GEOSPATIAL PROXIMITY

by

SONIA CHIB

THESIS
Presented to the Faculty of
The University of Texas at Dallas
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December, 2009
Today, there are many enterprise applications that provide an end user with an abundance of information in a simple text format, but for a good user experience some applications use maps to convey location based information. Geospatial Proximity is a web based GIS application integrated with BlackBook that queries geographic datasets and shows the results on a Google Map. This project was conceptualized to provide a better user interface to dedicated users of the BlackBook application. BlackBook is a semantically enabled research framework funded by IARPA: Intelligence Advanced Research Projects Activity.
I thank Dr. Bhavani Thuraisingham, Dr. Latifur Khan, Dr. Murat Kantarcioglu, Scott Streit and Lance Byrd for their extensive discussions and feedback of our work for IARPA.
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ........................................................................................................ iv

LIST OF FIGURES .................................................................................................................. vii

LIST OF TABLES ..................................................................................................................... viii

CHAPTER 1 INTRODUCTION ................................................................................................. 1

1.1 Blackbook ...................................................................................................................... 1
1.2 Problem Statement ......................................................................................................... 1
1.3 Project Objective ............................................................................................................ 1

CHAPTER 2 RELATED TOOLS AND TECHNOLOGIES ..................................................... 3

2.1 The Semantic Web Concept .......................................................................................... 3
2.2 Blackbook ...................................................................................................................... 6
2.3 Hibernate Spatial .......................................................................................................... 10
2.4 MySQL 5.1 .................................................................................................................... 11

CHAPTER 3 MAPPING SERVICES API’S ........................................................................... 12

3.1 Comparison of the Mapping Services ........................................................................... 12
3.2 Description for Google Map API .................................................................................. 13

CHAPTER 4 SPATIAL DATA AND DATA MIGRATION ....................................................... 27

4.1 Geospatial Data ............................................................................................................. 27
4.2 Spatial ETL tools .......................................................................................................... 28
4.3 Tools Used For Migration ............................................................................................ 28
4.4 Data Migration ............................................................................................................. 29

CHAPTER 5 GEOSPATIAL PROXIMITY ........................................................................... 33

5.1 Overview ...................................................................................................................... 33
5.2 Architectural Design .................................................................................................. 35
5.3 Detailed Architecture of the web application ........................................36
CHAPTER 6 GOOGLE MAP INTEGRATION WITH BLACKBOOK ..................42
CHAPTER 7 CHALLENGES .......................................................................47
CHAPTER 8 CONCLUSION AND FUTURE WORK ..................................50
CHAPTER 9 APPENDIX ..........................................................................52
REFERENCES .........................................................................................66
VITA
LIST OF FIGURES

Figure 1.1 Semantic Web Layers.................................................................4
Figure 2.1 Blackbook Architecture.............................................................10
Figure 2.2 Web services.............................................................................11
Figure 3.1 Pixel Coordinates.....................................................................17
Figure 3.2 Clustering..................................................................................23
Figure 4.1 Data Translation........................................................................30
Figure 5.1 High level view of Web Application...........................................35
Figure 5.2 Flow of Messages......................................................................37
Figure 5.3 Blackbook’s Semantic Data.........................................................38
Figure 5.4 Resultant Data...........................................................................39
Figure 5.5 Individual marker......................................................................40
LIST OF TABLES

Table 4.1 Migration Tools........................................................................................................29
CHAPTER 1
INTRODUCTION

1.1 Blackbook
A Semantic application uses data annotated with semantic tags and ontologies to infer new knowledge. BlackBook provides a semantically-enhanced search engine that returns more relevant results if both, a query and data both are annotated with semantic tags. BlackBook can be categorized as a semantic application since underlying ontologies (or a user's knowledge) are used to make knowledge based inferences that might otherwise not be made. BlackBook's MVC architecture provides support for visualization, transformation, data source integration, asynchronous operations, and is vocabulary agnostic. It is used as a research tool by researchers and data analysts to draw inferences from the semantic data.

1.2 Problem Statement
BlackBook is based on open source technologies. IARPA realized the need for Google Maps like visualizing tool to handle spatial information. Hence it was decided to implement an open source tool that can be easily adapted for different spatial information.

1.3 Project Objective
A large number of web applications need to integrate spatial information which is different from traditional information. Processing of, and queries for spatial information require special treatment than regular databases.
This project aims to provide a generalized architecture that can be easily adapted to integrate and manage any other spatial information.

The purpose of the project is to create a plug-n-play visualizer that can be integrated with a semantic application such as BlackBook. The project must find points of interest on the earth’s surface for a given place. The application should handle any type of spatial information that can be integrated from different data sources and is then ready for processing a query. The resulting points of interest must be displayed as markers on a Google Map. Finally, the map and the spatial information should be incorporated in BlackBook.

The rest of this thesis is organized as follows. Chapter 2 provides an overview of the tools and technologies used in the project. In Chapter 3, an overview of Mapping Services and the Google Map API is discussed in detail. In Chapter 4, we discuss spatial data and migration techniques used for Homeland Security Infrastructure Program (HSIP) data. Chapter 5 describes Geospatial proximity and further discusses the architecture and design used to achieve the project objectives. Chapter 6 addresses steps taken to integrate Google Maps with BlackBook. Chapter 7 focuses on the challenges faced during the implementation of the project. Chapter 8 describes future work that can be incorporated in project. The final section, chapter 9 concludes the thesis.
CHAPTER 2
RELATED TOOLS AND TECHNOLOGIES

2.1 The Semantic Web Concept

Overview

The Semantic Web provides a common framework that allows data to be shared and reused across applications, enterprises, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners.

The current web represents the information using natural languages, graphics and multimedia objects which can be easily understood and processed by an average user. Some tasks on the web require combining data on the web from different sources e.g hotel and travel information may come from different sites when making reservations for a trip. Humans can combine this information and process it quite easily. However, machines can’t combine such information and process it.

So we need to have the data that should be available to machines for further processing. Data should be possibly combined and merged on a Web scale. Data may describe other data and machines may also need to reason about that data. So we need a Web of data.

The vision of the semantic web is to extend the principles of the Web from documents to data. Data should be accessed using the general Web architecture by using for example URI’s. Data
should be related to one another like documents are. This also means creation of a common framework that allows data to be shared and reused across applications, enterprises and community boundaries, to be processed manually as well as automatically by tools, including revealing possibly new relationships among pieces of data.

**Architecture**

![Figure 1.1 Semantic Web Layers](image)

The Semantic Web principles [17] are implemented in the layers of Web technologies and standards. The Unicode and URI layers ensure that we use international character sets and provide means for identifying objects in the Semantic Web. With the XML layer with namespace and schema definitions, we can integrate the Semantic Web definitions with the other XML based standards. With RDF and RDF Schema, it is possible to make statements about objects with URI’s and define vocabularies that can be referred to by URI’s.
This is the layer where we can give types to resources and links. The Ontology layer supports the evolution of vocabularies as it can define relations between the different concepts. The Logic layer enables the writing of rules while the Proof layer executes the rules and evaluates together with the Trust layer mechanism for applications whether to trust the given proof or not.

**Application Areas**

Semantic Web can be used in a variety of application areas:

- Data Integration – whereby data in various locations and various formats can be integrated in one seamless application
- Resource Discovery and Classification – to provide better, domain specific search engine capabilities
- Cataloging – For describing the content and content relationships available at a particular web site, page or digital library
- By Intelligent Software Agents – to facilitate knowledge sharing and exchange
- Content Rating
- In describing collections of pages that represent a single “logical” document
- For describing intellectual property rights of Web pages and many others.

BlackBook supports integration of disparate data sources of many types (RDBMS’s, Documents, RSS) that are mapped into Blackbook2 as RDF/OWL, either by ingest or real-time mapping solutions such as D2RQ. BlackBook uses reasoning algorithms to derive the inferences from this data.
2.2 Blackbook

1) Overview

The main objective of the Blackbook [17] project is to improve intelligence analysis by coordinated exposition of multiple data sources across the intelligence agency community.

The Blackbook system is a JEE server-based [7] RDF processor that provides an asynchronous interface to back-end datasources. It is an integration framework based on semantic web technologies like RDF, RDF Schema, OWL and SPARQL. It relies on open standards like Jena, Jung, Lucene, JAAS, D2RQ etc to promote robustness and interoperability. Blackbook provides a default web application interface, SOAP interface and RESTful interface.

Blackbook connects several datasources

- 911 Report (Unstructured transform via NetOWL -> RDF)
- Monterey – Terrorist incidents (RDBMS -> RDF transform)
- Medline – Bio-data (XML -> RDF)
- Sandia – Terrorist profiles (RDBMS -> RDF transform)
- Anubis – Bio-equipment and bio-scientists (RDBMS -> RDF transform)
- Artemis – Bio-weapons proliferation (RDBMS via D2RQ)
- BACWORTH – DIA (web-services)
- Google-Maps – NGA (via Google-map API)
- CBRN Proliferation Hotlist – CIA (RDBMS -> RDF transform)
- Global Name Recognition service and 3 DBs - JIEDDO
- ICRaD Mediawiki w/ Semantic extension – CIA (dbPedia-like adapter)
- CPD Hercules – CIA (RDBMS via D2RQ)
2) **Objectives**

The purpose of Blackbook is to provide analysts with an easy-to-use tool to access valuable data. The tool federates queries across datasources. These datasources are databases or applications located either locally or remotely on the network. Blackbook allows analysts to make logical inferences across the datasources, add their own knowledge and share that knowledge with other analysts using the system.

3) **Business Functions**

   a) **Text Search**

   A user performs a text search against all available datasources. These datasources include those available through Web Services. Text searches search for matching values in the database. For example, if a text search is for “Smith,” the results may be for a person with the same last name or a street named “Smith Street”.

   The results from a text search bring back the URI of the RDF document.

   b) **Dip**

   Dips perform searches on user-specified datasources. These searches look for name-value pairs, so that a Dip for a person named “Smith” will not return a street named “Smith Street”.

   The Dip analogy is to take a value from a text search and “dip” that value into other datasources to see what will stick.
c) Materialize

Text searches also return the Uniform Resource Identifier (URI). These URIs return the source of the RDF document. The source may be a RDF or non-RDF document stored locally or in a remote location.

For example, a URI may point to a MS Word Document (.doc) stored in a database located across the network. The URI goes across the network as an HTTPS link. This allows an encrypted data exchange via SSL. The user’s web browser knows how to visualize the document returned based on its MIME type. In this case, the web browser will visualize the .doc file with MS Word.

4) Interfaces

d) Import Process

The import process allows an analyst to manipulate the OWL representation of an RDF document. Analysts build their own logical inferences through a user interface.

This interface also includes importing algorithms developed by researchers. These algorithms perform social network analysis. The algorithms run against the datasources as a batch process, without any analyst input. Text Search, Dip, Materialize the Text Search, Dip and Materialize interfaces are the Business functions of Blackbook. The description of these functions is in the previous topic.
e) **MIME type of RDF/XML**

The purpose of this interface is to plug-and-play open source visualizers. The system sends a RDF/XML document, with a MIME type of “RDF/XML,” back to the user’s web browser. The web browser will then know to visualize the RDF/XML document. If the web browser does not know what to do with the RDF/XML document, it asks the user to download it as a file.

f) **Business Process Execution Language (BPEL)**

BPEL lets the user build a sophisticated query for the workflow of the Text Search and Dips. Using BPEL the user may specify the search order of datasources.

g) **Web Services**

Blackbook uses Web Services [7],[18],[19] to automate the data exchange mechanism with any capable enterprise application belonging to organizational partners. Other technologies, such as RMI or JMS, are capable in building the data exchange mechanism. However, Web Services gives three features other technologies do not provide:

1. Two-way SSL
2. Use of the Web protocol
3. No dependency on JEE server implementation
2.3 Hibernate Spatial

Hibernate Spatial is a generic extension to Hibernate [14] for handling geographic data. Hibernate Spatial is open source and licensed, like Hibernate, under the LGPL license.

Hibernate Spatial allows you to deal with geographic data in a standardized way. It abstracts away from the specific way your database supports geographic data, and provides a standardized, cross-database interface to geographic data storage and query functions.


Supported databases are: Oracle 10g/11g, Postgresql/Postgis, and MySQL.
2.4 MySQL 5.1

The MySQL™ [16] software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software. MySQL is a registered trademark of Sun Microsystems, Inc.
CHAPTER 3
MAPPING SERVICES API’S

An online mapping service [20], [21] allows mapping spatial data quickly onto Maps. Mapping services are used to develop mapping solutions which in traditional web applications to handle spatial information. There are three Web Mapping Services available free for non commercial use and, using Google Maps APIs is easier as it is highly documented and available on the web.

3.1 Comparison of the Mapping Services

Google Maps API

It has a fluid and cleaner interface that is easy to use and more appealing to an end user. The accuracy and precision of the marked location is much better than others. It provides international support for non USA locations. It has built in Aerial that is much clearer than others. It has smoother AJAX client easy to implement. It has simple and easy to use APIs. It has the largest developer base, as a result lots of documentation is available.

It suffers from two major drawbacks, no built-in Geocoding service and Routing capability which are as such not requirements for our project.
Yahoo Maps API

It has both built-in and external Geocoding capability. The API’s are open source and flexible to change. The number of requests is limited by IP address instead of appID which is another drawback of the Google Map API. It has built-in GeoRSS support and provides a flash version as well.

It has a major disadvantage in that it supports only the U.S. and Canada maps. Also, the fly outs are not quite as neat as Google. It does not have an aerial photo option and no reverse Geocoding is provided.

MapQuest API

MapQuest has excellent built-in routing (driving directions) capability and built-in Geocoding capability but it does not have a great amount of documentation support. Neither does it have a smooth AJAX client nor does it provide reverse Geocoding and the rate is limited by appID + web site URL (instead of end-user IP), the same as Google. Also, it does not have a neat interface or a photos option as Google

3.2 Description for Google Map API

Google Maps [8],[21],[20] is a web mapping service application and technology provided by Google, free for use that powers many map-based services, and maps embedded on third-party
websites via the Google Maps API. It offers street maps, a route planner for traveling by foot, car, or public transport and an urban business locator for numerous countries around the world. According to one of its creators (Lars Rasmussen), Google Maps is "a way of organizing the world's information geographically".

Google created the **Google Maps API** to allow developers to integrate Google Maps into their websites with their own data points. By using the Google Maps API, it is possible to embed the full Google Maps site into an external website. Developers are required to request an API key [13], which is bound to the website and directory entered when creating the key.

**Using Google Maps API**

The easiest way to start learning about the Google Maps API is to see a simple example. The following web page displays a 500x300 map centered on Palo Alto, California:

```html
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8"/>
<title>Google Maps JavaScript API Example</title>
<script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=abcdefg&amp;sensor=true_or_falso" type="text/javascript">
</script>
<script type="text/javascript">
</script>
</head>
```
function initialize() {

    if (GBrowserIsCompatible()) {

        var map = new GMap2(document.getElementById("map_canvas"));
        map.setCenter(new GLatLng(37.4419, -122.1419), 13);
        map.setUIToDefault();
    }
}

</script>
</head>
<body onload="initialize()" onunload="GUnload()">

    <div id="map_canvas" style="width: 500px; height: 300px"></div>
</body>
</html>

You can look at this example and play around with it, but you'll have to replace the key in that file with your own Maps API key. (If you register a key for a particular directory, it works for all subdirectories as well.)

In this simple example, there are five things to note:

- We include the Maps API JavaScript using a script tag.
- We create a div element named "map_canvas" to hold the Map.
- We write a JavaScript function to create a "map" object.
- We center the map on a given geographic point and set the UI to a default configuration.
- We initialize the map object from the body tag’s onLoad event.

Note:

```html
<script src='http://maps.google.com/maps?file=api&v=2&amp;key=abcdefg&amp;sensor=true_or_false'
type='text/javascript'>
</script>
</script>
```

The `http://maps.google.com/maps?file=api&v=2&amp;key=abcdefg` URL points to the location of the JavaScript file that includes all of the symbols and definitions you need for using the Google Maps API. Your page must contain a script tag pointing to this URL, using the key you received when you signed up for the API. In this example the key is shown as "abcdefg."

Once we have a map, we need a way to refer to locations on the map. The GLatLng object provides such a mechanism within the Google Maps API. You construct a GLatLng object, passing its parameters in the order `{latitude, longitude}` as is customary in cartography:

```javascript
var myGeographicCoordinates = new GLatLng(myLatitude, myLongitude)
```

GLatLng objects have many uses within the Google Maps API. The GMarker object takes a GLatLng in its constructor, for example, and places a marker overlay on the map at the given geographic location.

**Google Maps Coordinates**

There are three coordinate systems [4], [11], [12] that the Google Maps API uses:

- Pixel coordinates referencing a point on an image tile
Tile coordinates referencing a tile within a tile layer

The zoom layer, which defines the total number of tiles

We have worked with the Pixel coordinate system which is discussed below.

**Pixel Coordinates**

Each tile within Google Maps consists of 256 x 256 pixels. A point on a particular tile can therefore be referenced using a GPoint \(x,y\) pair. The origin \((0,0)\) for each tile is denoted at the northwest corner of the tile. Therefore, for the single tile which represents the entire earth, the origin is set to lie at the North Pole and -180 degrees longitude, where you see Alaska. \(x\) (longitude) values increase east, while \(y\) (latitude) values increase south to the southeast point \((255,255)\).

![Figure 3.1 Pixel Cordinates](image)

At higher zoom levels, the pixel space expands by doubling both in the \(x\) and \(y\) directions. For example, at zoom level 1, the map consists of 4 256x256 pixels tiles, resulting in a pixel space of...
512x512. At zoom level 19, each \( x \) and \( y \) pixel on the map can be referenced using a value between 0 and \( 256 \times 2^{19} \).

- Class GMap2 is the core class:
  
  We instantiate class GMap2 in order to create a map. This is the central class in the API. Everything else is auxiliary.

- class GMarker
  
  - A GMarker marks a position on the map. It implements the GOverlay interface and thus is added to the map using the GMap2.addOverlay () method.
  
  - A marker object has a latlng, which is the geographical position where the marker is anchored on the map, and an icon. If the icon is not set in the constructor, the default icon G_DEFAULT_ICON is used.
  
  - After it is added to a map, the info window of that map can be opened through the marker. The marker object will fire mouse events and info window events.

**Map Overlays:**

Overlays are objects on the map that are tied to latitude/longitude coordinates, so they move when you drag or zoom the map. Overlays reflect objects that you "add" to the map to designate points, lines, or areas.

The Map always forms the first layer of display and everything else is placed on top of that layer which basically forms the Overlays.
The Maps API has several types of overlays:

- Points on the map are displayed using **markers**, and often display a custom icon.
- Lines on the map are displayed using **polylines** (representing a collection of points).
- Areas on the map are displayed either as **polygons** if they are areas of an arbitrary shape or as **ground overlays** if they are rectangular. Polygons are similar to polylines in that they consist of a collection of points with a closed loop and may take any shape. Ground overlays are often used for areas that map either directly or indirectly to tiles on the map.
- The map itself is displayed using a **tile overlay**. You can modify this with your own set of tiles by using a GTileLayerOverlay or even by creating your map type using a GMapType.
- The **info window** is also a special kind of overlay. It is used to provide the information about the clicked overlay. Each overlay or marker can have individual Info windows for giving detailed information or hyper links to other sites.

  Each overlay implements the GOverlay interface. Overlays can be added to a map using the GMap2.addOverlay() method and removed using the GMap2.removeOverlay() method.

**Markers:**

Markers are the most commonly used overlay to display positional landmarks on the Map. Markers identify points on the map. By default, they use use G_DEFAULT_ICON, though you can specify a custom icon. The GMarker constructor takes a GLatLng and an optional GMarkerOptions objects as arguments.
Markers are interactive; we can attach listener to events on them and specify the desired response. By default, they receive "click" events, for example, and are often used within event listeners to bring up info windows.

**Icons**

Markers are images to point a location on the map. Those images are known as icons. Markers may define an icon to show in place of the default icon. Defining an icon is complex because of the number of different images that make up a single icon in the Maps API. At a minimum, an icon must define the foreground image, the size of type GSize, and an icon offset to position the icon.

The simplest icons are based on the G_DEFAULT_ICON type. Creating an icon based on this type allows you to quickly change the default icon by modifying only a few properties.

**Using the Marker Manager**

Adding a large number of markers to a Google map slows down rendering of the map and introduces too much visual clutter, especially at certain zoom levels. The marker manager utility provides a solution to both of these issues, allowing efficient display of hundreds of markers on the same map and the ability to specify at which zoom levels markers should appear.

The marker manager utility is provided within the GMaps Utility Library. This library is open source, and contains utilities that are not part of the core Google Maps API. To add utilities contained within this library, you add the JavaScript source directly with a <script> tag.
The MarkerManager object within the markermanager.js library offloads management of markers registered with the utility, keeping track of which markers are visible at certain zoom levels within the current view, and passing only these markers to the map for drawing purposes. The manager monitors the map's current viewport and zoom level, dynamically adding or removing markers from the map as they become active. In addition, by allowing markers to specify the zoom levels at which they display themselves, developers can implement marker clustering. Such management can greatly speed up map rendering and reduce visual clutter.

To use a marker manager, create a MarkerManager object. In the simplest case, just pass a map to it.

```javascript
var map = new GMap2(document.getElementById("map_canvas"));
map.setCenter(new GLatLng(41, -98), 4);
var mgr = new MarkerManager(map);
```

You may also specify a number of options to fine-tune the marker manager's performance. These options are passed via a MarkerManagerOptions object, which contains the following fields:

- `maxZoom`: specifies the maximum zoom level monitored by this marker manager. The default value is the highest zoom level supported by Google maps.
- **borderPadding**: specifies the extra padding, in pixels, monitored by the manager outside the current viewport. This allows for markers just out of sight to be displayed on the map, improving panning over small ranges. The default value is 100.

- **trackMarkers**: specifies whether movement of movements of markers should be tracked by the marker manager. If you wish to have managed markers that change their positions through the `setPoint()` method, set this value to true. By default, this flag is set to false. Note that if you move markers with this value set to false, they will appear in both the original location and the new location(s).

Once you create a manager, you will want to add markers to it. `MarkerManager` supports adding single markers one at a time using the `addMarker()` method or a collection passed as an array using the `addMarkers()` method. Single markers added using `addMarker()` will appear immediately on the map provided that they fall within the current view and specified zoom level constraints.

Adding markers collectively using `addMarkers()` is recommended as it is more efficient. Markers added using the `addMarkers()` method will not appear on the map until you explicitly call the `MarkerManager's refresh()` method, which adds all markers within the current viewport and border padding region to the map. After this initial display, `MarkerManager` takes care of all visual updates by monitoring the map's "moveend" events.
**ClusterMarker**

ClusterMarker is a marker manager for use with maps powered by the Google Maps API.

ClusterMarker detects any group(s) of two or more markers whose icons visually intersect when displayed. Each group of intersecting markers is then replaced with a single *cluster marker*. The cluster marker, when clicked, simply centers and zooms the map in on the markers whose icons intersect. Alternatively you can write your own function to be executed when a cluster marker is clicked.

![Normal vs ClusterMarker](image)

**Figure 3.2 Clustering**

ClusterMarker provides added features and hence we decided to use this enhanced manager. It made a huge leap in the performance of the project.
This library is open source, and contains utilities that are not part of the core Google Maps API. To add utilities contained within this library, you download and add the JavaScript source directly with a <script> tag.

<script type="text/javascript"
src="maps-clustermarker/ClusterMarker/ClusterMarker.js"></script>
<script type="text/javascript" src="maps-clustermarker/HtmlControl/HtmlControl.js"></script>

API References:

3.2.1 Constructor: var myCluster = new ClusterMarker(map, options);

- **map** (GMap2) is a required argument. This must be a reference to the map which you want to add your Clustermarker to.

- **options** is an **optional** argument, passed to the ClusterMarker constructor as an object literal.

  Properties that can be set using the options object literal are:

  - **borderPadding** (integer).
  - **clusteringEnabled** (boolean).
  - **clusterMarkerClick** (function).
  - **clusterMarkerIcon** (GIcon) Pass a GIcon to the constructor to change the default appearance of the cluster marker (the green arrow marker).
  - **clusterMarkerTitle** (string) Customise the tooltip that appears when the mouse is over a cluster marker.
- **fitMapMaxZoom** (integer).
- **fitMapToMarkers** (boolean) Now removed from the script and replaced with a new method `fitMapToMarkers()` that performs the same task.
- **intersectPadding** (integer).
- **markers** (array of markers) This is the collection of markers that ClusterMarker will manage.

### 3.2.2 Methods

- **addMarkers** (array of markers) adds the array of markers to ClusterMarker to manage.
  
  If some markers have already been added to your ClusterMarker then the new markers will be added to the existing markers.

  ```javascript
  myCluster.addMarkers(markers);
  ```

- **fitMapToMarkers()** zooms and moves the map so that all markers (if not clustered or hidden) would fit the map view.

  ```javascript
  myCluster.fitMapToMarkers();
  ```

- **removeMarkers()** simply removes all managed markers from ClusterMarker. After calling removeMarkers() you can then use the addMarkers() method to manage another (new) collection of markers.

  ```javascript
  myCluster.removeMarkers();
  ```
• **refresh(boolean)** causes ClusterMarker to re-calculate all markers' visibilities and update the map.

You can optionally pass a boolean value such as refresh (true); to force ClusterMarker to perform a complete update rather than it's default optimized update.

The default update is used when you call either refresh() or refresh(false) and is optimized for speed - if you find clusters not being updated then use the refresh(true); option.

myCluster.refresh();

• **triggerClick(integer)** triggers a click event on the marker identified in the array of markers by the index passed to triggerClick.

The map will move and/or zoom in if necessary to uncluster that marker.

Currently this method can be slow if the map has to zoom many levels to uncluster the chosen marker.

myCluster.triggerClick(8)
CHAPTER 4
SPATIAL DATA AND DATA MIGRATION

4.1 Geospatial Data

Spatial data [1],[2],[3] indicates the location of geographic features on the Earth. 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) [4],[16].

*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 the 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.

**4.2 Spatial ETL tools**

There 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 functions read data from a specified data source and extract 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 organizations with an opportunity to improve productivity and take better advantage of the data that they already have.

**4.3 Tools Used For Migration**

There are several open sources 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 Table 4.1.
Table 4.1 Migration Tools

<table>
<thead>
<tr>
<th>Source Format</th>
<th>Tool Used</th>
<th>Destination Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESRI Personal GeoDatabase</td>
<td>ArcGis (Arc Toolbox-conversion tools)</td>
<td>ESRI Shapefile</td>
</tr>
<tr>
<td>ESRI Shapefile</td>
<td>VB Script</td>
<td>ESRI Personal GeoDatabase</td>
</tr>
<tr>
<td>ESRI Personal GeoDatabase</td>
<td>OGR2OGR,FME safe software</td>
<td>MySQL</td>
</tr>
<tr>
<td>ESRI Shapefile</td>
<td>shp2mysql</td>
<td>MySQL</td>
</tr>
<tr>
<td>ESRI SDC</td>
<td>ArcGis conversion tools</td>
<td>ESRI Shapefile</td>
</tr>
</tbody>
</table>

*FME* is a complete spatial ETL solution that enables GIS Professionals to quickly translate, transform, integrate and distribute spatial data.

- Convert and integrate data in 225+ formats
- Manipulate data into the exact data model you need
- Share spatial data and ETL tasks over the web

4.4 Data Migration

Majority of the given spatial data 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.

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 that 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.

![Figure 4.1 Data Translation](image-url)

Figure 4.1 Data Translation
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. A buffer query that uses a 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 a 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.

As most of the other projects used MySQL in BlackBook, to maintain consistency we use MySQL by simplifying geometry as a Double data type.

Entire MS Access data was converted to MySQL Scripts which was later integrated with the project.

_Procedure for data conversion using FME safe software [9]:_

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.
CHAPTER 5
GEOSPATIAL PROXIMITY

5.1 Overview

Let us start with a scenario where the Blackbook application is being used by a researcher. We know Blackbook provides a search utility on the first screen. The user will enter a search string in the textbox and hit the search icon. BlackBook uses various reasoning algorithms to query the integrated data sources (which has been converted to RDF). As the application is used by intelligence agencies, the search result consists of people, organization and incidents that have connection with the search string. Blackbook is based on MVC architecture. The entire result data known as nodes is “modeled” into in-memory data container. All the visualizer uses the same in-memory data model to show the result. Initially, the result is “viewed” in tabular list format. A user can select a different visualizer such as networked that gives a Wired view that displays the level of association among the nodes and search string.

All these visualizers were text based displaying only the title for each node and used RDF schema to show detailed information. It was decided that to improve usability Blackbook should have one more specific visualizer that would cater to location specific needs. There was a requirement to display the location for the node on the map.
The result nodes consisting of people, organization or incidents were already present in memory model. The task was to read location from that in-memory data model and place them on the Google Map. Those are known as markers in terms of Google maps. Each marker was clickable; the user can click any marker to find its nearby landmarks or points of interest.

This visualizer was thought in lieu to help the security official to catch someone’s suspicious lies. A security clearance officer can query a passenger about his home residence. The officer can type in the address specified in passport and the location is displayed on the Google map as a marker. The officer can query the passenger for any nearby landmarks he can think of. Simultaneously, he can pull up all the nearby landmarks around that location using the Geospatial Proximity application. The officer can catch if the passenger is genuine or suspicious.

In technical terms, the Geospatial Proximity project can be summarized as:

Given a location on map which is represented by latitude/longitude, find the points of interest within the proximate distance.

When the Blackbook search generates the result nodes, it can be visualized on the Google map as markers. When the proximity distance is provided for any marker, the points of interest for that marker are displayed with the feature menu. The Feature menu helps the user to select and view only those points or landmarks that falls under the selected feature.
5.2 Architectural Design

![Diagram of architectural design](image)

**Figure 5.1 High level view of Web Application**

Firstly, the Blackbook result has to be plotted on a new visualizer Google Map. Then all geospatial data has to be migrated to a MySQL format. This geospatial data consisted of Vector and Raster data sources; which are proprietary read only by ARCGIS. The next step was to write code that uses these spatial datasets for locating points of interest within the specified proximity. Later, we will analyze the response time for finding the points of interest.

We create an enterprise application to find points that are within the specified proximity of a given location. The location is clicked 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 interest from this spatial dataset.

5.3 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 (Monetary, 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 beans can support multiple clients and provide scalability when the number of request increases.
Figure 5.2 Flow of Messages

Figure 5.2 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 a set of points that fall 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 a marker on the Geospatial Proximity Tab.

Figure 5.3 Blackbook’s Semantic Data
Figure 5.3 shows Blackbook’s semantic data on a Google map. To find the points of interest around any of this data, the user needs to click on the marker and provide the proximity radius.

In Figure 5.4 the resultant data is shown on the geospatial proximity tab. If the resultant data is huge, markers are clustered together into a single green marker. Users can click on those markers and see individual locations. To provide more control to a user, the interface also categorizes the markers type as grocery stores, dams, etc.
Figure 5.5 displays the individual marker with information on its click.

![Figure 5.5 Individual marker](image)

**Geospatial project: Integration with Blackbook**

1. mkdir workspace-geospatial
2. cd ~/workspace-geospatial
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`


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.
CHAPTER 6
GOOGLE MAP INTEGRATION WITH BLACKBOOK

Google Maps is the most popular and efficient 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.

Brief outline of the integration steps:

- The geospatial data can be stored in the MySQL database. EJB 3.0 has an in-built Hibernate component.
- Access the geospatial data using Hibernate SQL.
- Once we fetch the points of interest from the MySQL database, the 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 map 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 an html client we can pass those points 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 the API, markers will be 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 is given below. The following changes are made to configuration 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’s website to sign up for Google Key.
- 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]]></CDATA>[v=2
    &amp;]]></CDATA]>amp;<![CDATA[key=ABQIAAAA1wvIiGmoHRIHUF5ekOWIMkxT_D
    2nltWVOjMhMSrkFYSsFd04bKxQiBsC29glgzh9hLbjgu7Dhe6KO4g]]>
  </default-google-key>
</properties>
</profile>

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

*pom.xml*

  o Gkey value is extracted 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):

  <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}" />
  </filterset>
<!-- Google Key from the settings.xml -->

</filterset>

*faces-bean.xml.template

  o  Gkey 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>
1) Understanding geographic data:

It required GIS knowledge as a prerequisite to deal with spatial data. The most important column in the given tables was the geometry column, which is hidden from the user. In order to efficiently retrieve the geometry from the “geom” column many open source and propriety tools were experimented.

Also, at the time of migration of geometry data from ArcGIS file formats into MySQL the conversion processes returned errors. This problem was fixed by increasing the maximum packet size to 64 MB in MySQL’s configuration file.

2) Coordinate System:

MySQL follows a planar XY coordinate system and to perform any spatial operations it considers planar XY coordinates for in-built spatial functions. It does not support any other coordinate system. The converted spatial data used Geocentric Coordinate system that considers the curvature of the earth. We had to calculate the proximate points considering the curvatures of the earth and hence we were not able to use the in-built spatial functions. We used a special distance formula to estimate the proximity between two earth points by using each latitude and longitude.
3) Dependency between BlackBook and Geospatial Proximity:

The original data has 35 databases and around 330 relations. Once the data was migrated to SQL scripts, it was integrated with the Geospatial proximity project. Once done, it was uploaded to the repository which took 2 hours on an average for uploading. But the downloading always fails because of the *humongous* size of the data files.

After integration with BlackBook, BlackBook was dependent on Geospatial proximity. A quick download was of utmost importance and with such huge data; BlackBook’s download time was shooting up. It was agreed to split the databases into individual files for each relation. For the developer's purpose only a few data files were uploaded along with the project at once.

Modifying the setting for the geospatial proximity, downloading spatial data was made into a separate activity. The user was able to download the Blackbook with geospatial proximity, but without spatial data. Separate instructions were provided on how to download the geospatial data.

4) The configuration Plug-in: Maven SQL Plug-in

The plug-in is used to read the spatial data from the SQL script and create the corresponding relations in a MySQL database. As the SQL Script was read through this plug-in, certain characters were not fed to the MySQL Server in the correct format. A rigorous script was written which reads entire datasets and corrects those erroneous characters.
5) Clustering of the markers:

Markers in the result were displayed on a Google Map screen. Sometimes the result was so large, that it took a couple of seconds before the entire result was plotted on the screen. The visual on screen looked cumbersome and unclean. Along with performance, the look and feel was less than desired the benchmark.

The Google Map API provides a cluster manager [10] feature which handles a large set of markers at a lower zoom level. As the zoom level increases, the markers are disintegrated and displayed as individual markers. Displaying the cluster marker at lower zoom improved the performance as well as provided a clean look and feel to the end user.

There were still many markers that belonged to different feature categories and were displayed at a higher zoom level even when a user might not be interested in all of them. So a feature control list was provided to the user so that the user can view markers that fall under the selected feature.
CHAPTER 8
CONCLUSION AND FUTURE WORK

Geospatial proximity achieves the objective of Blackbook by providing easy-to-use Google Maps to access and analyze valuable data. The project enables a Blackbook user to find points of interest for an input location and displays results on a Google map.

Following gives the scope for future work.

- Rehost repository:
  - The spatial data has 350 relations and more than 40 million records. If all the data is uploaded to RABA Server, the uploading time is high. As a result, the server slows down the download for other users.
  - Currently, the data is residing on the Google repository, this separates the data download from its project download.

- Decrease the download time.
  - The response time to download geospatial dataset is high and it increases the downtime for developers.

- Decouple from BlackBook webapp.
  - To reduce the download time for BlackBook developers when downloading the application, we can separate the geospatial data from main code of Blackbook.
• Implementation with Lucene
  o We can convert relational spatial data into Lucene structures to make it search faster for proximate points.
CHAPTER 9
APPENDIX

Setting.xml: It is the Configuration file to set up the environment for the Geospatial project.

```
<settings>
   <!--
   NOTE: Even when installing on a MS-Windows machine, make sure to always use forward slashes ('/') in any path that is defined in this file. Example of an alternative Windows repository:
   <localRepository>C:/blackbook/.m2/repository</localRepository>
   -->
   <!-- uncomment and modify the <localRepository> property if you want to modify the location of the maven repository. Make sure to use a full path name with no variables or properties. When this is accessed from within an ant task, use the property name of "$\{settings.localRepository\}"
   If you want to change the location of this settings.xml file then do so and name it on the maven command line with the -s option.
   <localRepository>[your home directory]/.m2/repository</localRepository>
   -->
   <profiles>
      <profile>
         <id>overrides</id>
         <properties>
            <!-- Modify the next line to provide the location of JBoss on your system, if needed -->
            <jboss.home>${user.home}/pkg/jboss</jboss.home>
```

<!-- uncomment and modify these properties to provide non-default values -->
<.datasources.list>default defaultLocal medline monterey the911Report</datasources.list>
<lucene.dir>${user.home}/workspace-blackbook-data/data1</lucene.dir>
<testdata.dir>${user.home}/workspace-blackbook-data/testData</testdata.dir>
<max.heap>512m</max.heap>
<max.permgen>256m</max.permgen>
<source.keystore.file>${resources.dir}/jboss/server/default/conf/server.keystore</source.keystore.file>
<source.truststore.file>${resources.dir}/jboss/server/default/conf/trusted.keystore</source.truststore.file>
<!-- Uncomment these other.javaopts in order to enable remote debugging of jboss. You must reconfigure jboss by building in workspace-blackbook/config for this to take effect. -->
<other.javaopts>-Xdebug -Xrunjdwp:transport=dt_socket,address=8787,server=y,suspend=n</other.javaopts>

<!-- Classification properties -->

<!-- A comma-separated list of roles of each type. Note that these roles must actually be assigned to the user in order for them to be visible in Blackbook. The order must be lowest to highest classification. -->
<classification>UNCLASSIFIED,CONFIDENTIAL,SECRET,TOP SECRET</classification>
<compartment>SI</compartment>
<releasability>USA</releasability>

<!-- Classification banner settings: -->
<!-- classification.testing is true if a caveat should appear on screen that the classification markings are for testing purposes only, and not truly classified. Otherwise, the property can be omitted or set to false. -->
<classification.testing>false</classification.testing>

<!-- Classification banner to appear on screen (background color of banner will be set automatically) -->
<classification.banner>UNCLASSIFIED//FOUO</classification.banner>

<!-- Color mappings for link classifications. The order is important - must be highest to lowest classification. The role "Multiple" doesn't actually exist as a role, but is used to indicate a link with multiple role levels. -->
<roles.colormap>TOPSECRET=FF6600,SECRET=FF0000,CONFIDENTIAL=0000FF,UNCLASSIFIED=009900,Multiple=808080</roles.colormap>

<!-- Automated ingest settings: -->

</properties>
</profile>
</profiles>

<activeProfiles>
<activeProfile>overrides</activeProfile>
<activeProfile>cert</activeProfile>
<activeProfile>blackbook-devel-configuration</activeProfile>
<activeProfile>security-devel-configuration</activeProfile>
<activeProfile>workflow-devel-configuration</activeProfile>
<activeProfile>workspace-devel-configuration</activeProfile>
<activeProfile>geospatial-devel-configuration</activeProfile>
<activeProfile>include-blackbook-applet</activeProfile>
<activeProfile>deploy-blackbook-ear</activeProfile>

<!--
<activeProfile>deploy-workflow-ear</activeProfile>
<activeProfile>run-functional-tests</activeProfile>
<activeProfile>run-jboss</activeProfile>
-->

</activeProfiles>
<pluginGroups>
  <pluginGroup>blackbook</pluginGroup>
</pluginGroups>
</settings>

**Pom.xml in Geospatial Proximity:** It sets up the environment for the project.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>geospatial</groupId>
  <version>1.0</version>
  <artifactId>app</artifactId>
  <name>super pom</name>
  <packaging>pom</packaging>
  <repositories>
    <repository>
      <id>geospatial</id>
      <name>Breit Maven Repository</name>
      <layout>default</layout>
      <url>http://www.thestreits.com/maven</url>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
    </repository>
  </repositories>
</project>
```
<name>Maven Repository Switchboard</name>
<layout>default</layout>
<url>http://repol.maven.org/maven2</url>
<snapshots>
  <enabled>false</enabled>
</snapshots>
</repository>

<repository>
  <id>jboss</id>
  <name>JBoss Repository</name>
  <layout>default</layout>
  <url>http://repository.jboss.com/maven2/</url>
  <snapshots>
    <enabled>false</enabled>
  </snapshots>
</repository>
</repositories>

<pluginRepositories>

</pluginRepositories>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.5</source>
        <target>1.5</target>
      </configuration>
    </plugin>
  </plugins>
</build>

<profiles>
  <profile>
    <id>geospatial-devel-configuration</id>
    <modules>
      <module>geospatial-ejb</module>
      <module>geospatial-war</module>
      <module>geospatial-ear</module>
    </modules>
  </profile>
</profiles>
<id>functional-test</id>
<activation>
  <property>
    <name>enableCiProfile</name>
    <value>true</value>
  </property>
</activation>
<modules>
  <module>geospatial-functional-tests</module>
</modules>
</profile>
</profiles>

<reporting>
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-report-plugin</artifactId>
  </plugin>
</plugins>
</reporting>
</project>

Pom.xml for the EJB Project:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>geospatial</groupId>
  <artifactId>geospatial-ejb</artifactId>
  <version>1.0</version>
  <name>geospatial-ejb</name>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-ejb-plugin</artifactId>
        <configuration>
          <ejbVersion>3.0</ejbVersion>
          <generateClient>true</generateClient>
          <archive>
            <manifest>
              <addClasspath>true</addClasspath>
            </manifest>
          </archive>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```
<plugin>
  <artifactId>maven-antrun-plugin</artifactId>
  <executions>
    <execution>
      <id>construct-jboss-xml</id>
      <phase>process-classes</phase>
      <configuration>
        <tasks>
          <copy
            file="${file.jboss.xml}"
tofile="target/classes/META-INF/jboss.xml"
overwrite="true">
            <filterset>
              <filter
token="DOMAIN"
value="${login.domain}" />
              <filter
token="AUTH_METHOD"
value="${login.auth.method}" />
            </filterset>
          <tasks>
        </configuration>
        <goals>
          <goal>run</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
</plugins>
</build>
</parent>
<groupId>geospatial</groupId>
<artifactId>app</artifactId>
</parent>
</dependencies>

<dependency>
  <groupId>jboss</groupId>
  <artifactId>jboss-annotations-ebj3</artifactId>
  <version>4.2.0.GA</version>
</dependency>
</dependency>
<dependency>
  <groupId>jboss</groupId>
  <artifactId>dom4j</artifactId>
  <version>1.6.1</version>
</dependency>
</dependency>
<dependency>
  <groupId>jboss</groupId>
  <artifactId>jboss-ebj3x</artifactId>
  <version>4.2.0.GA</version>
</dependency>
<dependency>
  <groupId>jboss</groupId>
  <artifactId>ejb3-persistence</artifactId>
  <version>4.2.0.GA</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>jboss</groupId>
  <artifactId>jboss-j2ee</artifactId>
  <version>4.2.0.GA</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>jboss</groupId>
  <artifactId>jbossall-client</artifactId>
  <version>4.2.0.GA</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>jboss</groupId>
  <artifactId>jbossws-client</artifactId>
  <version>4.2.0.GA.ws</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>jboss</groupId>
  <artifactId>jbossws-spi</artifactId>
  <version>4.2.0.GA.ws</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>jboss</groupId>
  <artifactId>jboss-jaxws</artifactId>
  <version>4.2.0.GA.ws</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>commons-codec</groupId>
  <artifactId>commons-codec</artifactId>
  <version>1.3</version>
</dependency>
<dependency>
  <groupId>commons-collections</groupId>
  <artifactId>commons-collections</artifactId>
  <version>3.2</version>
</dependency>
<dependency>
  <groupId>commons-beanutils</groupId>
  <artifactId>commons-beanutils</artifactId>
  <version>1.7.0</version>
</dependency>
<groupId>commons-digester</groupId>
<artifactId>commons-digester</artifactId>
<version>1.8</version>
</dependency>
<dependency>
<groupId>avalon-framework</groupId>
<artifactId>avalon-framework</artifactId>
<version>4.1.3</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
<dependency>
<groupId>serveltapi</groupId>
<artifactId>serveltapi</artifactId>
<version>2.3</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>stax</groupId>
<artifactId>stax</artifactId>
<version>1.1.1-dev</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>xmlbeans</groupId>
<artifactId>xmlbeans-jsr173-api</artifactId>
<version>2.0-dev</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>commons-lang</groupId>
<artifactId>commons-lang</artifactId>
<version>2.2</version>
</dependency>
<dependency>
<groupId>logkit</groupId>
<artifactId>logkit</artifactId>
<version>1.0.1</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>oro</groupId>
<artifactId>oro</artifactId>
<version>2.0.8</version>
<scope>provided</scope>

<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.3</version>
    <scope>provided</scope>
</dependency>

</dependencies>
</project>

Pom.xml for the Web module:

<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <parent>
        <groupId>geospatial</groupId>
        <artifactId>app</artifactId>
        <version>1.0</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <groupId>geospatial</groupId>
    <artifactId>geospatial-war</artifactId>
    <packaging>war</packaging>
    <name>geospatial-war</name>
    <version>1.0</version>
    <dependencies>
        <dependency>
            <groupId>geospatial</groupId>
            <artifactId>geospatial-ejb</artifactId>
            <version>1.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>jboss</groupId>
            <artifactId>jboss-annotations.ejb3</artifactId>
            <version>4.2.0.GA</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>jboss</groupId>
            <artifactId>jboss-ejb3x</artifactId>
            <version>4.2.0.GA</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>jboss</groupId>
            <artifactId>ejb3-persistence</artifactId>
        </dependency>
    </dependencies>
</project>
<version>4.2.0.GA</version>
<scope>provided</scope>
</dependency>
<dependency>
    <groupId>jboss</groupId>
    <artifactId>jboss-j2ee</artifactId>
    <version>4.2.0.GA</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>jboss</groupId>
    <artifactId>jbossall-client</artifactId>
    <version>4.2.0.GA</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>jboss</groupId>
    <artifactId>jbossws-client</artifactId>
    <version>4.2.0.GA</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>myfaces</groupId>
    <artifactId>myfaces-api</artifactId>
    <version>1.1.5</version>
</dependency>
<dependency>
    <groupId>myfaces</groupId>
    <artifactId>myfaces-impl</artifactId>
    <version>1.1.5</version>
</dependency>
<dependency>
    <groupId>myfaces</groupId>
    <artifactId>tomahawk</artifactId>
    <version>1.1.5</version>
</dependency>
<dependency>
    <groupId>commons-beanutils</groupId>
    <artifactId>commons-beanutils</artifactId>
    <version>1.7.0</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>commons-codec</groupId>
    <artifactId>commons-codec</artifactId>
    <version>1.3</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>commons-collections</groupId>
    <artifactId>commons-collections</artifactId>
    <version>3.1c</version>
    <scope>provided</scope>
<dependency>
    <groupId>commons-digester</groupId>
    <artifactId>commons-digester</artifactId>
    <version>1.7</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>commons-el</groupId>
    <artifactId>commons-el</artifactId>
    <version>1.0</version>
</dependency>
<dependency>
    <groupId>commons-lang</groupId>
    <artifactId>commons-lang</artifactId>
    <version>2.2</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>commons-validator</groupId>
    <artifactId>commons-validator</artifactId>
    <version>1.1.4</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.2</version>
</dependency>
<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.1.2</version>
</dependency>
<dependency>
    <groupId>taglibs</groupId>
    <artifactId>standard</artifactId>
    <version>1.1.2</version>
</dependency>
<dependency>
    <groupId>colt</groupId>
    <artifactId>colt</artifactId>
    <version>1.2.0</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>oro</groupId>
    <artifactId>oro</artifactId>
    <version>2.0.8</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>xerces</groupId>
    <artifactId>xercesImpl</artifactId>

<version>2.7.1</version>
<scope>provided</scope>
</dependency>
<dependency>
  <groupId>com.ibm.icu</groupId>
  <artifactId>icu4j</artifactId>
  <version>3.4.4</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>jdom</groupId>
  <artifactId>jdom</artifactId>
  <version>1.0</version>
</dependency>
<dependency>
  <groupId>com.sun.facelets</groupId>
  <artifactId>jsf-facelets</artifactId>
  <version>1.1.11</version>
</dependency>
<dependency>
  <groupId>javax.el</groupId>
  <artifactId>el-api</artifactId>
  <version>1.0</version>
</dependency>
<dependency>
  <groupId>com.sun.el</groupId>
  <artifactId>el-ri</artifactId>
  <version>1.0</version>
</dependency>
<dependency>
  <groupId>opensymphony</groupId>
  <artifactId>oscache</artifactId>
  <version>2.3</version>
</dependency>
</dependencies>
<build>
<plugins>
<plugin>
  <artifactId>maven-antrun-plugin</artifactId>
  <executions>
    <execution>
      <id>copy</id>
      <phase>process-sources</phase>
      <goals>
        <goal>run</goal>
      </goals>
      <configuration>
        <tasks>
          <copy
            file="src/main/xmltemplates/WEB-INF/web.xml"
<tasks>
  <copy file="src/main/xmltemplates/WEB-INF/jboss-web.xml"
    toFile="src/main/webapp/WEB-INF/jboss-web.xml"
    overwrite="true">
    <filterset>
      <filter token="DOMAIN"
        value="${login.domain}" />
    </filterset>
  </copy>
</tasks>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>
</project>
REFERENCES


4. Markus Schneider, Thomas Behr, Topological Relationships Between Complex Spatial Objects Nagapramod Mandagere, Buffer Operations in GIS in: University of Minnesota

5. http://rabasrv.jhuapl.edu

6. Songtao Jiang a, Byung Suk Lee a, Zhen He b, Cost modeling of spatial operators using non-parametric regression.


16. www.mysql.com/
17. http://rabasrv.jhuapl.edu/karma/index.php
VITA

Sonia Chib was born in Mumbai, India, on December 1, the daughter of Chamel Singh Chib and Raksha Chib. She turned vegetarian at the age of 12. After getting her Bachelors Degree in Information Technology from Mumbai University in 2006, she worked for Amdocs in the Information Technology sector. After a year, she entered the United States to pursue a Masters in Computer Science from the University of Texas at Dallas. For the past two years, she has worked in the UTD Semantic Lab as a Research Assistant. After finishing her work in summer 2009, she joined WorldLink, Frisco.