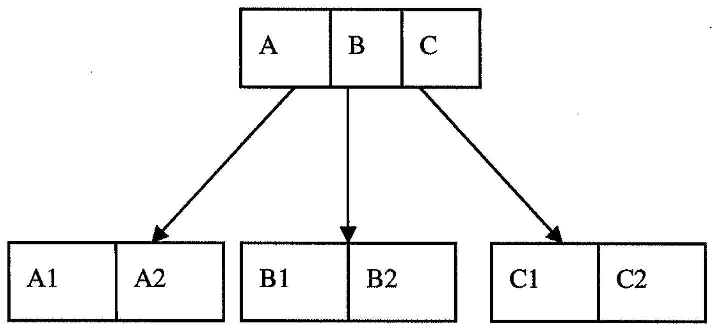


Figure 2: Rtree for the above set of rectangles

The smaller regions contained within the larger regions are arranged in the tree in such a manner that smaller regions form the child nodes and large regions containing these regions form the parent of these nodes as shown in Fig1 and Fig2.

3.1 Proposed Solution:

a) Data Migration

Majority of the given spatial is in ESRI's proprietary and open source file formats. ARCGIS is a tool provided by ESRI that provides a standards-based platform for spatial analysis, data management, and mapping. ARCGIS was used to view the data and make some data translations within the file formats supported by ESRI. Since this is an open source project the data was translated from ESRI file formats to Mysql.

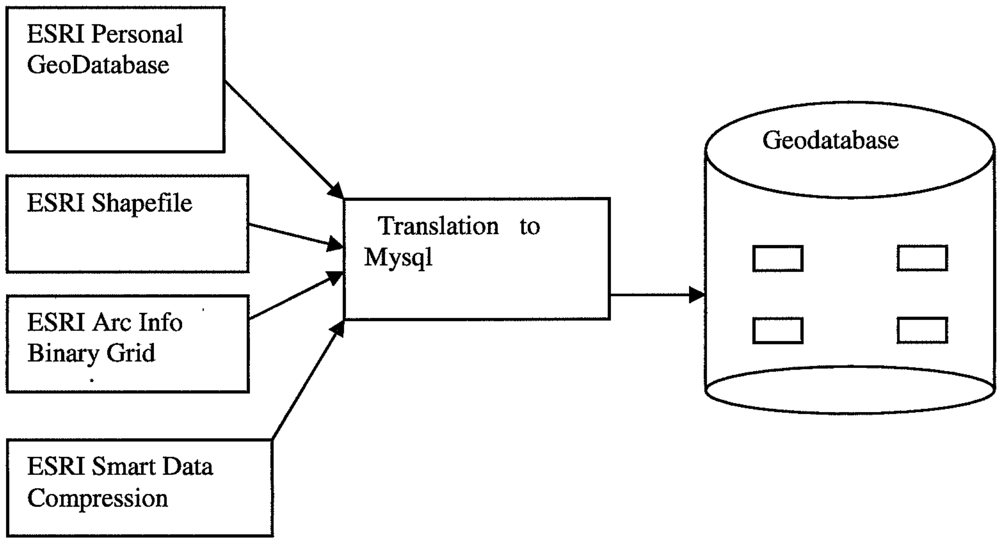

ESRI Personal GeoDatabase
ESRI Shapefile
ESRI Arc Info Binary Grid
ESRI Smart Data Compression
Translation to Mysql
Geodatabase

Figure 3: Data Translation

In addition ARCGIS is not supported on UNIX and Linux Platforms. Further it supports only stand alone and offline applications. ARCGIS does not support real time querying from an online Google Map. There are two alternatives which can be followed in order to use the data (given in ESRI's file formats).

First approach would have the GIS data being queried offline. Then convert the result of this query to KML that Google earth or Google map would understand and view the data on the map. Since this is static and slow, it is not acceptable for finding points of interest from a web page.

Second approach would be to convert the ARCGIS data into open source file formats. This new data that follows open source file formats can be used by any web application to query and manipulate the display to suit their query requirements. Mysql can be used to store the spatial data when it consists of simple geometry like point data.

However when the data consists of more complex geometries like point, polygon and polyline it is better to store in Postgresql's spatially enabled database know as postgis. The buffer queried that uses buffer operator is used to retrieve the points within the specified proximity with high accuracy. However, the response of buffer query in Postgresql is too slow to be practically useful for the application.

In order to improve the response time, the complex geometries were simplified by converting them into simple geometry type like point. This is done by replacing the complex geometry type with its centroid.The query response time is significantly smaller using this approach.

b) Tools Used For Migration

There are several open source as well as proprietary tools that aid in translation of spatial data. A list of such tools used for translation of spatial data from one format to another is shown in Table1.

Table 1: Migration Tools

Source Format	*Tool Used*	*Destination Format*
ESRI Personal GeoDatabase	ArcGis (Arc Toolbox-conversion tools)	ESRI Shapefile
ESRI Shapefile	VB Script	ESRI Personal GeoDatabase
ESRI Personal GeoDatabase	OGR2OGR,FME safe software	Mysql
ESRI Shapefile	shp2mysql	Mysql
ESRI SDC	ArcGis conversion tools	ESRI Shapefile

Spatial ETL tools: These are Extract, Transform and Load tools that can read, write and manipulate spatial data. ETL tools are like information channels that connect two different systems. The Extract function reads data from a specified source data source and extracts the required data. Then the Transform function processes the data thus obtained, transforming it and even possibly combining it with other data to tie it together into the correct structure for the destination database. Finally, the Load function writes the resulting data to a target database. These tools provide the organizations with an opportunity to improve productivity and take better advantage of the data that they already have.

Procedure for data conversion using FME safe software:

Following steps are used to perform a quick automated translation using, the same source and destination coordinate system.

- Drag a source file(s) and drop it in the FME window.
- Select your destination format which can be either MySql or PostGIS.
- Browse to a destination folder and enter a filename.
- If more than one mapping file appears in the Mapping Files area, choose the Automated Translation.
- Press OK.

FME will start the translation, and the output will be stored in your destination folder.

4 Google Map Integration with BlackBook

Google Maps is the best option to view geospatial data on web. To integrate the Google Map with the web application, we need to pass point data to the Google Map APIs.
The geospatial data can be stored in the Postgresql. EJB 3.0 has in-built hibernate component.
Access the geospatial data using Hibernate SQL .Once we fetch the points of interest from the Postgresql, next step is to pass this point data to the Google Map API. The API will plot the given Point on the google map at the client side.

To integrate the google in a website, one must include a Google Map Key in the html/xhtml page. the Google Map key can be located at the google map key sign up. The key provides a way to access the Google Map Api from any webpage. we can pass the Points of interest to the html/xhtml page from a java class(a bean or simple jdbc java code). Then from html client we can pass those point to the map api. The Google map api support asynchronous calls, hence there is no blocking at the client side. When the data returns from API, there will markers shown on the Google map.

A brief outline of how the Google Maps API key is brought in so the GeoSpatial data can be presented. The following documents changes that need to made to these files:

- Property default-google-key is added to .m2/settings.xml.
- Token GOOGLE_KEY is added to blackbook-war/pom.xml.
- Managed Property gkey in the faces-bean.xml.template
- Google key in GoogleMapResultsPanel.xhtml is retrieved from gkey.

***setting.xml**

-
  - Presently, Property default-google-key contains the key for https://localhost:8443/blackbook/ .
  - To make the geospatial component work on other IP address, obtain a new Googlekey based on the server's IP address from google here.
  - To obtain the key you have to accept the terms, and provide the url for the web application.The Google key will be valid for that corresponding URL only.
  - Place that key value after "key= " subpart of the <default-google-key> in the **settings.xml** file as follows:
  - Add within the <profiles></profiles> tag:

```
 <profile> <!-- default Google API Key to call  -->
   <id>blackbook-devel-configuration</id>
     <properties>
       <default-google-
key><![CDATA[http://maps.google.com/maps?file=api&amp;]]>amp;<![CDATA[v=2&amp;]]>amp;
<![CDATA[key=ABQIAAAAwv1iGmoHRlHUFEkrOWIMkxT_D2nltWVOjMhMSrkFYSsFd04bKx
QiBsC29glgzh9hLbjgu7Dhe6KO4g]]> </default-google-key>
    </properties>
 </profile>
```

The key should be changed to the key obtained for that machine as per the instructions above.

***pom.xml**

-
  - This value is pulled into pom.xml which should have the entry for
  - <artifactId>maven-antrun-plugin</artifactId> edited to include the final filter shown below (only the final filter is new):

```
 <copy
 file="src/main/webapp/WEB-INF/faces-beans.xml.template"
 tofile="src/main/webapp/WEB-INF/faces-beans.xml"
 overwrite="true">
   <filterset>
     <filter
       token="SVN_REVISION"
```

```
    value="${svn.revision}" />
   <filter
    token="REPLACE_NOTE"
    value="This file was generated on ${build.date}, do not edit since it will be overwritten." />
   <filter
    token="GOOGLE_KEY"
    value="${default-google-key}" /><!-- Google Key from the settings.xml -->
  </filterset>
 </copy>
```

***faces-bean.xml.template**

-
    - This value then needs to be pulled into the faces-bean.xml.template so that it can be accessed within the interface. This is done by editing the GoogleMap Bean to have the additional managed-property for googlekey.

```
<managed-bean><!-- Added for geospatial component -->
 <managed-bean-name>GoogleMap</managed-bean-name>
 <managed-bean-class>
  blackbook.faces.NavigationAction
 </managed-bean-class>
 <managed-bean-scope>session</managed-bean-scope>
 <managed-property>
  <property-name>label</property-name>
  <value>Google Map</value>
 </managed-property>
 <managed-property>
  <property-name>path</property-name>
  <value>results/GoogleMapResults.faces</value>
 </managed-property>
 <managed-property>
  <property-name>gkey</property-name>
  <value>@GOOGLE_KEY@</value>
 </managed-property>
 <managed-property>
  <property-name>outcome</property-name>
  <value>GoogleMapResults</value>
 </managed-property>
</managed-bean>
```

- Now this value can be accessed and used for the GoogleMaps in **GoogleMapResultsPanel.xhtml** like so:

```
<script src="#{GoogleMap.gkey}" type="text/javascript"></script>
```

5 Geospatial Proximity Implementation

Overview

Once multiple datasets has been homogenized to a single coordinate system and converted to MySql format, the next step is to use this spatial datasets for locating points of interests within the specified proximity. Later, we will analyze the response time for finding the points of interest.

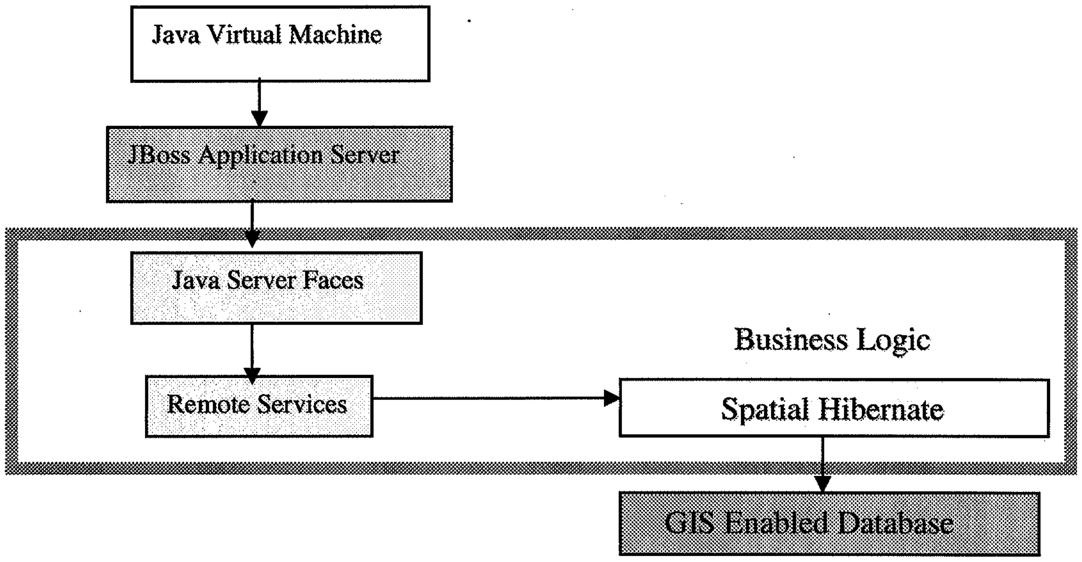


Figure 4: High level view of Web Application

We create an EJB3 application to find points that are within the specified proximity of a given location. The location is selected from the Blackbook's semantic data that is displayed on the Google Maps tab Remote method uses Hibernate to access the application data objects. The Data Objects are different geospatial datasets stored in MySQL. At the time of installing the application, the spatial datasets are inserted into Mysql database. Then, for each dataset in the MySQL database, a data object is created by the application. A stateless bean queries this dataset to find points of interests from this spatial dataset.

Detailed Architecture of the web application:

Figure 4 shows the flow of the messages in the web based GIS application. The inputs to the application are location address in latitude/longitude form; a WGS84 coordinate system and radius to specify proximity. Input comes from the Google Maps that display Blackbook's Semantic data (monetery, Anubis,etc). All the points within the radius (specified proximity) are known as Point Of Interest, POI. The Remote service known as POI manager finds the points of interest by querying the spatial data objects. POI remote interface accepts the request from clients for finding points of interest.

POI manager implements the remote interface. POI manager is a stateless session bean that does not maintain a conversational state for a particular client. When a client invokes the Remote method of a stateless bean, the bean's instance variables contains a state, but only for the duration of the invocation.

When the method is finished, the state is no longer retained. Stateless session bean can support multiple clients and provides scalability when the number of request increases.

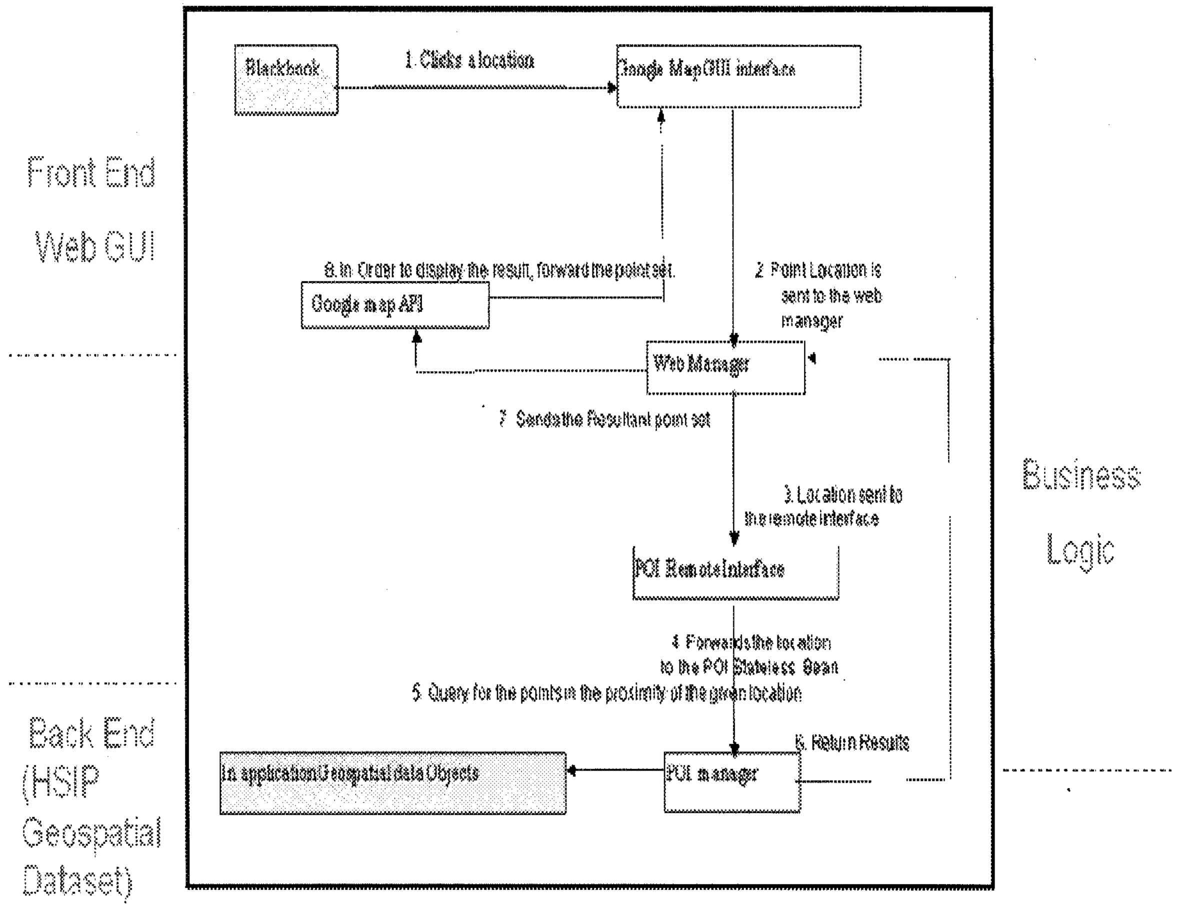

Front End
Web GUI
Business
Logic
Back End
(HSIP
Geospatial
Dataset)
Blackbook
1. Clicks a location
Google MapGUI interface
2. Point Location is sent to the web manager
8. In Order to display the result, forward the point set.
Google map API
Web Manager
7. Sends the Resultant point set
3. Location sent to the remote interface
POI Remote Interface
4. Forwards the location to the POI Stateless Bean
5. Query for the points in the proximity of the given location
6. Return Results
In application Geospatial data Objects
POI manager

Figure 5. Flow of Messages

Figure 5 explains the flow of messages in more detail. Online user marks a point on the two dimensional Google map. Google map interface reads the location in latitude/longitude form. Java server faces intercept any change in the location value at the client side. The java server faces gives the location of the point to the web manager at the client side. The web manager invokes the remote method to find proximate points of interest for the given location. The stateless session bean implements the invoked interface. It receives the location and radius. The resultant set consists of set of points that falls in the radius of the given location. The result is forwarded to the web manager that creates the Data Model which consists of the point of interest. Now, each point location is sent to the Google map API which displays the point as marker on the Geospatial Proximity Tab.

Following figure 6 shows blackbook's semantic data on the Google map. To find the Points of interest around any of those data, the user needs to click on the marker and provide the proximity radius.

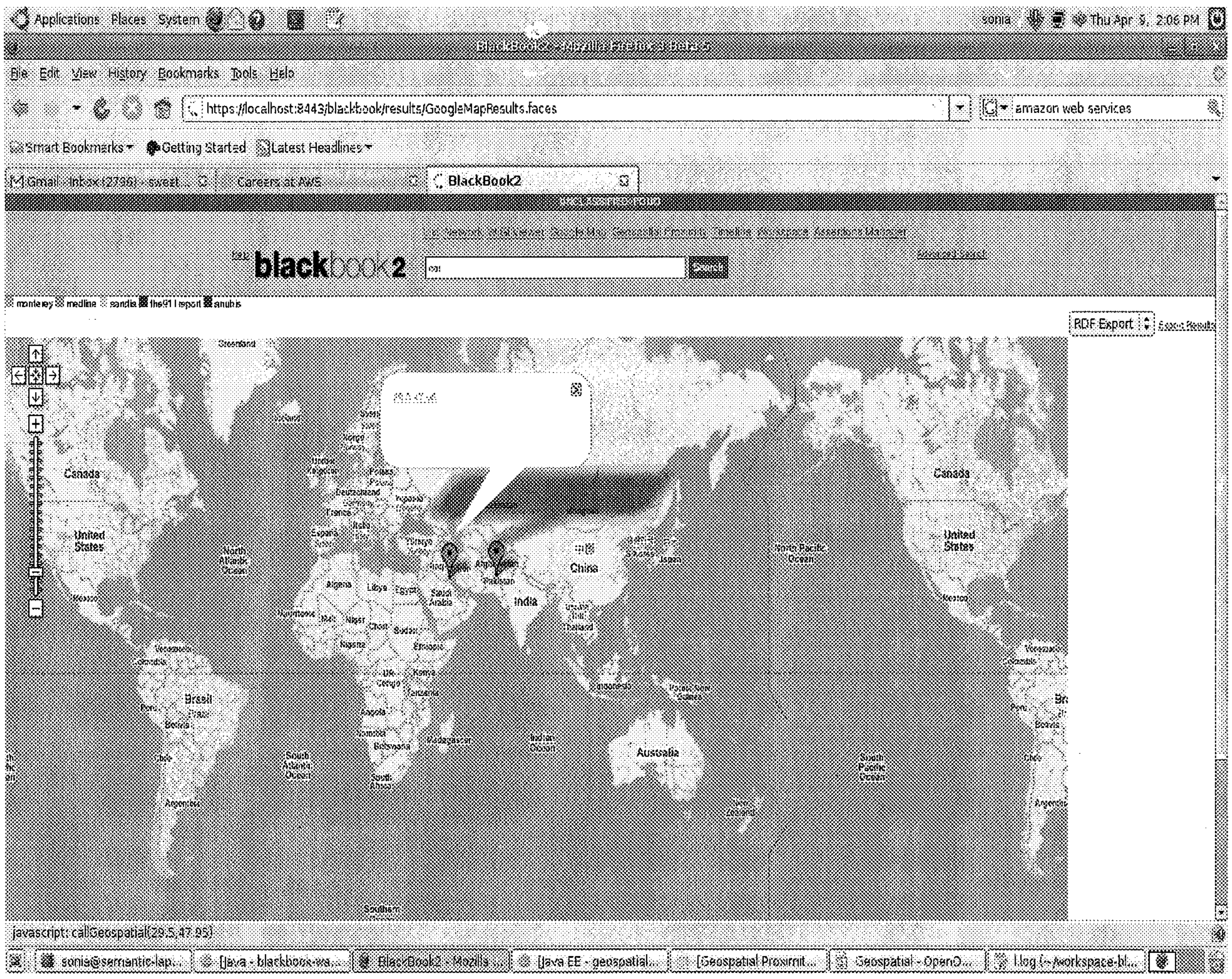
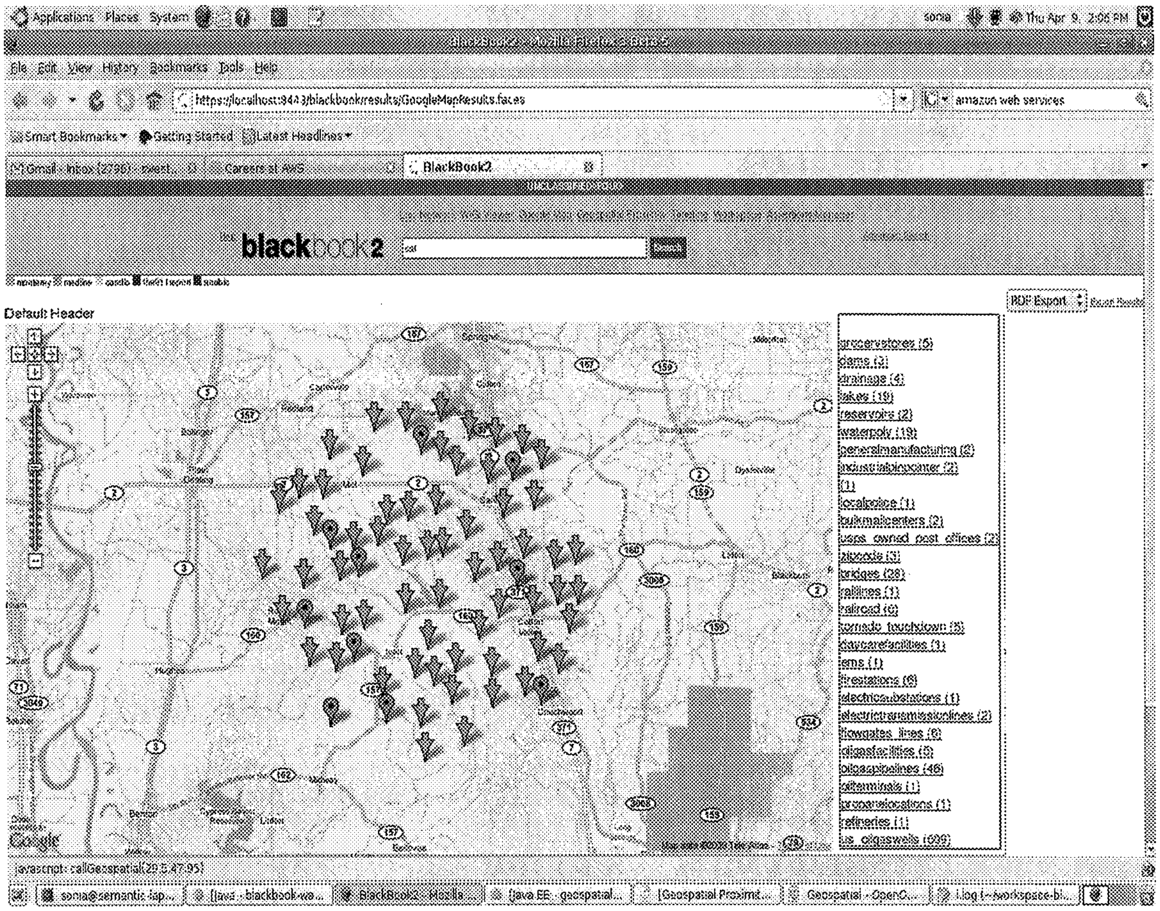

Applications Places System
Thu Apr 5, 2:06 PM
File Edit View History Bookmarks Tools Help
https://localhost:8443/blackbook/results/GoogleMapResults.faces
amazon web services
Smart Bookmarks Getting Started Latest Headlines
BlackBook2
blackbook2
RDF Export
javascript: callGeospatial(29.5,47.95)

Figure 6. Blackbook's Semantic Data

In figure 7 the resultant data is shown on the geospatial proximity tab. If the resultant data is huge, numbers of markers are clustered together into a single green marker. User can click on those and see individual locations. To provide more control to user, the Interface also categorizes the markers type as Grocerystores, dams, etc.


Applications Places System
File Edit View History Bookmarks Tools Help
https://localhost:8443/blackbook/results/GoogleMapResults.faces
amazon web services
Smart Bookmarks Getting Started Latest Headlines
BlackBook2
blackbook2
RDF Export
Default Header

grocerystores (5)
dams (3)
drainage (4)
lakes (19)
reservoirs (2)
waterpoly (19)
generalmanufacturing (3)
industrialpipcomer (2)
(1)
localpolice (1)
bulkmailcenters (2)
usps_owned_post_offices (2)
zipcode (3)
bridges (28)
raillines (1)
railroad (6)
tornado_touchdown (5)
daycarefacilities (1)
ems (1)
firestations (6)
electricsubstations (1)
electrictransmissionlines (2)
flowgates_lines (6)
oilgasfacilities (5)
oilgaspipelines (46)
oilterminals (1)
propanelocations (1)
refineries (1)
us_oilgaswells (609)

javascript: callGeospatial(29.5,47.95)

Figure 7. Resultant Data

Figure 8 displays the individual marker with information on its click.

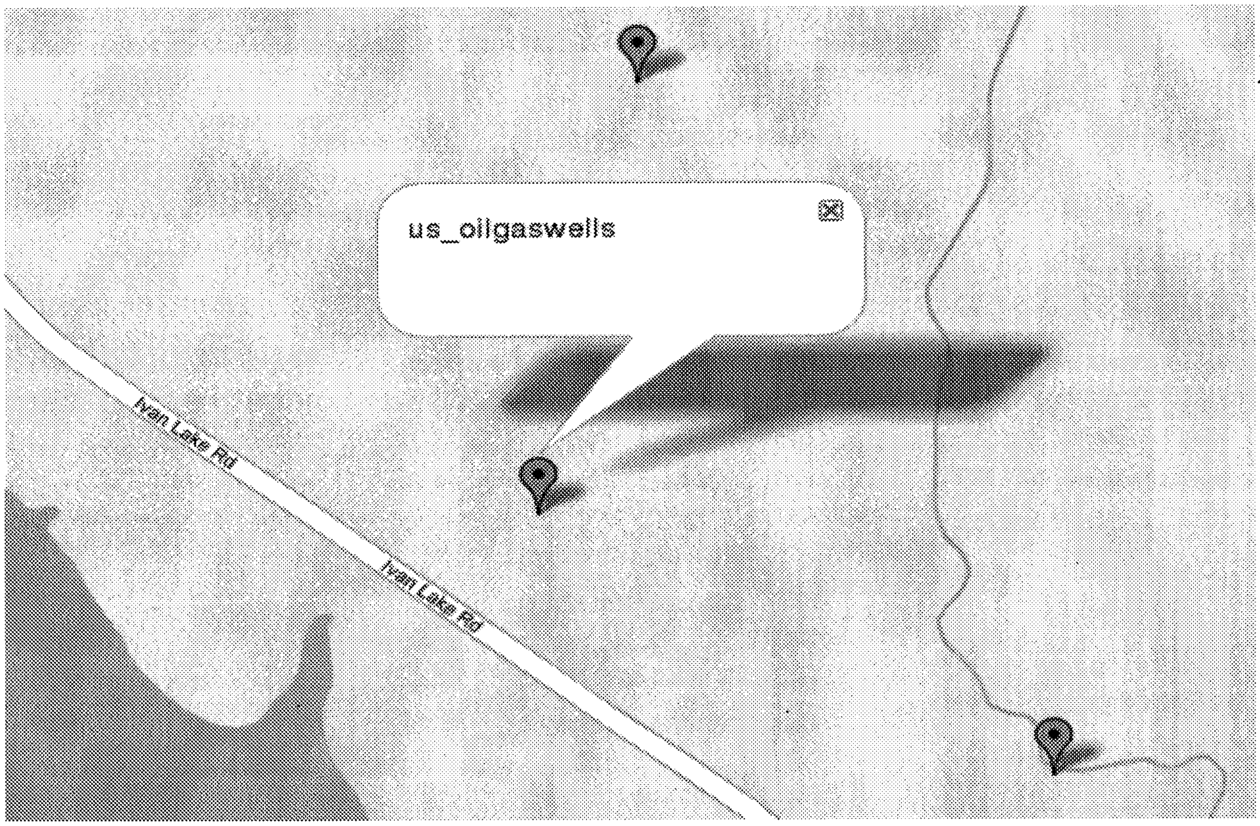


Figure 8 Individual marker

Geospatial project: Integration with Blackbook

1. mkdir workspace-geospatial
2. cd ~/workspace-geospatial
3. svn checkout http://workspace-geospatial.googlecode.com/svn/trunk/ .
4. cd ~/workspace-geospatial/config
5. mvn clean install
6. cd ~/workspace-geospatial/
7. mvn clean install
8. cd ~/workspace-blackbook/
9. mvn clean install

Install the project with geospatial data using following steps:

1. mkdir workspace-geospatial
2. cd ~/workspace-geospatial
3. svn checkout http://workspace-geospatial.googlecode.com/svn/trunk/ .
4. cd ~/workspace-geospatial/config-test-data
5. mvn clean install
6. cd ~/workspace-geospatial/config

7. mvn clean install
8. cd ~/workspace-geospatial/
9. mvn clean install
10. cd ~/workspace-blackbook/
11. mvn clean install

All these steps should happen before the Workspace-blackbook is clean installed.

6 Conclusions:

Geospatial proximity/data management helps in achieving the objective of Blackbook by providing easy-to-use Google Maps UI to access and analyze valuable data. The project enables Blackbook user to find Points of interest for an input location and display result on a Google map.

References

1. Aboulnaga, J. Naughton, Accurate estimation of the cost of spatial selections, in: Proceedings of the 16th IEEE International Conference on Data Engineering, San Diego, CA, 2000, pp. 123–134.
2. W. Aref, H. Samet, Optimization strategies for spatial query processing, in: Proceedings of the 17th International Conference on Very Large Data Bases, Barcelona, Spain, 1991, pp. 81–90.
3. T. Brinkhoff, H. Kriegel, R. Schneider, B. Seeger, Multi-step processing of spatial joins, in: Proceedings of the ACM SIGMOD Inernational Conference on Management of Data, Minneapolis, MN, 1994, pp. 237–246.
4. Markus Schneider, Thomas Behr, Topological Relationships Between Complex Spatial Objects
5. Nagapramod Mandagere, Buffer Operations in GIS in: University of Minnesota
6. Songtao Jiang a, Byung Suk Lee a, Zhen He b, Cost modeling of spatial operators using non-parametric regression q,in:
7. J2EE Tutorial, http://java.sun.com/j2ee/tutorial/1_3-fcs/index.html
8. Google Map Tutorial, http://code.google.com/apis/maps/documentation/introduction.html
9. Reference: http://www.safe.com/support/onlinelearning/documentation.php

ACKNOWLEDGEMENTS

We thank IARPA for their support. We also thank Scott Streit and Lance Byrd for their valuable inputs to this project.