AGILE: GIScience Series, 3, 67, 2022. https://doi.org/10.5194/agile-giss-3-67-2022
Proceedings of the 25th AGILE Conference on Geographic Information Science, 2022.
Editors: E. Parseliunas, A. Mansourian, P. Partsinevelos, and J. Suziedelyte-Visockiene.
This contribution underwent peer review based on a full paper submission.

Open source vector tile creation for spatial data infrastructure applications

Wallner Andreas Georg[1], Piechl Thomas[2], Paulus Gernot[1] and Anders Karl-Heinrich[1]

[1] Carinthian University of Applied Science (CUAS) - Spatial Information Management, Villach, Austria
[2] Amt der Kärntner Landesregierung, Klagenfurt, Austria

Correspondence: Wallner Andreas Georg (andreasgeorg.wallner@edu.fh-kaernten.ac.at)

Abstract. Accessing geospatial data via the internet is a common way for data integration. This web mapping approach often relies on web services like the web map tile service (WMTS) for accessing maps or the web feature service (WFS) for accessing vector data. An alternative way, which combines aspects like the higher performance through tiling and the usage of vector data is the usage of tiled vector data. This short paper describes an approach for the creation of tiled vector data using standard PostgreSQL, with spatial extension PostGIS, functionality. For that, custom PostGIS functions are implemented to select relevant vector data out of the PostgreSQL database, dynamically generalize/simplify geometries and transform the data to Mapbox mvt format. This approach of creating tiled vector data shall be used for further implementations of web maps for building more efficient spatial data infrastructures.

Keywords. tiling vector data, generalisation, web mapping, PostgreSQL

1 Introduction

Online web applications often rely on external spatial data sources offered through the world wide web (Fu and Sun, 2011). One of the most important services for web mapping is the traditional Open Geospatial Consortium (OGC) Web Map Service and OGC Web Map Tile Service (WMS/WMTS). The WMS is used to offer maps as images over the internet. Original data is often stored in vector format. According to the request of a client, the WMS responds with a pre-calculated geo-referenced image (Fu and Sun, 2011). Because WMS responds with one, often bigger image, this can take some time. Therefore, the WMTS was developed. The idea is to cut the resulting images into equally sized tiles, and only to request needed tiles, rather than a whole image (OGC (a), 2010). For vector data, the OGC Web Feature Service (WFS) allows the request for geographical features via the internet. WFS serves direct access to fine-grained geographic information at feature level. The standard describes operations for discover, query and transfer geospatial vector data in different formats like for example GML, JSON, CSV or even shapefile (OGC (b), 2010). The transmission of vector data can take a lot of time due to the complexity of vector objects (Li et al., 2017). Therefore, the interest in tiled vector data services increases, to increase the performance of the data transfer and to efficiently use vector data for web mapping (OGC, 2018).

1.1 Tiling spatial data

The concept of tiling spatial data is nothing new. Goodchild (1990) describes general concepts for tiling large geospatial data to make it more accessible or manageable. Early implementations are even going back to the 1970s/80, for example, the implementations of the US Wildlife Service. Within the "*Wetlands Analytical Mapping System*" they organized spatial data in so-called "geounits" corresponding to one of the United States Geological Survey (USGS) quadrangle scales, typically 1:24000 (Pywell and Niedzwiadek, 1980). This was years before the first web-map server implementations. The idea of tiled data is to split the data into equal areas called tiles and to only serve needed tiles. Data which is outside of a client's view is not so important or sometimes even might not be used by the user. This technique improved the usage of geodata over the web immensely (Gaffuri, 2012).

1.2 Vector tiles as an alternative to WMTS

An efficient alternative, which has the capabilities to replace WMTS as the currently most used service for web mapping, is the use of vector tiles (VT). This technology came up with Google which has been using VT since 2010 for their mobile version of Google Maps (Netlek et al., 2020). The general idea is to directly send tiled vector data rather than raster images. Currently, there is no fully OGC standard for VT. Vector tiles, however, become more and more popular in the GI community (OGC, 2018). At the moment, OGC evaluates VT and the initiative for the development of an open standard just passed "pilot phase 2" (OGC, 2021). This initiative includes a draft for an "OGC-API tiles" to serve maps or tiled feature data divided into individual tiles.

1.3 Scope

This short paper results as a part of research cooperation between the Carinthian University of Applied Science (CUAS) and Carinthia's governmental spatial data infrastructure (SDI) named KAGIS. Over the last year, KAGIS build up an infrastructure for offering vector tiles relying on Open Source software solutions. They also see a need in improving expertise in the field of VT. Part of this project is to investigate the advantages and disadvantages of VT for spatial data infrastructures based on literature research and test implementations using the vector tile environment of KAGIS focusing on the use case of web mapping and serving VT data. This short paper is focusing on the creation of VT in mvt format using standard PostGIS (PostgreSQL with spatial extension) functionality. For that, custom functions were implemented including the selection of relevant vector data out of the database, aspects of dynamic generalization/simplification of geometries and the transformation to Mapbox Vector Tile (mvt) format.

2 Methodology and means of implementation

Within this chapter selected previous studies and the concept for the implementation of custom PostGIS functions to serve VT data are described.

2.1 Previous work

Ingensand et al. (2016) and Martinelli and Roth (2016) are describing an approach where PostgreSQL functionality is used to serve data as vector tiles. Since version 2.4 of the spatial extension for PostgreSQL, named PostGIS, it is possible to create VT directly out of the database. Ingensand et al. (2016) are describing the first concepts and prototypical implementations of the swiss governmental SDI VT services hosted via geo.admin.ch. In their case study of 2016, they used TopoJSON as a data format, because of its good capabilities to store continuous polygons like zip-code areas. In the 2016 case study of Ingensand et al., they already mentioned that this mvt format might be better suitable than TopoJSON and worth investigating. The current swiss VT implementations are based on the mvt format, due to its support of different clients, file sizes and efficient encoding (geo.admin.ch, 2022).

Martinelli and Roth (2016) are describing another approach of using PostGIS for creating VT data during the project "OSM2Vectortiles". The motivation was to offer Open Street Map (OSM) data in VT format. The project resulted in the successor project "OpenMapTiles" which offers freely available OSM vector tiles based on the MapTiler API, which serves the tiles in Mapbox/Google Proto Buffer format with "pbf" file extension.

2.2 Standard PostGIS functionality

To serve data as mvt the PostGIS function "ST_AsMVT" can be used. Within the function the tiling, according to a regular tiling scheme, and encoding of relevant attributes is implemented. The parameters for the "ST_AsMVT" PostGIS function are as described in (PostGIS (a), 2022):

- row: Row data with at least a geometry column
- name: Name of the mvt layer
- extend: Tile extends in pixel (Default is square of 4096)
- geom_name: Name of the geometry column
- feature_id_name: Name of the column holding the unique identifier (Primary key)

The PostGIS documentation states, that for using the "ST_AsMVT" function the geometry column must be in tile grid coordinates, according to the Mapbox vector tile specification. For transforming the geographical coordinates into the relative mvt reference system the PostGIS function "ST_AsMVTGeom" can be used. This function transforms the coordinated as well as tries to provide the validity of the geometry. The parameters for the "ST_AsMVTGeom" PostGIS function are as described in (PostGIS (b), 2022):

- geom: Name of the geometry column
- bonds: Bounding box of the tile
- extent: Tile extends in pixel (Default is square of 4096)
- buffer: Buffer distance (pixel) to optionally clip geometries (Default 256)
- clip_geom: Boolean variable if geometry shall be clipped (Default: True)

2.4 Generalisation aspects

Gaffuri (2011) describes the need for generalization in web mapping and the benefits of vector data for implementing dynamic generalization. This dynamic graphic generalization should adapt the spatial data display to the user's needs (Gaffuri, 2011). Generalisation aspects within tiled vector data can be supported by (Ingensand et al., 2016):

- Preserving different levels of detail (LOD) geometries of the same objects for different zoom levels
- Dynamic simplification of geometries according to zoom level
- Selection of relevant vector objects for a certain zoom level

These techniques are a great way to decrease the file sizes of vector tiles. Within the implementation of the custom PostGIS functions to serve data in mvt format these aspects are implemented and tested.

3 Implementation of custom PostGIS functions to serve mvt data

This chapter is mainly focusing on the dynamic generalisation aspects within the function and the pre-processing of the VT. Other aspects, like input, output or the transformation to mvt coordinates using standard PostGIS functionality, are already explained in the chapter "Methodology and means of implementation".

3.1 Different LOD geometries of the same objects

Ingensand et al. (2016), is describing an approach of using different LOD geometries of the same objects for different zoom levels. According to this idea, different data layers for the VT map can be used. For small scales (zoomed out) a general overview layer with simplified and aggregated objects is used whereas for big scales (zoomed in) a more detailed layer is used. The usage of different data layers can be implemented in two different ways. Both are tested within the implementation. The first one is to provide two different layers with two different PostGIS functions for the VT. The generalized layer is included within the small scale (zoomed-out) tiles and the detailed layer within the big scale (zoomed-in) tiles, whereas no tiles include both layers. This can be implemented within the "WHERE" clause of the different PostGIS functions. For example, a general WHERE clause for the whole custom function like "WHERE z < 10" would result in an mvt layer only served for zoom levels smaller than 10.

The second option to implement different LOD layers for different zoom levels is to assign different PostGIS data tables to the same PostGIS function. Within the function, an "IF-ELSE" clause concerning the zoom level is controlling the data origin for the map layer. Within this approach, both PostGIS tables must share the same attributes and data types, which are finally served by the PostGIS function. A comparative illustration of both aspects can be seen in Figure 1.

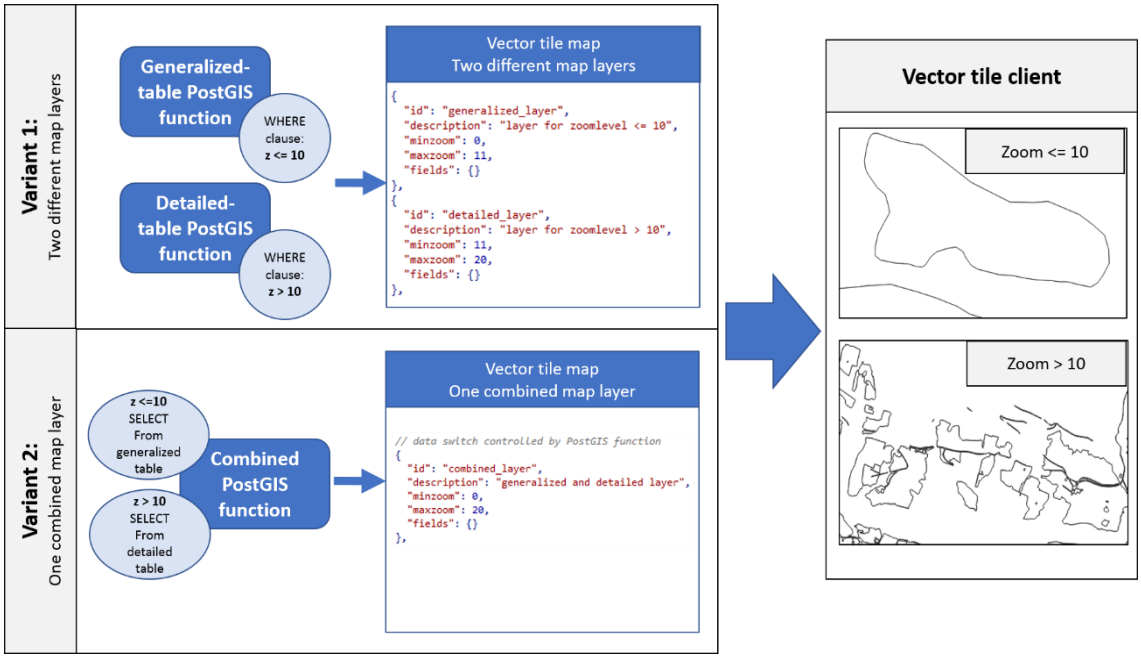

Figure 1. Comparative illustration of two implemented variants to use different LOD geometry tables of the same map topic. Top: two different layers with two different PostGIS functions for the VT map, Bottom: different PostGIS data tables assigned to the same PostGIS function.

Figure 2. Illustration of geometry simplification in relation to zoom level. Left: simplified polygons small zoom levels (z). Right: More detailed geometries for larger zoom levels (z).

3.2 Dynamic simplification

Another approach, mentioned by Ingensand et al. (2016), is the dynamic simplification of geometries according to the zoom level. Within the project's implementation, this concept is implemented with the use of the "ST_Simplify" PostGIS function (PostGIS (c), 2022) within the custom mvt PostGIS function. "ST_Simplify" uses the "Douglas-Peucker" algorithm to return a simplified version of the geometry of the input. The simplified geometry leads to smaller payloads of the VT. The function has the following parameters.

- Input geometry: Geometry originated from the PostGIS data table
- Tolerance: a factor influencing the level of simplification

The tolerance parameter is influencing the level of simplification. The higher the tolerance value the higher the simplification. Within the custom PostGIS function for serving mvt data, the zoom value can be used to dynamically manipulate the tolerance value and therefore dynamically simplify the geometries. Using the "St_Simplify" PostGIS function may not be suitable for every use case. Geometries can lose characteristic features and topology between objects may not be maintained (PostGIS (c), 2022). For the use case of web mapping and topological background map, this might not be the same relevance as for data used in analysis. Within the implementation, the relation of the zoom level on the tolerance value was selected for each layer individually, by visual observing the served VT data and visual comparisons of already existing raster tiled "role-model" maps. Another PostGIS function tested is the "ST_SimplifyPreserveTopology" function which should maintain the topological relations within the simplification process. The two functions resulted in no visual differences within the VT data. An illustration of the geometry simplification according to zoom level can be seen in Figure 2.

3.3 Selection of relevant vector objects

The last generalization aspect mentioned by Ingensand et al. (2016), is the selection of relevant vector objects for a certain zoom level. This aspect is implemented in two different ways within the WHERE clause of the developed custom PostGIS functions. The first is attribute driven, where a certain attribute is describing for which zoom levels the data shall be included. This can be a direct Integer value representing a zoom level or a class attribute. Predefined rules within the custom PostGIS function control which classes shall be included for a certain zoom level. The second way is geometry driven, where the area of a polygon, in relation to the zoom level, is controlling the selection. This method is only working for polygon layers. A representation of that can be seen in Figure 3. An example SQL example for an implemented custom PostGIS function can be seen in Appendix 1.

Figure 3. Selection of relevant vector objects for a certain zoom level in relation of the vector objects area. As the user zooms in more and more smaller buildings become visible.

4 Summary and discussion

This short paper describes key aspects of the implementation of custom PostGIS functions for serving data in mvt format. Within the custom functions standard PostGIS functionality is used to select relevant vector data out of the database, dynamically simplify geometries and transform the data to tiled mvt format. These custom functions serve as a starting point for further implementations of VT by KAGIS. With the use of a VT capable map server, like GeoServer, these custom PostGIS functions, in addition to a metadata description

based on Mapbox specifications, can be used to serve VT data for the purpose of high-performance web mapping. The aspects described in this short paper are part of further implementations of new web mapping technologies for Carinthia's governmental SDI. The custom PostGIS functions for creating tiled vector data shall be used for further implementations of KAGIS for the use case of high-performance web maps and serving geospatial data based on vector tile data in a more efficient way.

As described earlier other case studies, like Ingensand et al. (2016), are focusing on the usage of PostGIS functionality to serve VT data. For generalization, they used PostgreSQL/PostGIS functionality as well. The functionality was applied to the line and polygon objects to create vector data for different LOD. All different LOD layers were computed as new tables within PostgreSQL and are used for the different zoom levels within the VT tiling scheme whereas within this study the simplification is executed dynamically in relation to the current zoom level within the custom PostGIS functions.

A similar approach was implemented by Martinelli and Roth (2016) during the project "OSM2Vectortiles". All the objects of a layer are stored within one table. Because only a subset of data should be visible within a certain zoom level, SQL views were created, which shall function as a generalisation concept. These views filter the data according to object size (area or length) concerning the expected zoom level and were used to generate the tiles. As described in chapter 3.3 the selection of relevant objects according to zoom level is integrated within the custom PostGIS function to serve mvt data.

Both approaches are alternative ways to implement the generalisation aspects, for VT data within a PostGIS environment as described in this short paper. Nevertheless, these generalisation techniques can result in unwanted visual errors within a client. Li et al. (2017), describe such possible errors as gaps within lines/polygons or unclean rendered symbology. These gaps also can be seen within Figure 2. Therefore, a further evaluation of these unwanted errors is needed. This can be done by comparing the served mvt data with the original data to detect changes in positional/geometrical accuracy. As described earlier it always depends on the use case of the VT data. There are other requirements for the use case of web mapping and topological background map, as for data used in analysis. Furthermore, the implemented generalisation aspects are rather simple and are only focusing on the three techniques described by Ingensand et al. (2016). The implementation of more complex generalisation aspects, like displacement or exaggeration, require further investigations.

Acknowledgments

As already mentioned, the development of the concept of using PostGIS functionality and its implementation of it was executed within the research cooperation between CUAS and KAGIS and an external consultant named Wilhelm Berg. Wilhelm Berg is owner of the Austrian company: "BergWerk GIS EDV-Dienstleistungen e.U" (Berg, 2022). Hereby we would like to give thanks to Wilhelm Berg for the support in expertise and concept development.

References

Berg, W.: BergWerk GIS EDV-Dienstleistungen e.U. - Homepage, available at: https://www.bergwerk-gis.at/, last access: 5 April 2022, 2022.

Fu, P. and Sun, J.: Web GIS Principles and Applications, 1st edition, California: Esri Press, 2011.

Gaffuri, J.: Improving Web Mapping with Generalization, Cartographica: The International Journal for Geographic Information and Geovisualization, 46, 83-91, https://doi:10.3138/carto.46.2.83, 2011.

Gaffuri, J.: Toward Web Mapping with Vector Data, in: Geographic Information Science, GIScience 2012, edited by Xiao, N., Kwan, M., Goodchild, M.F. and Shekhar, S., Lecture Notes in Computer Science, vol 7478. Springer, Berlin, Heidelberg, https://doi.org/10.1007/978-3-642-33024-7_7, 2012.

geo.admin.ch: Vector Tiles Service: Verfügbare Dienste und Daten. Bundesamt für Landestopografie swisstopo, KOGIS (Koordination, Geoinformation und Services), available at: : https://www.geo.admin.ch/de/geo-dienstleistungen/geodienste/darstellungsdienste-webmapping-webgis-anwendungen/vector_tiles_service.html, last access: 5 April 2022, 2022.

Goodchild, M.F.: Tiling large geographical databases, in: Design and Implementation of Large Spatial Databases, SSD 1989, edited by Buchmann, A.P., Günther, O., Smith, T.R. and Wang, Y., Lecture Notes in Computer Science, vol 409. Springer, Berlin, Heidelberg, https://doi.org/10.1007/3-540-52208-5_25, 1990.

Ingensand, J., Nappez, M., Moullet, C., Gasser, L., Ertz, O. and Composto, S.: Implementation of Tiled Vector Services: A Case Study, SDW@ GIScience, 26-34, 2016

Li, L., Hu, W., Zhu, H., Li, Y. and Zhang, H.: Tiled vector data model for the geographical features of symbolized

maps, PLoS ONE, 12, 1–26, https://doi.org/10.1371/journal.pone.0176387, 2017.

Martinelli, L. and Roth, M.: Updatable Vector Tiles from OpenStreetMap, Bachelor thesis, HSR Hochschule für Technik Rapperswil, Rapperswil-Jona, Switzerland, 2016.

Netlek, R., Masopust, J., Pavlicek, F. and Vilem, P.: Performance Testing on Vector vs. Raster Map Tiles—Comparative Study on Load Metrics, ISPRS International Journal of Geo-Information, 9, 101-124, https://doi.org/10.3390/ijgi9020101, 2020.

OGC (a): Open Geospatial Consortium (OGC) - OpenGIS Web Map Tile Service Implementation Standard, available at https://www.ogc.org/standards/wmts, last access: 5 April 2022, 2010.

OGC (b): Open Geospatial Consortium (OGC) - OpenGIS Web Feature Service 2.0 Interface Standard, available at: https://www.ogc.org/standards/wfs, last access: 5 April 2022, 2010.

OGC: Open Geospatial Consortium (OGC) - Testbed-13: Vector Tiles Engineering Report, available at: https://docs.ogc.org/per/17-041.html, last access: 5 April 2022, 2018.

OGC: Open Geospatial Consortium (OGC) - Vector Tiles Pilot Phase 2, available at: https://www.ogc.org/projects/initiatives/vtp2, last access: 5 April 2022, 2021.

PostGIS (a): ST_AsMVT online documentation, available at: https://postgis.net/docs/ST_AsMVT.html, last access: 5 April 2022, 2022.

PostGIS (b): ST_AsMVTGeom online documentation, available at: https://postgis.net/docs/ST_AsMVTGeom.html, last access: 5 April 2022, 2022.

PostGIS (c): St_Simplify online documentation, available at: https://postgis.net/docs/ST_AsMVTGeom.html, last access: 5 April 2022, 2022.

Pywell, H.R. and Niedzwiadek, H.A.: The wetlands analytical mapping system: WAMS, USA, 1980.

Appendix 1

Exemplary SQL statement for creation of a custom PostGIS function serving data in mvt format.

```
CREATE OR REPLACE FUNCTION mvt.functionname(mvt_layer_name text,z integer,x integer,y integer,query_params json)
RETURNS bytea
      LANGUAGE 'plpgsql'

AS $BODY$
DECLARE
   mvt bytea;
      _bbox geometry:=ST_TileEnvelope(z, x, y);
      _zres float :=ZRes(z);
BEGIN
SELECT INTO mvt ST_AsMVT(tile, mvt_layer_name, 4096, 'geom', 'objectid')
FROM (
          SELECT
                  objectid
              , ST_AsMVTGeom(ST_Simplify(geom, _zres)
                   , _bbox
                   , 4096
                   , 8
                  , true
              ) AS geom
               , additional_attribute AS attribute_name
          FROM mvt.tablename
           WHERE
              geom && _bbox
                  and st_area(geom) > _zres
           ) as tile WHERE geom IS NOT NULL;
 RETURN mvt;
END
$BODY$;
```