GGL: A geo-processing definition language that enhance spatial SQL with parameterization

Fernando González Cortés[a,c] and Thomas Leduc[b,d]
[a] Geographical Information Science Consultant, Valencia, Spain
[b] CERMA UMR CNRS/MCC 1563 - ensa Nantes
6, quai François Mitterrand, BP 16202 - 44262 Nantes cedex 2 - France
[c] fernando@fergonco.es, d thomas.leduc@cerma.archi.fr

ABSTRACT

The proliferation of Spatial Data Infrastructures (SDI) along with improved acquisition techniques induces online dissemination of great amounts of data, increasing in the same way the complexity of new models and the need of tools to process and analyze them. GearScape project provides a language based on SQL standard and enabled spatially according to OGC specification for geographic information. In order to make scripts more usable, this language has been extended so that they can be defined on abstract data and, therefore, applied on a wide set of inputs. The aim of this short paper is to present and justify this extension.

INTRODUCTION

Spatial languages are suitable tools to define and communicate analysis on data. In particular, Structured Query Language (SQL) extended with the OGC Implementation specification for geographic information (OGC, 2006) is one of the most useful solutions to perform spatial data analysis (Bocher *et al*. 2008, Leduc *et al*. 2009). SQL has several advantages over imperative programming. Processes are built in a declarative way and the SQL engine looks after optimizations, index usage, etc. This simplicity also reduces the learning curve and makes it harder to introduce a bug in SQL code than in imperative code.

Several extensions have been made to adapt SQL implementations to spatial data analysis. Oracle spatial and PostGIS are well-known examples, but there are also generic solutions that can be applied to different DBMS in order to enable them spatially. Although GearScape is not a DBMS, it provides the GearScape Geoprocessing Language (GGL) which implements both SQL (part of the ISO/IEC 9075:1992 standard) and OGC spatial specifications.

However, the analysis defined in Spatial SQL are tightly coupled with the data they operate on and therefore it is not possible to apply the same analysis on several data sets without prior modification of the direct references to the tables involved. To solve this drawback, some extension to the GGL implementation has been made. This extension encourages collaborative script development.

After a brief overview of the relevance of the declarative programming paradigm in the geo-processing context, we will focus on its main drawback: the lack of parameterization ability. The remainder of this paper is dedicated to the bypass solution we developed in the GGL implementation. At last, we present a use case to demonstrate the viability of our approach.

PROPOSED SOLUTION

In SQL scripts, all data references are hard-coded. To address this drawback, GGL implementation has been extended letting the user to define a script on abstract input data. These abstract inputs are defined as script parameters so that the script body references the parameters

instead of the actual sources. Scripts thus defined does not contain a concrete geoprocess but a geoprocess definition, that can be applied to a wide set of data independently of their structure (provided they match the abstract requirements).

This solution enables the execution of a generic geoprocess chain on different data sets, translating a concrete problem into a generic one. It is possible to imagine scenarios where advanced users communicate geoprocesses to basic users to help them perform their spatial analysis, or where two or more advanced users help each other to solve complex problems by sharing some scripts.

Contrarily to PL/SQL (Procedural Language/Structured Query Language), the solution we propose is not a procedural one. What is required is a simple solution to put scalar literals and tables as script parameters keeping the simplicity of SQL and not introducing new programming paradigms to be learned. Tabular parameters are not just a single character string but a data structure which defines the fields required by the script code. The fact that tabular parameters in GGL contain information about the fields that will be used in the script makes it possible to validate the actual parameters before the execution begins. As an example of this encapsulation process, at the beginning of the script execution process, a wizard is provided so as to filter valid sources, based on the parameters' types.

The SQL*Plus Substitution Variables mechanism (Jones, 2004) also provides a declarative enhancement that uncouples SQL script code from the tables accessed at runtime. However, as this solution treats parameters as character strings, no information about the field requirements of referenced tables can be defined, and therefore, it is not possible to validate the field references in the script code before runtime.

Parameterization of SQL queries

Let us consider a table named countries containing just one field 'name' with country names and another one called 'population' with countries population. On such a table, the query "*SELECT name FROM countries WHERE population > 20.000.000*" would obtain the names of those countries which population is greater than 20 millions.

The concrete problem of obtaining those countries can be turned into the general problem of obtaining the rows with a field greater than a specified amount. A generic rephrasing of the previous query is: "*SELECT descriptive_field FROM source WHERE check_field > threshold_value*" using an abstract numeric value called "threshold_value" and an abstract table called 'source' which must contain two fields: 'descriptive_field' of any type; and 'check_field' of a numeric type in order to be evaluated by the *greater than* operator.

It is obvious that such a generic script suited to the extraction of some cities in the aforementioned 'countries' table, is also convenient to the identification of some cities with an area greater than 100 square kilometers, or of the railways longer than 1000 km, etc.

USE CASE

The use case presented in this section aims to prove the viability of the proposed solution. It provides a easy to run general solution to identify some shapes modifications in a geographic layer.

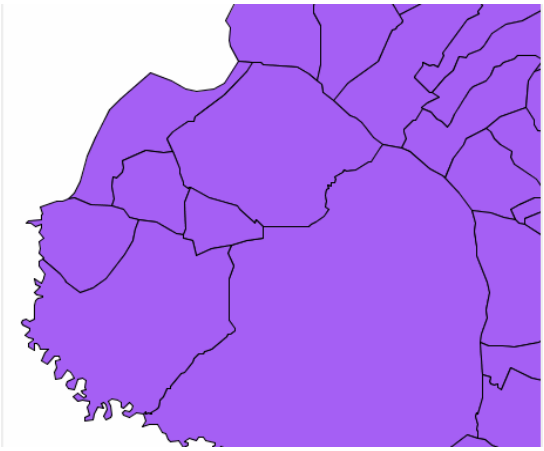

Figure 1: Original source.

In this example, a data source (see figure 1) has been modified and the user needs to know where and what the modifications are. Figure 2 shows the modified source. Identify them all is not an easy task.

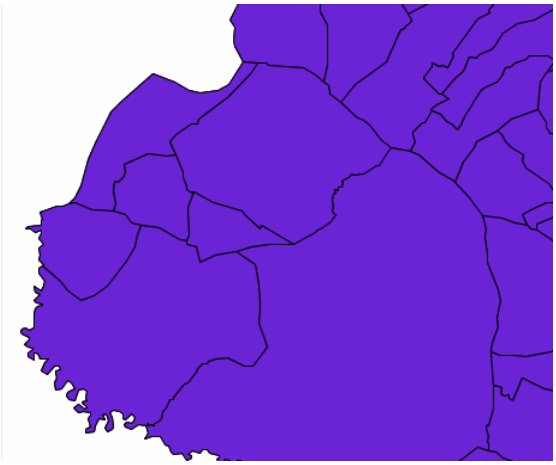

Figure 2: Modified source.

However, provided that the source has a primary key field called *id*, it is possible to use a spatial join query so as to obtain a new source with all the registered modifications: "*select symdifference(geom1, geom2) from source1 s1, source2 s2 where s1.id = s2.id"*.

```
-- Compares two sources and generates a result with the
differences
DOC('First source to compare');
DECLARE source1:TABLE(geom1:GEOMETRY, id:ANY);
DOC('Second source to compare');
DECLARE source2:TABLE(geom2:GEOMETRY, id:ANY);

CREATE TABLE symdiff AS
    SELECT symdifference(geom1, geom2)
        as the_geom, geom1, geom2, s1.id
    FROM source1 s1, source2 s2
    WHERE s1.id = s2.id;
```

Figure 3: GGL script that obtains the differences between two sources.

When the script presented in figure 3 runs, GearScape analyzes it and shows an execution wizard to the user, presenting the first script comment in the script description step as it can be seen in figure 4 and asking all the actual values to be passed to the script parameters (see figure 5).

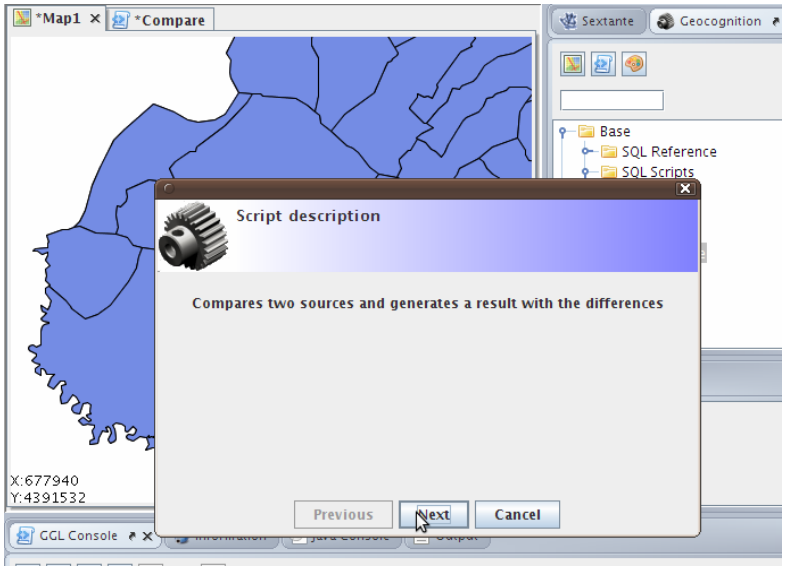

Figure 4: GGL wizard showing the script description.

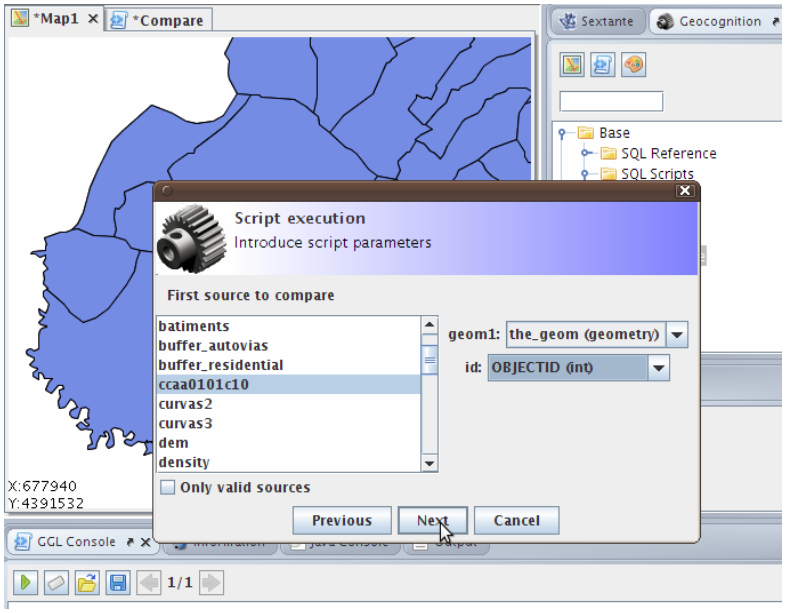

Figure 5: GGL wizard asking for the script parameters.

Finally, the GGL engine executes the script with the data the user has specified and the result contains the modified features (see figure 6).

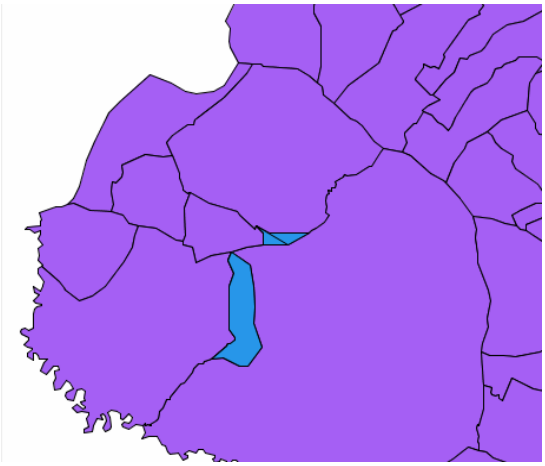

Figure 6: Result of the GGL script.

CONCLUSION AND OUTLOOK

What has been presented in this paper is a simple solution to pass argument values to a portion of spatial SQL code. It makes possible the resolution of general analysis problems by defining general geoprocessing operations, being therefore interesting for a greater number of people and, encouraging collaboration between them.

In prospect and as a result of the 52°North Student Innovation Prize for Geoinformatics (González *et al.* 2009), GGL scripts have been integrated in 52ºNorth WPS implementation. Main benefit is that the deployment of scripts on a running server makes them accessible to a wider set of users.

Finally, there is a work in progress that will hopefully allow the execution of scripts in computer grids in order to reduce both execution time and memory constraints.

AVAILABILITY

GGL and its GearScape front-end are available for public download at http://www.gearscape.org/.

BIBLIOGRAPHY

Bocher, E., Leduc, T., Moreau, G., and González Cortés, F. (2008). GDMS : an abstraction layer to enhance Spatial Data Infrastructures usability. In 11th AGILE International Conference on Geographic Information Science — AGILE'2008, Girona, Spain.

Clinton, B. (1994). Executive order 12906: Coordinating geographic data acquisition and access: The national spatial data infrastructure. www.archives.gov/federal-register/executive-orders/pdf/12906.pdf.

Franklin, C. (1992). An introduction to geographic information systems : linking maps to databases. Database, 15(2) :12–21.

González Cortés, V., Schäffer, B., and González Cortés, F. (2009). Publicación y uso de scripts SQL en servidores WPS transaccionales. In VI Jornadas Técnicas de la IDE de España, JIDEE 2009, España.

INSPIRE (2007). Directive of the European Parliament and of the Council establishing an Infrastructure for SpatialInformation in the European Community, http://www.ec-gis.org/inspire/.

ISO (1992). Structured Query Language – 92, International Organization for Standarization, 1992.

Jones C. (2004). Oracle SQL*Plus Technology - Substitution Variables Technical Whitepaper, 2004. http://www.oracle.com/technology/support/tech/sql_plus/htdocs/sub_var.html

Kiehle C. (2006). Business logic for geoprocessing of distributed data, Computers & Geosciences, vol. 32, 2006, pp. 1746–1757.

Leduc, T., Bocher, E., González Cortés, F., and Moreau, G. (2009). GDMS-R: A mixed SQL to manage raster and vector data. In GIS Ostrava 2009 - Symposium on Seamless Geoinformation Technologies, Ostrava, Czech Republic.

OGC (2006). OpenGIS implementation specification for geographic information – Simple feature access – Part 1: Common architecture, Open Geospatial Consortium. http://www.opengeospatial.org/standards/sfs.

OGC (2006). OpenGIS implementation specification for geographic information – Simple feature access – Part 2: SQL option, Open Geospatial Consortium. http://www.opengeospatial.org/standards/sfs.