Final report for the “Geo-BI Dashboards” Google Summer of Code 2008 project (with the OSGeo)

by Etienne Dubé

What did I do during SoC?

A (not so) short intro to Business Intelligence (BI), and its relation to open source geospatial software.

My Google Summer of Code project with the OSGeo involved the development of a web mapping component for geo-analytical business intelligence (Geo-BI) dashboards. Since this is a rather new field for which no current OSGeo project exists, some definitions are in order.

According to Wikipedia, *"Business intelligence (BI) refers to technologies, applications and practices for the collection, integration, analysis, and presentation of business information and sometimes to the information itself. The purpose of business intelligence--a term that dates at least to 1958--is to support better business decision making. Thus, BI is also described as a decision support system (DSS) […]"* [1]. In other words, the main focus of BI is to provide knowledge from data, in order to help in decision making. As the name implies, BI is mainly destined to businesses, but other types of organizations are making use of BI: government, research and education.

Various practices and technologies are part of the BI field. The ones we’re interested in, for the context of this project, are data warehousing, Online Analytical Processing (OLAP) and dashboards.

Once again, from Wikipedia: *"A data warehouse is a repository of an organization's electronically stored data. Data warehouses are designed to facilitate reporting and analysis"* [2]. A data warehouse (DW) rely on databases management systems (DBMS) to store an organization’s historical data. DWs are usually kept separate from operational (transactional) systems; data are extracted from these operational systems (and often external data sources), transformed to correct errors and to fit the target schema (usually a denormalized *star schema*) of the DW, then loaded to the DW. Tools such as ETL (Extract, Transform and Load) can help with the task of integrating data sources and populating the warehouse. Geospatial data can also be integrated in a DW, thanks to spatial DBMS such as PostgreSQL with the PostGIS add-on. Data warehousing is a broad and complex subject; a very good introductory book is *The Data Warehouse Toolkit*, by Ralph Kimball and Margy Ross, published by Wiley (2nd edition; ISBN: 0471200247).

OLAP, for Online Analytical Processing, is *“an approach to quickly provide answers to analytical queries that are multi-dimensional in nature”* [3]. It also has been defined by Nigel Pendse as “Fast Analysis of Shared Multidimensional Information” [4]. In a more

technical sense, OLAP is a kind of database technology which relies on a multidimensional data model. While relational (SQL) databases are organized in terms of relations (tables), attributes (columns) and tuples (rows), OLAP datasets, or "data cubes", are expressed in terms of dimensions, levels, members and measures. Most implementations of OLAP use a data warehouse as their main data source. Special query languages exist for querying OLAP data cube: the most common is MDX (*Multidimensional expressions*), first developed by Microsoft (for SQL Server Analysis Services) but also found in other OLAP servers. MDX is to OLAP data cubes what SQL is to relational databases. Mondrian, an open source OLAP server, is available from Pentaho (an open source BI company) at http://mondrian.pentaho.org. It is developed in Java and uses MDX as its query language. A consortium of organizations working with open source OLAP software also developed an open source, product-neutral Java API for OLAP, named olap4j (http://www.olap4j.org). For now Mondrian and XML for Analysis (XMLA) data sources are supported with olap4j. Geospatial data is beginning to find its place in OLAP, thanks to various research projects focusing on Spatial OLAP (SOLAP) [5]. SOLAP adds capabilities such as mapping OLAP data (on thematic maps), navigating between levels of a spatial dimension on an interactive map (e.g. drilling down from a country to a state/province level) and in some cases doing spatial analysis natively in a geospatial data cube. Commercial Spatial OLAP products, offering some geospatial capabilities, are now available on the market (e.g. JMap Spatial OLAP). However, prior to the introduction of this project, open source offerings for Spatial OLAP have been pretty much inexistent.

Dashboards are BI applications that offer a quick visual glance at data, mostly exposing key performance indicators and high-level views. They are destined to executives and decision makers, and as such are designed to be easy to use, without having to know a query language. Dashboards can use a variety of data sources, both relational and OLAP. Dashboard designers will constructs views of the data with are of interest to decision makers. Interactive dashboards can allow the end-user to *drill* into the data, in order to view it at more detailed levels and, conversely, *roll-up* to go back to general views.

Increasingly, users want to have map displays in dashboards and other OLAP-based software. This can be done using ordinary (relational-based) web mapping technology, but this option offers limited interactivity for navigating through OLAP data structures (e.g. drilling down on a spatial member). It also increases the complexity of developing OLAP-based dashboards due to the efforts required to synchronize map displays (with data coming from a relational source, e.g. a PostGIS database) and OLAP displays (e.g. crosstabs, with data pulled from an OLAP server). An integrated approach, where geospatial data can be pulled directly from the (Spatial) OLAP server will overcome this limitation. This requires the development of an OLAP server capable of handling geospatial data types, and a web-mapping client which allows the query of Spatial OLAP data in order to plot it on a map.

Towards open source geospatial BI software

During the course of my masters degree and with the support of my supervisor, Prof. Thierry Badard (also the mentor for this GSoC project) and co-supervisor, Prof. Yvan

Bédard, I have tackled the first step, by starting the development of a Spatial OLAP server. It was done by modifying the open source Mondrian OLAP server, in order to add support for geospatial data types. The result is a project we name GeoMondrian (available shortly at http://geosoa.scg.ulaval.ca). For now, our experimental version is able to read geometry data from a PostgreSQL/PostGIS database and provide it to clients (using either the native Mondrian API or the olap4j API) as JTS (http://www.vividsolutions.com/jts) objects. More features are planned, such as MDX functions for handling geospatial data (to create spatial calculated measures, for example) and support for other geospatial DBMS.

The second step, developing a web-mapping client capable of querying Spatial OLAP, was the main goal of my Google Summer of Code project. The result is an open source web-mapping Spatial OLAP client that we named *Spatialytics*. It can be used in various web applications, ranging from simple, easy-to-use Geo-BI dashboards to fully fledged OLAP web clients including crosstabs, graphics and supporting a variety of navigation operators (drill-down, roll-up, pivot, etc.).

Architecture of Spatialytics

Spatialytics is composed of a client and a server component. The client is an HTML/JavaScript web application and is based on OpenLayers (http://www.openlayers.org) for the web mapping support, and uses Dojo (http://dojotoolkit.org) for user-interface widgets and client-server communication relying on Ajax techniques, i.e. XMLHttpRequest). The server is implemented as a Java servlet, and relies on olap4j (http://www.olap4j.org) and the Mondrian OLAP server (http://www.pentaho.org) for access to OLAP data, JTS for handling geometry objects and a part of the MapFish (http://trac.mapfish.org) server library for serializing GeoJSON data.

The server is queried using a simple HTTP GET or POST request, with a basic key/value pair encoding. Accepted parameters are an MDX query (for retrieving the needed OLAP data), OLAP navigation operators to apply to the query (e.g. drill-down on a member) and the data format to return; for now, data is returned in a JSON encoding, either GeoJSON or SolapJSON. The server is able to return simple GeoJSON (http://geojson.org) data, but this wouldn't be sufficient for displaying OLAP data (crosstabs, charts, etc.) and handling queries and navigation operators in a client. Thus, a new JSON encoding, named *SolapJSON*, was designed. It is a combination of an OLAP result set (cells and members returned by a query), a GeoJSON data set (used to display the map in OpenLayers) and some metadata. SolapJSON is documented in the source code package of Spatialytics.

The client currently supports simple drill-down and roll-up OLAP navigation operators with the map display. Drilling down is the operation of displaying the children of a selected member, in order to view data at a more detailed level. For example, the user can click on a country in a choropleth map (based on dynamic equal or fixed intervals), representing the GDP of countries, in order to view the individual GDP of each state/province in this country. Rolling up is the inverse operation, i.e. going back to a less

detailed level. OLAP dimensions can contain many levels, and thus the user is able to view data at many different levels of detail simply by clicking on the features of the map.

What have I learned during SoC?

What I learned during the course of this project can be divided into two categories: technical and non-technical knowledge.

Technical

- JavaScript and Ajax programming: this project was my first real-world use of JavaScript and Ajax techniques. I learned how to code properly in JavaScript (including prototype-based OO and the use of closure functions) and how to use XMLHttpRequest and the DOM API to dynamically update the page.
- OpenLayers: I had a chance to learn how to do pretty funky stuff with this web mapping client. Looking at its code also proved useful for learning how a large JavaScript project is structured.
- Dojo: by using this JavaScript toolkit, I learned a lot on Ajax and the amazing things that can be done in DHTML.
- OLAP and MDX: by working with olap4j (especially with my contribution of a MDX query transformation package to this project), I learned new tricks with MDX and how to implement basic OLAP navigation operators (drill-down, roll-up) with this query language.
- Java Servlets: I perfected my knowledge of Java Servlet programming and the use of Servlet containers (especially Apache Tomcat).

Non-technical

- Collaborating with other open source projects: my contribution to olap4j has been a nice opportunity for learning how to interact with an open source project community, develop a new functionality and contribute it as a patch. The feedback to my contribution has been very positive. It also demonstrated to me that open source development often takes place in a large ecosystem rather than in a closed vase; a developer working on project X and depending on project Y (as a library or component) is usually in a good position to contribute to this project Y.
- Handling blocking problems and managing change: the support for OLAP navigation operators (drill-down, roll-up, etc.) in the application required features in external components (olap4j) that were not yet implemented. This was an important blocking problem and the best way to resolve it was to implement these features in olap4j. However, the time required to do so could not be spent on other parts of the GSoC project. Together with my mentor, we decided which initial objectives were the most important and needed to be done, and which ones I could put aside to work on the required MDX query transformation package for olap4j. It turned out well, since I had enough time to make a good and reusable contribution to olap4j, and to finish the most important parts of Spatialytics.

What future improvements are planned?

The current version of Spatialytics is a fully functional proof-of-concept of an OLAP web mapping client, but it has some limitations due to a few shortcuts that were taken to complete the project in the given timeframe. The limitations and planned improvements are as follows:

- Only choropleth thematic maps (with polygon features) are supported.
- Spatial members are only supported on the ROWS axis of a query. This will be changed so that mapping works with spatial members placed on any axis.
- Only the first column of an OLAP query result (cellset) is used for the values to display (and to determine polygon colors) in the choropleth maps. A control will be added to select the column of the cellset to be displayed on the map. Also, multiple columns could be displayed on the map at once, using alternate mapping styles such as pie charts or bar graphs symbols.
- Current OLAP navigation operators, relying on the MDX query transform package contributed to olap4j, only work with a single dimension on an axis. It is planned to improve the query transforms to support more than one dimension on an axis (by cross-joining members). Also, other types of drill-down and roll-up operators are to be implemented.

Other planned improvements include:

- Support for other thematic mapping styles, including proportional symbols, pie charts and bar graph symbols, display of line and point geometries.
- Legend display.
- Crosstabs and graphics views. This will enable the development of rich Geo-BI dashboards and SOLAP applications which include maps, tables and charts.
- Developing a free Spatial OLAP data cube to be included with the Spatialytics distribution.

Where can I get the code?

A website for the project is currently in development, at http://www.spatialytics.org. Stay tuned for a first official release by the end of November 2008!

Endnotes

[1] Wikipedia, 2008, *Business Intelligence*, http://en.wikipedia.org/wiki/Business_intelligence

[2] Wikipedia, 2008, *Data warehouse*, http://en.wikipedia.org/wiki/Data_warehouse

[3] Wikipedia, 2008, *Online Analytical Processing*, http://en.wikipedia.org/wiki/Online_analytical_processing

[4] Nigel Pendse (2008-03-03). "What is OLAP? An analysis of what the often misused OLAP term is supposed to mean" (http://www.olapreport.com/fasmi.htm). OLAP Report. Retrieved on 2008-03-18.

[5] For examples, see the paper available at http://www.ij-healthgeographics.com/imedia/5548770972242483_article.pdf. It deals with the implementation of a SOLAP client in a Public Health Surveillance context. Other examples and papers could be found on the website of the GeoSOA research group (http://geosoa.scg.ulaval.ca) and on the one of the NSERC Industrial Research Chair in Geospatial Databases for Decision Support, at http://mdspatialdb.chair.scg.ulaval.ca/english/Eindex.asp . Disclaimer: I have been involved in these research groups, thus I am familiar with the research they have done, and this is why I'm quoting them. Other people around the world are also working on this topic.