GEOMETRY VIEWER FOR PGADMIN4: A PROCESS GUIDED BY THE GOOGLE SUMMER OF CODE

X. Gong[1], F. Erwee[2], V. Rautenbach[3, *]

[1] Institute of Remote Sensing and Geographic Information Systems, School of Earth and Space Sciences, Peking University - xurigoong@gmail.com
[2] Verge Technologies, Pretoria, South Africa - ferwee@gmail.com
[3] Centre for Geoinformation Science, Department of Geography, Geoinformatics and Meteorology, University of Pretoria, South Africa - victoria.rautenbach@up.ac.za

Commission IV, WG IV/4

KEY WORDS: PostGIS, pgAdmin, geometry viewer, Leaflet JS

ABSTRACT:

The latest version of pgAdmin4 was released in mid-2016 and moved to a web-based application that was written in Python and jQuery with Bootstrap, using the Flask framework. This new architecture of pgAdmin4 provided an excellent opportunity to integrate a geometry viewer into the application. This progress started as the geometry viewer was selected as a project for the 2018 Google Summer of Code (GSoC). The requirements for the geometry viewer was elicited through conversations with the mentors and emails to the discussion list of PostGIS and pgAdmin. Once the formal design was finalized the development started. The spatial technology stack implemented to expand pgAdmin4 with a geometry viewer was the JavaScript mapping library Leaflet JS and WKX - parser/serializer library that supports several spatial vector formats. Both these fulfilled the requirements of the coding standard of pgAdmin that all client-side code must be developed in JavaScript using jQuery and other plugins. Leaflet JS is well known for its ease of use and compatibility. WKX is lesser known but well supported and concise to the need to parse the spatial data before rendering on the Leaflet map. The decision on both of these libraries was motivated by their minimal size and possibilities for expansion for future extensions of the viewer. The first version of the geometry viewer was well-received and is currently integrated into the latest versions of pgAdmin4.

1. INTRODUCTION

PostgreSQL (https://www.postgresql.org) is a popular open source object-relational database that focuses on extensibility. One of these extensions is PostGIS (https://postgis.net) which adds support for geographic objects (i.e. vector and raster data) and geospatial functions (e.g. intersects and buffers). PostgreSQL with PostGIS is the most popular database system and is used in various geospatial applications. This can be attributed to the fact that PostgreSQL is open source and the maturity of the support for both vector and raster geographic objects available in PostGIS.

pgAdmin (https://www.pgadmin.org/) is generally used as the graphic user interface (GUI) management tool for PostgreSQL and installed by default with new installations. Prior to 2018, if a user executed a query on geographic objects, the user would need to use an external application, such as QGIS or OpenJump to view the resulting geometries. pgAdmin4 was released in mid-2016 as a web-based application written in Python and jQuery with Bootstrap, using the Flask framework. This new architecture of pgAdmin4 provided an excellent opportunity to integrate a geometry viewer into the application. After the launch of pgAdmin4, an enhancement ticket for a geometry viewer was created, however, there was no progress on this ticket until the 2018 GSoC programme. In 2018, a geometry viewer for pgAdmin4 was developed as part of the 2018 Google Summer of Code (GSoC).

In this paper, we present an overview of the requirements for the pgAdmin geometry viewer and the process followed to implement the viewer during the 2018 GSoC programme. The remainder of the paper is structured as follows: Section 2 provides an overview of the Google Summer of Code programme; in Section 3 we present the requirements of the geometry viewer; the design and implementation are discussed in Section 4; and conclusion are offered in Section 5.

2. GOOGLE SUMMER OF CODE

Google has two programmes to introduce pre-university and university students to open source, namely Google Code-in (GCI) and Google Summer of Code (GSoC) (https://summerofcode.withgoogle.com/), respectively. GSoC was first established by Google in 2005 and has grown ever since. GSoC is an online, international program targeted to university students, that aims at fostering their participation in open source software communities (Rautenbach et al, 2018; Brookes et al, 2015). Mentoring organizations select students that will be developing software applications for 12 weeks and receiving support and feedback from mentors within the software community. Successful students are paid stipends by Google. OSGeo is a veteran organization having participated in GSoC and having graduated 190 (at 2018) students from all over the world every year since 2007.

In 2018, a total of 1 072 students from 59 countries completed the GSoC program (Google, 2018). Students worked with 212 open source organizations with over 2 100 international mentors.

* Corresponding author

The aim of GSoC is to bring new developers into open source software communities, as well as exposing students to real world software development. For the open source communities, GSoC is a great opportunity for the students to implement fixed, updates or nice-to-have features that the developers do not always get to.

3. REQUIREMENTS OF A GEOMETRY VIEWER FOR PGADMIN4

After the student projects for the 2018 GSoC programme was announced, there was a three week period known as Community Bonding. During this time, the students would get to know the community, the project and also research their project more. For the geometry viewer, the mentors collected some basic requirements from the PostGIS community, but during this period, the students, communicated more with both the PostGIS and pgAdmin communities to gain a better understanding on their needs. The requirements elicitation did not stop after the bonding period, and updates were regularly sent to the communities asking for input.

The following main requirements for the geometry viewer was established:

- **Allow the user to view: an entire table, only selected rows, or the results from a query**
  The geometry viewer in pgAdmin4 will allow users to view the geometries in a spatial database and the results of queries executed. Users can directly view the geometries on a map within the pgAdmin4 GUI. It should also allow selection of geometry row on the map and show the associated column value for that row.

- **Support 2d for both the Geometry and Geography datatypes**
  PostGIS has built-in supports both Geometry and Geography data type (i.e. "geographic" coordinates) (PostGIS 2019). The Geometry data type in PostGIS is based on the ISO 19125-2:2004, *Geographic information - Simple feature access -- Part 2: SQL option* and extends it with support for 3dm, 3dz, 4d and SQL-MM format. The geometry viewer should support the 2d data in both types (see Table 1).

Table 1: Data type that the geometry viewer should support

Data Type	Subtype	Should Support
Geometry	2d 3dm, 3dz, 4d SQL-MM*	Yes
Geography	2d	Yes

** The geometry viewer does not support 3dm, 3dz, 4d, and SQL-MM.*

- **Support different SRIDs**
  PostGIS supports different kinds of geographic and projection coordinate system with over 3000 predefined European Petroleum Survey Group (EPSG) and the related Spatial Reference Identifiers (SRIDs) available in *spatial_ref_sys* table. An SRID is a unique identifier associated with a specific coordinate system, tolerance, and resolution.

  It's possible that there are geometries with different SRIDs in the same table and the viewer should support different coordinate systems. However, the viewer should avoid drawing geometries with mixed SRID in the same map.

- **Handle results with multiple columns containing geospatial objects**
  It's possible that there are two or more columns in the query results with geospatial objects. The viewer should render one column in one map and offer the user the options to select another column to view on the map rather.

- **Performance – ability to handle large datasets**
  The viewer should be efficient enough to show tens of thousands of geometries on a map and it should also set a proper limit for the data it will render to avoid crashing with too much data.

- **Should consist of open source technology exclusively**
  The spatial technology stack implemented to expand pgAdmin4 with a geometry viewer was the JavaScript mapping library Leaflet JS and WKX - parser/serializer library that supports several spatial vector formats. Both these fulfilled the requirements of the coding standard of pgAdmin that all client-side code must be developed in JavaScript using jQuery and other plugins.

  Leaflet JS is well known for its ease of use and compatibility. WKX is lesser known but well supported and concise to the need to parse the spatial data before rendering on the Leaflet map. The decision on both of these libraries was motivated by their minimal size and possibilities for expansion for future extensions of the viewer.

4. DESIGN AND IMPLEMENTATION OF THE GEOMETRY VIEWER FOR PGADMIN4

Once the formal design was complete, the development started. The GSoC programme allows for 12 weeks of development that would also include the testing of the solution developed. The students had to report weekly on his/her progress and plans for the upcoming week.

pgAdmin4 is a web-based application with a server written in Python and front-end pages. Each time the user submits a query using the pgAdmin interface, the web browser sends the query to the web server and then the server passes the query to the PostgreSQL database. Once the database completes the query, the web server wraps the result and pass it to the browser. If the result contains geometry data the geometry viewer will parse the data and show the geometries in a map (see the flow chart in Figure 1). Therefore, the viewer will not run the query automatically and it will just parse and render the output data queried by the origin query tool. By default, the geometries are output in Extended Well-Known Binary (EWKB) format that is not directly supported by most web map libraries so the program will parse and convert EWKB data before rendering geometries.

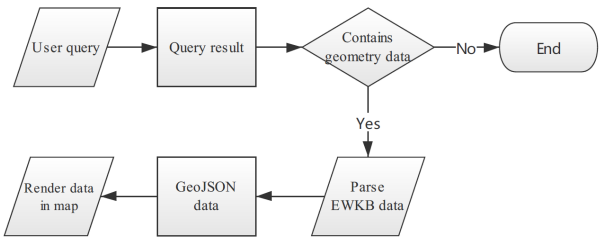
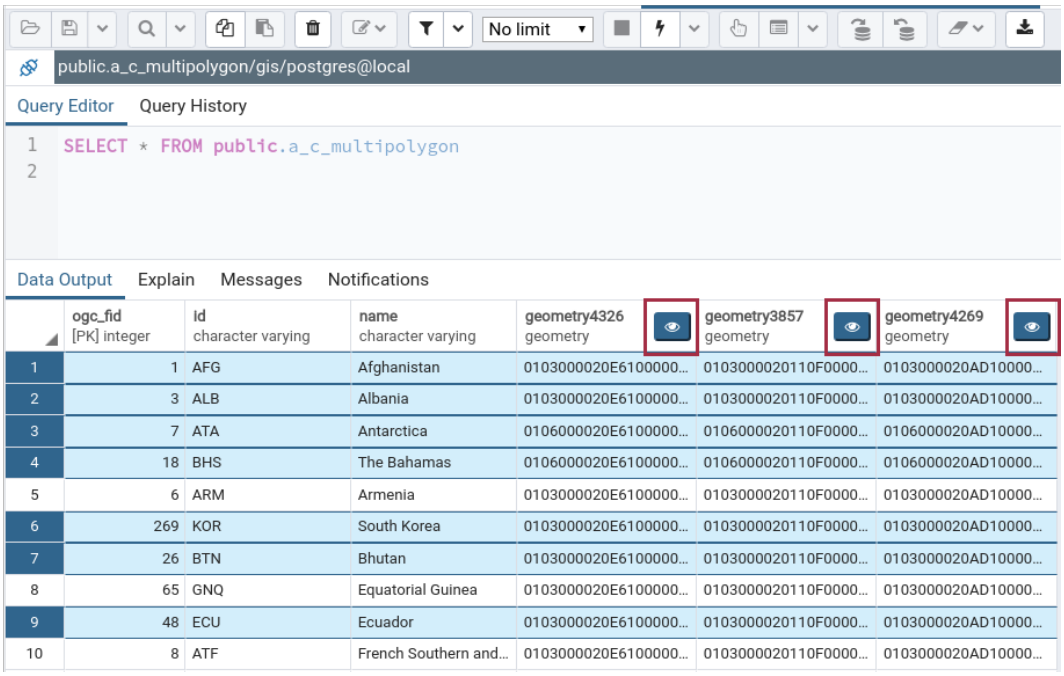


Figure 1. The flow chart of geometry viewer

pgAdmin4 web interface uses several panels to display the result of queries. The geometry viewer will be set as an additional panel

to display the map within. The browser will render both the grid panel to show table and the viewer panel to show map after each query. Then users can browse the geometries by zooming and panning in the map. In order to reduce the complexity of use and offer the connection between the already existed data gird and the viewer, we add a "view" button in each geometry column header and the map panel will show up after users clicking the button (see Figure 2). In this way, users can choose which column to show in the map even with multi geometry columns. Users would also be able to choose the rows they want to view by selecting the rows in data grid (if no row is select, it will load the entire result and render all the rows).



	ogc_fid [PK] integer	id character varying	name character varying	geometry4326 geometry	geometry3857 geometry	geometry4269 geometry
1	1	AFG	Afghanistan	0103000020E6100000...	0103000020110F0000...	0103000020AD10000...
2	3	ALB	Albania	0103000020E6100000...	0103000020110F0000...	0103000020AD10000...
3	7	ATA	Antarctica	0106000020E6100000...	0106000020110F0000...	0106000020AD10000...
4	18	BHS	The Bahamas	0106000020E6100000...	0106000020110F0000...	0106000020AD10000...
5	6	ARM	Armenia	0103000020E6100000...	0103000020110F0000...	0103000020AD10000...
6	269	KOR	South Korea	0103000020E6100000...	0103000020110F0000...	0103000020AD10000...
7	26	BTN	Bhutan	0103000020E6100000...	0103000020110F0000...	0103000020AD10000...
8	65	GNQ	Equatorial Guinea	0103000020E6100000...	0103000020110F0000...	0103000020AD10000...
9	48	ECU	Ecuador	0103000020E6100000...	0103000020110F0000...	0103000020AD10000...
10	8	ATF	French Southern and...	0103000020E6100000...	0103000020110F0000...	0103000020AD10000...

Figure 2. The "view" button in the geometry column header

After the header button is clicked, the program will parse the EWKB data in the column using the WKX library. Geometries in the same column may have different SRIDs and this makes it a bit difficult to handle. We proposed two basic solutions for this situation:

1. Transform the geometries with different coordinate systems into the same. In this way, the viewer needs to look up the spatial_ref_sys table for the SRID info to do the transformation.
2. Group the geometries by SRID and render the group with the largest number of geometries. Also, remind users that there are geometries with different SRIDs not rendered.

After discussing in the community, we chose the second solution as coordinate transformation is not an easy task in front-end and will create issues if it fails.

The Leaflet JS is used to create an interactive map and render geometries for its simplicity and high performance. It supports geometry collections as well as columns with mixed geometry types. If the query result would contain too much data that may crash the viewer. To avoid that, we limit both the amount and the total size of the geometry data because in some extreme cases some geometries may have very complex shapes with many vertices.

As shown in Figure 3, at first, we placed the map in a dialog, and it would pop up after the user clicking the "view" button. After discussing this design in PostGIS and pgAdmin mailing list, we decided to create a new tab alongside the data grid tab to display the map. This avoids the occlusion of the data grid and users can adjust the layout or close the tab as they like. When users click a feature in the map, a small pop-up will show up with its property table. If the SRID is 4326 (WGS84) the viewer will transform the coordinates to Web Mercator and add background tile layer. Figure 4 shows the result of the geometry viewer.

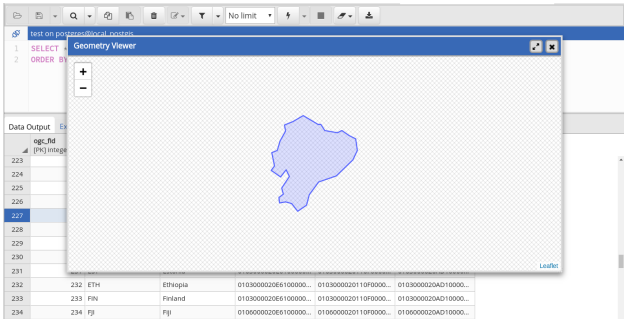
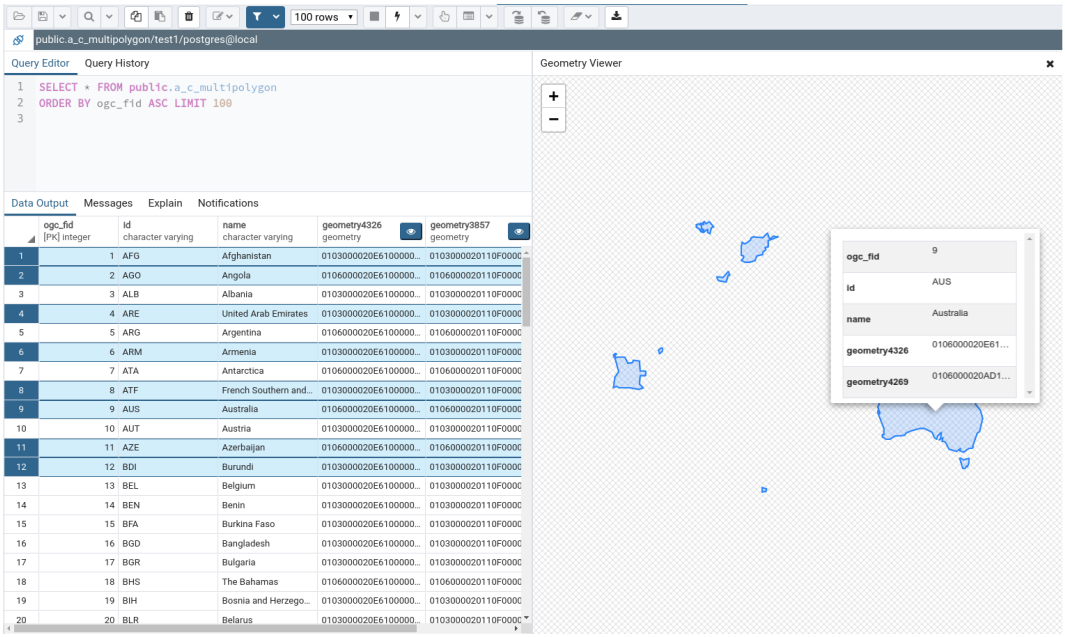


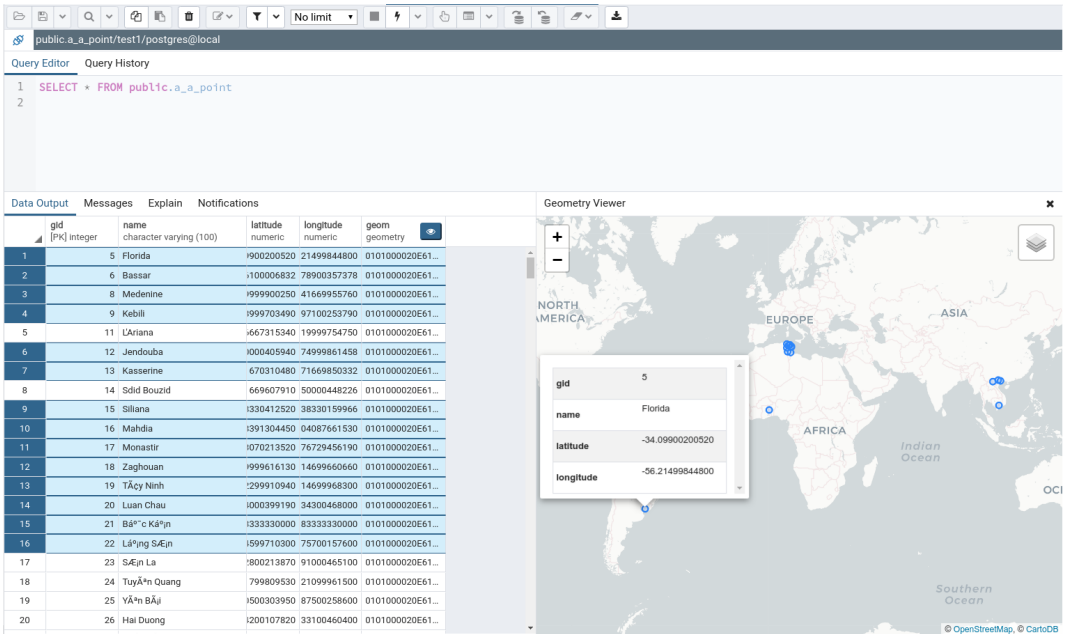
Figure 3. Geometry viewer in a dialog

After GSoC, the geometry viewer was integrated into the latest version of pgAdmin4 and very well received by the community. However, not all the community requirements have yet been addressed in the first iterations and another project on improving the geometry viewer was proposed for the 2019 GSoC. The focus would have been on fixing minor bugs and adding new features for overlaying queries and basic styling. Unfortunately, no student was found for this.

This contribution has been peer-reviewed.
https://doi.org/10.5194/isprs-archives-XLII-4-W14-79-2019



(a) Multipolygon



(b) Point geometry with background tile layer

Figure 4: The result of geometry viewer

5. CONCLUSION

In this paper, we presented the pgAdmin geometry viewer that was developed as part of the 2018 GSoC programme. The geometry viewer added the ability to view geometries within pgAdmin whereas user previous had to rely on external applications, for example, QGIS or OpenJump, to view query results.

The requirements elicitation relied greatly on the feedback and suggestions of the PostGIS and pgAdmin communities and can directly be translated to the success of the viewer. This aspect of communication with the community and understanding their needs is invaluable for the students. GSoC provides a very structured environment where students can as for assistance and their progress is also monitored. This allows students to be introduced gently to the open source community and would hopefully encourage future participation.

The Google GSoC and Code-in programme are great opportunities for the open source community to promote themselves and attract new developers to their projects. Additionally, the students that participate in these programmes have reported that the opportunity to work on a real-life project

full-time for 12 weeks is a great work experience and have significantly improved their development skills. The students also learn how open source communities work and how to contribute. The students would strongly recommend that their peers participate, if possible.

ACKNOWLEDGEMENTS

The authors would like to thank all the PostGIS and pgAdmin communities for their guidance and support. We would also like to thank Google and the Google Summer of Code administrators for hosting and coordinating this programme.

REFERENCES

Brookes, E.H., Kapoor, A., Patra, P., Marru, S., Singh, R., Pierce, M., 2015. GSoC 2015 student contributions to GenApp and Airavata, *Concurrency and Computation: Practice and Experience*, 23(7), pp. 1960-1970.

Google, 2018, That's a wrap for Google Summer of Code 2018, https://opensource.googleblog.com/2018/08/thats-a-wrap-gsoc-2018.html (6 June 2019)

ISO 19125-2: 2004 (2004). *Geographic information - Simple feature access -- Part 2: SQL option*, International Organization for Standardisation (ISO), Geneva, Switzerland.

PostGIS, 2019. PostGIS documentation, https://postgis.net/docs/using_postgis_dbmanagement.html (21 May 2019)

Rautenbach, V., Di Leo, M., Andreo, V., Delucchi, L., Kudrnovsky, H., McKenna, J., Acosta y Lara, S., 2018. Fostering pre-university student participation in OSGeo through the Google Code-in competition, *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume XLII-4/W8, 2018 FOSS4G 2018, 29–31 August 2018, Dar es Salaam, Tanzania.