Web client for PostGIS—the concept and implementation

Michal Kepka, Jan Ježek

Geomatics section, Department of Mathematics, Faculty of Applied Sciences,
University of West Bohemia, Univerzitní 8, 30614 Plzeň
{mkepka, jezekjan}@kma.zcu.cz

Abstract

PostgreSQL with PostGIS extension plays one of the major roles in many complex GIS frameworks. There exist many possibilities how to access such data storage, but most of them might be seen as not simple for new users. In this paper we would like to introduce the concept of the implementation of a web based PostGIS client application. The main emphasis of described solution is placed on simplicity and straightforward approach for visualisation of general SQL queries.

Introduction

Fast and online visualisation of complex SQL queries, that includes a spatial content without any scale limitation, is a challenge that can be hardly fulfilled with the software tools that are available nowadays. PostgreSQL with PostGIS plays a role of the flagship of Open Source RDBMS but there is just limited possibility of simple and fast queries visualisation. Widely used GIS applications offer certain solutions, but their approaches have some limits and can be seen as too complex.

We have designed the concept and implementation of software that will make output of SQL queries available through a web services. KML format was chosen as output data format as it is the most common format in the Internet environment. We have also designed a mechanism that helps to simplify the output data so that even large scale results can be easily cartographically visualised.

Server-side Java application with REST API has been implemented for that propose. The application accepts user's SQL query, executes it in the database and provide the HTTP service with results in KML format. Such KML is recursively generated according to bounding box of user request and provides relevant level of detail of particular data.

Benefits of the developed application are in a simple access and straight forward visualisation with the utility of SQL together with comprehensive list of spatial functions available in PostGIS. The developed application can be useful for data mining and analyses as well as for the education proposes in the field of the spatial databases.

Current state

Let us consider a situation that a vector data are stored in a spatial database. A user manages this database by an administration software tool, runs analysis in form of SQL query and receives results as ordinary table including the situation that geometry is the main result of the analysis. If a user wants to see geometry results in an appropriate map view there are several ways how to preview spatial data.

One way is connecting this database to an ordinary GIS application. We can mention one of very popular GIS applications QGIS. QGIS can connect to several types of databases and add a table with the geometry column as a new layer. Spatial features of this layer can be filtered by a search or by a query [1].

Second common way is publishing the data through a web service. There can be mentioned GeoServer [2] and MapServer [3] as examples of very popular server side software for serving spatial data through the web. A database can be connected to these products and then there is a possibility to publish particular data through standardised services (WMS, WFS [4]). Such a solution is very useful, but if we consider arbitrary queries, we will have to prepare specific configuration for each of those. GeoServer and MapServer support lots of standards for publishing spatial data e.g. WMS, WFS and WCS.

Another way is using some specialized desktop administration software for viewing and managing spatial data in the database. One of these is GeoRaptor, what is an extension for Oracle SQL Developer. GeoRaptor enables easy adding table as a layer into Spatial View where features are drawn in coordinate frame [5]. This solution provides an easy way how to preview spatial data directly in the administration tool where SQL queries are run, but usually doesn't enable adding any underlying map.

When we consider large volume of data we need also to use advanced visualisation algorithms into account (such as clustering and simplification [6]). Another possibility is to render data in advance to raster tiles and then provide the output in the raster format. Such a solution improves the data availability, but the drawback is in more complicated data update.

Software concept

Our use case can be described as follows. We have lots of large datasets stored in the PostGIS database and we need to run analyses and effectively publish large results of spatial queries. It means that we need an application that would satisfy given requirements – accepting queries in form of SQL and quickly displaying of results of spatial queries. The spatial results mean to be displayed directly in a map window or exported in some common GIS format (e.g. Shapefile, KML, GeoJSON) without any additional configuration. The main goals of the proposed solution are:

1. Analyses will be performed directly by database management system (DBMS).
2. Application will accept SQL.
3. Results of analyses with geometry columns will be produced in common GIS format or displayed directly in a map window.
4. Large data obtained from spatial queries will use convenient type of simplification to improve display performance.
5. Application will minimize number of steps on the process from analysis draft to display result.

If spatial data are stored in the database it is more convenient to access and process the data directly by SQL commands rather than by using third party software. Usually the SQL commands can be run in an administration client and query results are opened directly in a form of tables. Spatial databases mostly have conversion functions to convert the geometry data type to several GIS formats. Afterwards, the results can be exported from the database and can be opened in common GIS applications.

Several GIS formats were considered during design phase of the application. The GML format was taken into account because it is an OGC standard. GML format is very complex and is focused more on description of geographic objects than on their visualizing. ESRI Shapefile was excluded because it is a set of several files and it would bring more administration during files transfer. For the purpose of the application we have chosen the KML format as the output format for the geometry column. The reasons are:

- KML is an OGC standard for spatial data
- KML is widely supported by GIS application and very popular among amateur users at this time
- KML has an element called NetworkLink for bi-directional communication between server and client application

The developed application can use PostGIS function ST_asKML() to convert the geometry data type into the KML format [7]. Basically there are two concepts of such a KML output. First is a common static KML file with specific part of query result. This static KML file will contain geometry, other than geometry attributes and style how the geometry should be drawn. Second option is dynamic KML file that will contain description of data and NetworkLink element. The NetworkLink element enables bi-directional communication between a server and a client application [8]. Such an element contains URL of the server, the bounding box (BBOX) for the selection of appropriate part of the result that fits current map window and finally parameters describing conditions for data refreshing. An example of NetworkLink element with defined BBOX parameters of current map window is shown in Example 1.

The designed application can be divided into three main parts – database, server-side and thin web client. Design of the parts will be detailed described below.

Database part

The database part of designed application is the core part for storing and analysing of data. This part consists of database schema, data model and stored functions. The schema guarantees keeping tables with query results separate from other schemas with spatial data. Designed data model contains one metadata table, stored procedures and tables that contains query results. Fig. 1 shows designed metadata table schema and an example of one query result table.

Metadata table is designed to contain information of processed SQL query and values that facilitate selecting of the appropriate part of the result matching the map window which user is looking at.

Stored procedures control the launching of given SQL query and create new entry to metadata

Example 1: Source code of NetworkLink element

```
<NetworkLink>
   <name>Geometry Features</name>
   <visibility>1</visibility>
   <open>1</open>
   <description>Feature from database specified by SQL query</description>
   <refreshVisibility>0</refreshVisibility>
   <flyToView>0</flyToView>
   <Link>
      <href>http://whatstheplan.eu/analyst\_p4b/KMLServlet?
                                    queryId=1375356214822</href>
      <refreshInterval>2</refreshInterval>
      <viewRefreshMode>onStop</viewRefreshMode>
      <viewRefreshTime>1</viewRefreshTime>
      <viewFormat>BBOX=[bboxWest],[bboxSouth],
                                    [bboxEast],[bboxNorth]</viewFormat>
   </Link>
</NetworkLink>
```

table after finishing of SQL. Procedures hold processing of one SQL query as one transaction to keep consistency of the model. Main procedure returns identifier of query result to the server-side part of application. Another function enables update of SQL and finally there is a function that deletes result table and relevant entry in metadata table by given identifier of query result.

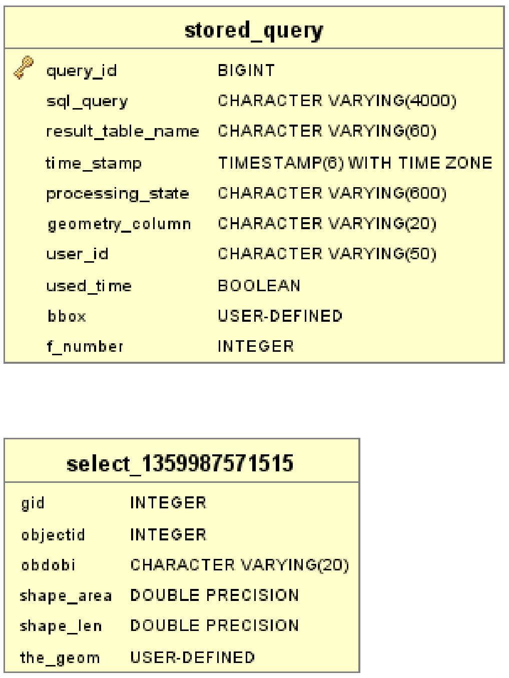
Server-side part

Server-side part of designed application manages and controls the whole application. This part can be further divided into several modules that are shown in Fig. 2 below.

There is a module called Connector that manages connections to the database with using connection pool mechanism. Module Receiver receives SQL queries from user interface, checks their correctness as SELECT query and transfers they to created function in the database through the Connector module. Receiver receives parameters of the map window to select appropriate part of result. SQL for data selection are compiled in SQL creator module. KML creator module receives from database the identifier of finished query result and returns it to the user interface. Result modeller module receives features of query result requested by user interface for visualization and prepares list of objects for KML output files. KML creator module is designed to use FreeMaker template (see [8] for FreeMarker details). The publishing module of the application has been designed according to the Representational State Transfer (REST) software architecture [9].

Graphical user interface

Graphical user interface (GUI) is another part of the designed application. This part keeps responsibility for communication with users, enables SQL queries insertion and simple result visualization in the map window or downloading results in form of KML files. GUI has been designed as a web thin client with basic functionality for SQL handling and result visualization.



stored_query	
query_id	BIGINT
sql_query	CHARACTER VARYING(4000)
result_table_name	CHARACTER VARYING(60)
time_stamp	TIMESTAMP(6) WITH TIME ZONE
processing_state	CHARACTER VARYING(600)
geometry_column	CHARACTER VARYING(20)
user_id	CHARACTER VARYING(50)
used_time	BOOLEAN
bbox	USER-DEFINED
f_number	INTEGER

select_1359987571515	
gid	INTEGER
objectid	INTEGER
obdobi	CHARACTER VARYING(20)
shape_area	DOUBLE PRECISION
shape_len	DOUBLE PRECISION
the_geom	USER-DEFINED

Figure 1: Metadata table with one query result table

Simplification algorithm

As has been already mentioned we take into consideration also the large datasets and large query results. The utilization of some simplification method for geometric parts of query results was considered already in first draft of the application. The query results could be simplified for an effective visualization after the analyses is made. It is necessary to select that kind of simplification methods that is not too much time-consuming and nevertheless can substantially facilitate visualization of results.

We can consider several methods for simplification on account of the datasets that have been available during the application development.

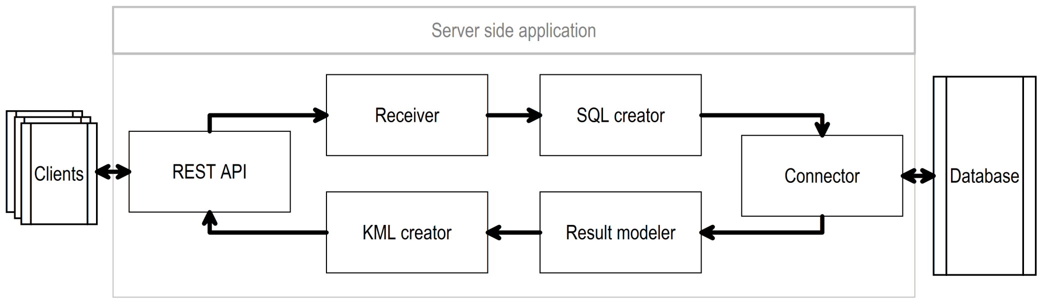


Figure 2: Designed schema of server-side part of application

k-means method

This method [10, 11] divides the set of data into predefined number of clusters. The clustering starts with definition of centroids. Each point is classified to the nearest centroid. New centroids are computed according to current shapes of clusters after classifying of all points. This procedure is iteratively repeated till reaching defined convergence criterion. Convergence criterion can be minimal change of clusters or minimal move of centroids between steps. The advantages of this method are their simplicity, short running time and the existence of k-means clustering module to PostgreSQL database [12]. The disadvantage is that solution depends on initial choice of first centroids. Several modifications of the basic method exist that refine diverse characteristics of basic method [13, 14]. Optimized k-means method already refines cluster centroids during classification of each point. K-medoids method selects existing points as clusters centroids that are closest to a precise centroid. Fuzzy k-means method determines degree of membership in clusters. Spherical k-means method differs from the previous method in method of cluster creation. This method starts with all points in one cluster. The first cluster is progressively divided into defined number of cluster.

Geohash method

Geohash is a geocode system that subdivides space into grid shape according to geographic coordinates [15, 16]. Geohash uses the region quad tree data structure and then assigns string codes to particular quads by using base32 character set. Length of the Geohash code defines precision of the encoding. The advantage of this method is arbitrary precision and unique identification of a cell. PostGIS contains method that returns Geohash representation of the geometry [4].

Modified Facility location algorithm

In accordance to the previous development, a clustering method developed at the Department of Computer Science at the University of West Bohemia was selected for the point datasets [11]. This method is based on a local search algorithm for the facility location but with several modifications. This clustering is not parameterized by a number of clusters as the original method is but by the facility cost. The higher facility cost faster reduces the amount of data and produces fewer clustering levels. Another modification is that this method stores all intermediate clusters and builds hierarchy of them. Method makes a single linear pass over the data and builds a hierarchy of clusters. Each cluster is represented by point nearest to centre of cluster, this point is called representative. The clustering is done according to given criteria. The basic criterion is the geometrical distance of points, but other point attributes can be used. If the number of representatives reaches defined count, representatives are processed and representative of the cluster of these representatives is selected. The clustering ends if all points are processed and all representatives on higher levels can't be further clustered.

Storing clusters in database

The selected clustering method saves created hierarchy of clusters in separate file for each level of hierarchy on hard disk. Because the developed application works with data stored in the database therefore it was necessary to design a way how to save the hierarchy to the database. Several possibilities were considered and the expected properties were satisfied properly by the modified preorder tree traversal algorithm [17]. The whole hierarchy is stored in one table with fixed number of columns. Only identifiers of points that refer to a query result in another standalone table are stored in this table. Table for the hierarchy contains the point identifier, the identifier of cluster representative, the level of hierarchy and the left and right value of index. The point indexing enables selection of the tree or a subtree in one query. Another benefit of this index is that the number of children of one node can be easy calculated from a difference of right and left value.

An example of the hierarchy is shown in Fig. 3 below with the direction of indexing. Fig. 3 shows the reason, why it is necessary to store identifier of point and parent together with the level number. It can be seen that the same point-parent pair could be found on several levels in the hierarchy.

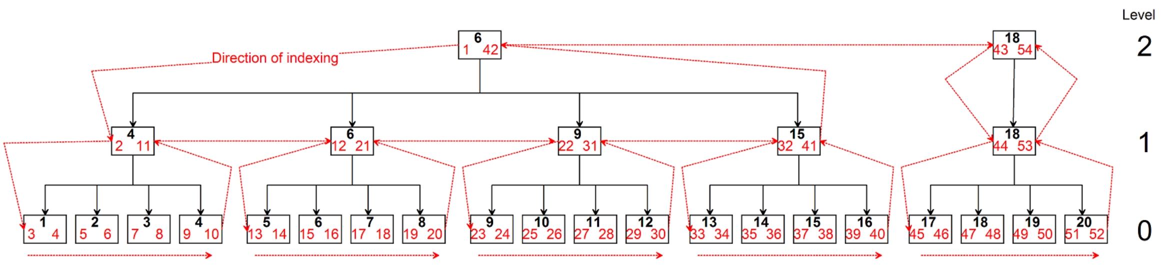


Figure 3: Example of cluster hierarchy

Table 1 shows storing of part of the hierarchy displayed in Fig. 3. Points from the table can be selected by several ways. The first way is to select points by defining the range of left and right value and alternatively with or without given level number. Another way is to utilize spatial functions from the database. Geometry of desired extent and appropriate hierarchy level is defined and points on given hierarchy level that intersect the extent geometry are selected.

The visualization of query result with available clustering is based on visualization of component levels of the created hierarchy of clusters. The level of hierarchy that will be visualized is selected according to the current extent of a map window in GIS viewer. The main problem of the cluster visualization was matching hierarchy levels with BBOX size of map windows. Because the number of levels of the hierarchy is derived from the number of points in dataset,

Table 1: Cluster hierarchy stored in table – example

ID_POINT	**ID_PARENT**	**LEVEL_POINT**	**LEFT_INDEX**	**RIGHT_INDEX**
1	4	0	3	4
⋮	⋮	⋮	⋮	⋮
6	6	0	15	16
⋮	⋮	⋮	⋮	⋮
6	6	1	12	21
⋮	⋮	⋮	⋮	⋮
6		2	1	42

it was necessary to select the matching way in addiction to spatial extent of dataset. The idea is based on matching the highest hierarchy level with maximum BBOX of the result dataset. Lower hierarchy levels are subsequently matched with smaller map window BBOX sizes. The matching is based on the size of ratio of current map window extent to maximum extent of dataset. That means that the smaller the value of the ratio is, the lower the level of the hierarchy is selected.

In standard case, the data for visualization are selected if they intersect current map window BBOX. Together with simplification, suitable level of the hierarchy is selected first and then the data on given level that intersect current map window BBOX are selected. This method further reduces the amount of data that user application receives. The disadvantage is the necessity of creation of the hierarchy over the query results.

Experiments and results

The new client application has been designed and then developed as a Java application called PostMap. The PostMap has been developed according to designed structure that was described previously. Basic modules three modules of PostMap are shown in Fig. 4.

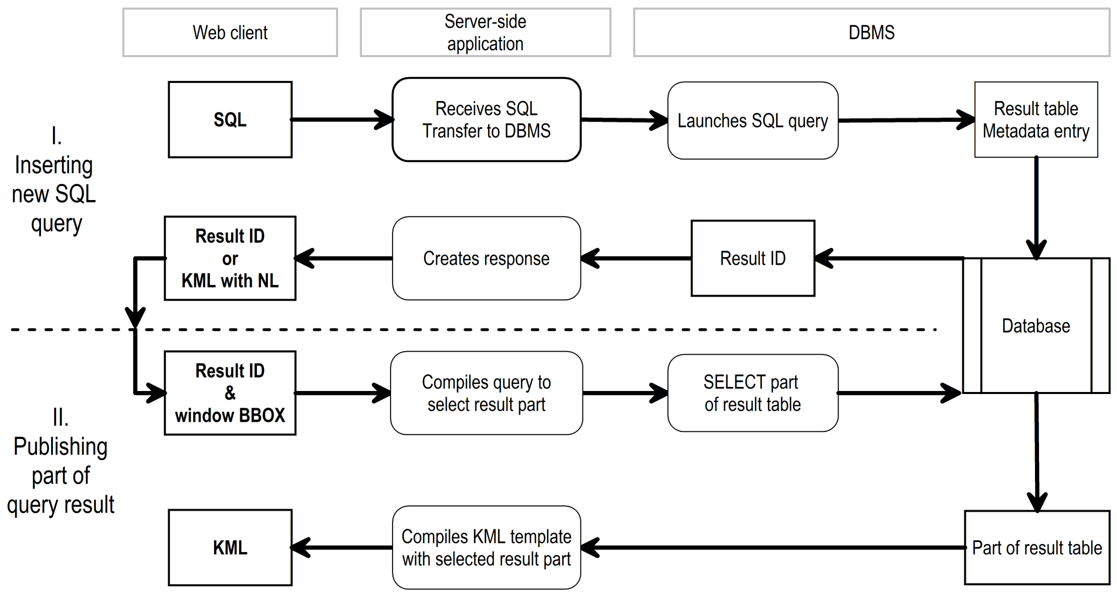

Web client
Server-side application
DBMS
I. Inserting new SQL query
SQL
Receives SQL Transfer to DBMS
Launches SQL query
Result table Metadata entry
Result ID or KML with NL
Creates response
Result ID
Database
II. Publishing part of query result
Result ID & window BBOX
Compiles query to select result part
SELECT part of result table
KML
Compiles KML template with selected result part
Part of result table

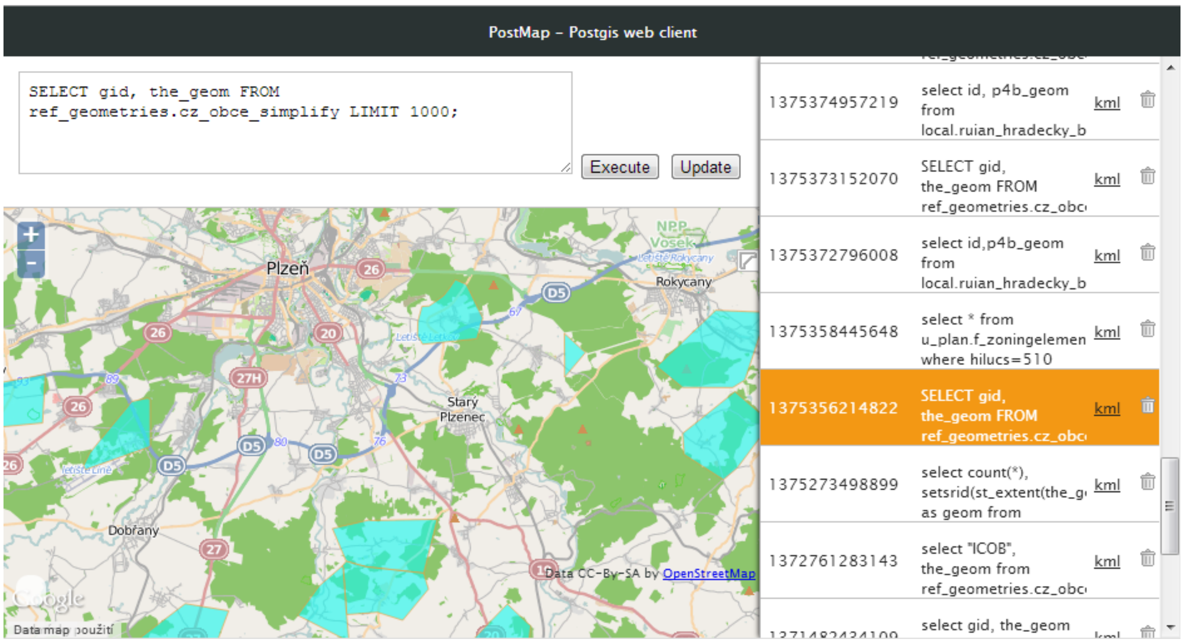
Figure 4: Schema of the application

First step represents running the analysis where SQL query is on the beginning and an identifier of the result is at the end in Web client. Second step stands for preview of the result where the identifier of the result and BBOX of current map window are on the start of process and KML file with part of result data intersecting current map window is at the end.

Graphical visualization of the query results was implemented by two methods. Both of implemented methods have certain advantages and disadvantages. First method is direct visualization of entire query result content in the map window of web graphical user interface (GUI). Second method is based on publishing results in KML files and using another GIS viewer as visualization client.

Graphical User Interface

The first part of PostMap which user meets is a web graphical user interface (GUI). The web GUI is a Javascript thin client in a form of web page that contains functional sections with client-server communication. The web GUI enables inserting and editing of SQL queries, contains list of last 5 user queries and a map window. After finishing of analysis, the content of query result is directly displayed in the map window. According to finished rendition of query result user can edit previous SQL query to reach intended result. Fig. 5 shows web client with opened query result.



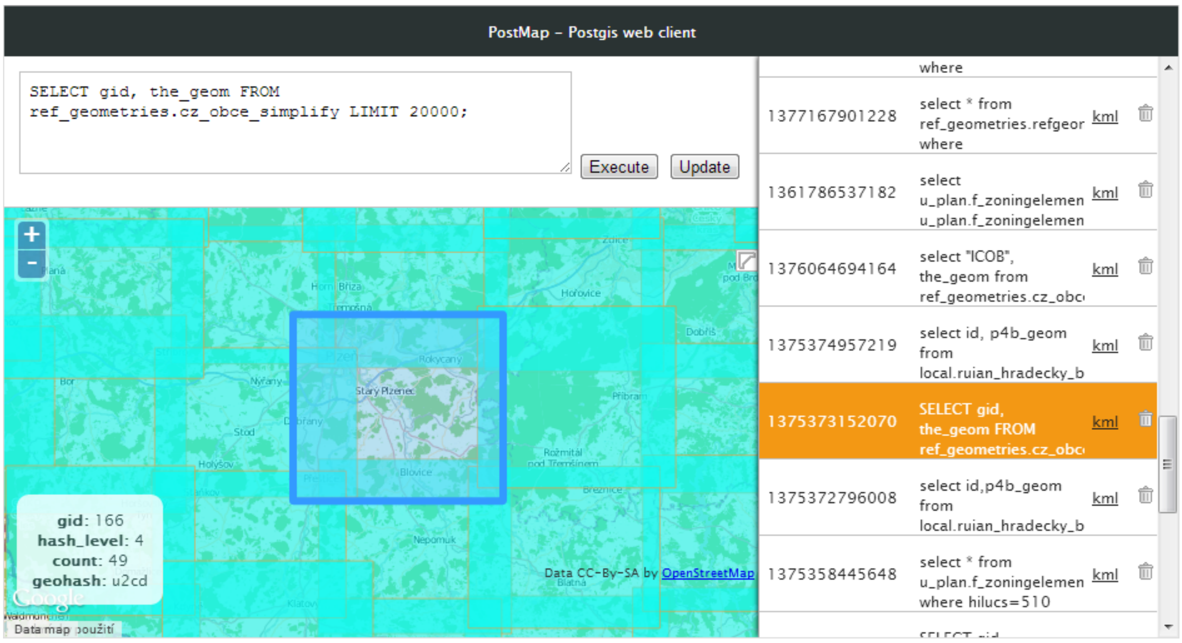
```
SELECT gid, the_geom FROM
ref_geometries.cz_obce_simplify LIMIT 1000;
```

1375374957219	select id, p4b_geom from local.ruian_hradecky_b	kml
1375373152070	SELECT gid, the_geom FROM ref_geometries.cz_obc	kml
1375372796008	select id,p4b_geom from local.ruian_hradecky_b	kml
1375358445648	select * from u_plan.f_zoningelemen where hilucs=510	kml
1375356214822	SELECT gid, the_geom FROM ref_geometries.cz_obc	kml
1375273498899	select count(*), setsrid(st_extent(the_g as geom from	kml
1372761283143	select "ICOB", the_geom from ref_geometries.cz_obc	kml
1371482434109	select gid, the_geom	kml

Figure 5: Web client with query result

Another simplification method was implemented to current version of PostMap. Geohash method is using for large results where is necessary an overview of features distribution rather than precision of features position. In cases of important position is used clustering method described above. An example of simplified result by Geohash method is shown on Fig. 6.

The second visualization way is based on publishing query results in KML format. The query result is exported from the application as dynamic KML file with the NetworkLink element after finishing of given query. In this case, user can open output file in any compatible GIS viewer according to his needs (e.g. GoogleEarth). Prerequisite for correct dynamic visualization is supporting of KML format with NetworkLink element in the selected GIS viewer. This support is necessary for communication between the server part and the third party GIS viewer. The KML file with NetworkLink is opened in a selected GIS viewer, GIS viewer starts to communicate with the server application and the server application starts to send requested part of query result. The GIS viewer requests part of a query result by sending the extent of current map window. The advantage of this visualization is the minimization



```
SELECT gid, the_geom FROM
ref_geometries.cz_obce_simplify LIMIT 20000;
```

gid: 166
hash_level: 4
count: 49
geohash: u2cd

	where		
1377167901228	select * from ref_geometries.refgeor where	kml	🗑
1361786537182	select u_plan.f_zoningelemen u_plan.f_zoningelemen	kml	🗑
1376064694164	select "ICOB", the_geom from ref_geometries.cz_obc	kml	🗑
1375374957219	select id, p4b_geom from local.ruian_hradecky_b	kml	🗑
1375373152070	SELECT gid, the_geom FROM ref_geometries.cz_obc	kml	🗑
1375372796008	select id,p4b_geom from local.ruian_hradecky_b	kml	🗑
1375358445648	select * from u_plan.f_zoningelemen where hilucs=510	kml	🗑

Figure 6: Web client with simplified query result

of data amount that is being used in GIS viewer at certain moment. The disadvantages are necessity of repeated transfer of small data amount over net and using of third-party GIS viewers to displaying query results. An example of visualization clustered features of query results through dynamic KML file is shown in Fig. 7.

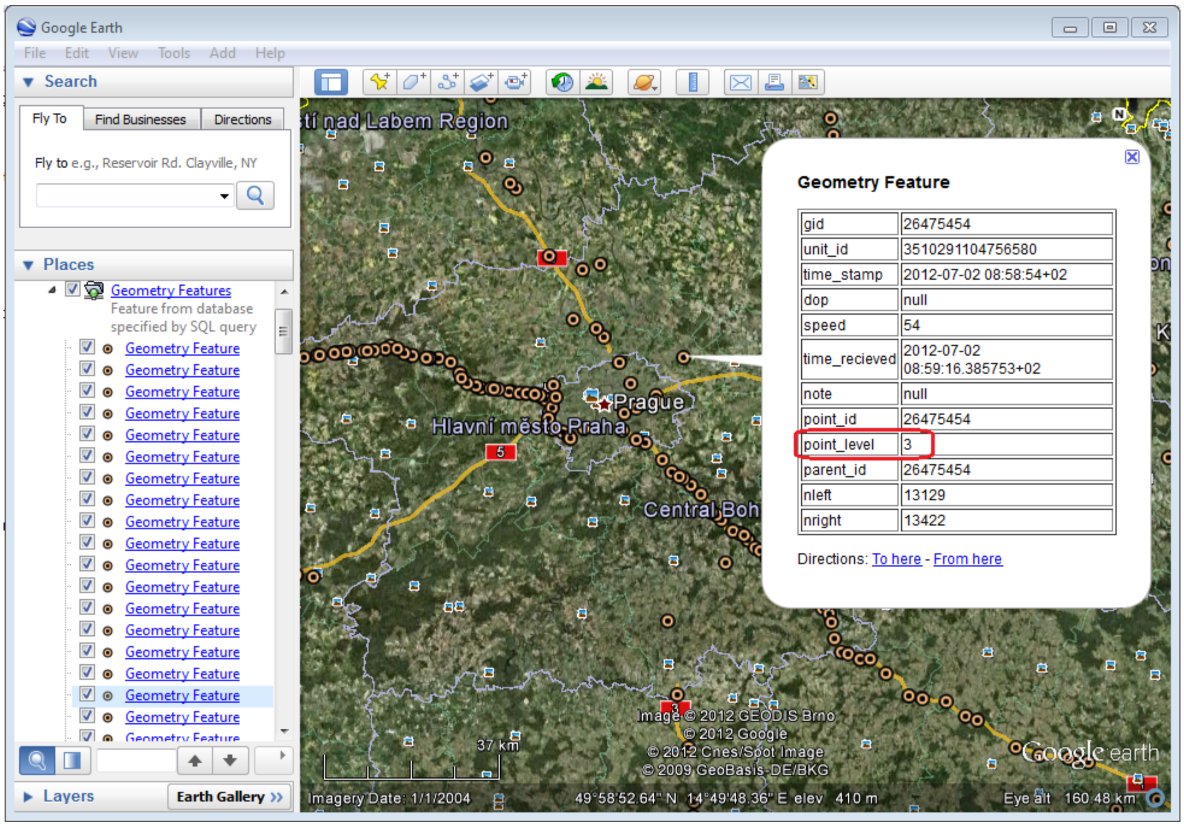
REST API

The current version of PostMap uses REST API for two tasks. The first task is the management of the user queries. There are implemented several URL for inserting, updating, deleting and listing of user queries. Responses of these requests depends on used HTTP methods, but mostly are returned responses in JSON format or only response codes. The second task is the management of stored queries metadata. This group of URLs returns description of stored user queries listed by several parameters. The main advantages of these RESTful services are the easier way how to add new functions to the PostMap and the easier dealing with URL requests. PostMap can be uses as a middleware to another application by accessing services over REST API.

Discussion

PostMap and its user interface enables to perform SQL queries based on the SELECT easily. Unlike current solutions there is no need for any configuration of data visualisation and results can be visualised on to a map. On other side we have identified these limitations:

- The SQL results have to be stored as temporal tables. Update of particular result essentially means to delete and create new temporal table again.
- There exists the risk of SQL injection what is a security issue. Such an issue can be



Geometry Feature

gid	26475454
unit_id	3510291104756580
time_stamp	2012-07-02 08:58:54+02
dop	null
speed	54
time_recieved	2012-07-02 08:59:16.385753+02
note	null
point_id	26475454
point_level	3
parent_id	26475454
nleft	13129
nright	13422

Directions: To here - From here

Figure 7: Web client with simplified query result

solved by set-up of authentication and authorization layer on the level of REST services.

- There is no possibility to apply cartographic visualisation of certain attribute in the form of cartograms. Such a problem can be solved in the future through the collection of templates for KML style.

- In case of using KML file with Networklink, geometry features are loaded only according to the current BBOX regardless of previously loaded data, although BBOXs could be partially overlapping with the one that were loaded before. Such an issue can be solved for example by storing previously published geometry features, comparison new BBOX to the previous and sending only the missing geometry features.

Conclusion

The main usage of the PostMap can be found in two fields. The first field are technical projects where can be used as basic software tool for user spatial analyses and their visualization. The PostMap stands for a middleware that connects database with stored data and functions with visualization module e.g. a web geoportal or another standalone application. The PostMap consumes regular SQL SELECT queries that can contain spatial functions and produces. The input SQL queries can be predefined and users fill up only several parameters. The outputs are represented by query results exported in form of common KML files or visualized in own web client directly. We suppose that the query result contains exactly one geometry attribute.

The use of the KML format enables the export of whole query result or performs the dynamic visualization.

The second field of use is education of general GIS subjects or subjects focused on spatial databases. PostMap can be used to introduce spatial functions and predicates. It is possible to demonstrate use of functions and predicates into SQL queries and to demonstrate the connection of SQL queries outputs to parameters change. The contribution of this application consists in very quick and easy web visualization of query results without using any other clients.

Acknowledgement

Author Michal Kepka was supported by the European Regional Development Fund (ERDF), project "NTIS – New Technologies for the Information Society", European Centre of Excellence, CZ.1.05/1.1.00/02.0090.

Author Jan Ježek was supported by the project EXLIZ – CZ.1.07/2.3.00/30.0013, which is co-financed by the European Social Fund and the state budget of the Czech Republic.

References

[1] QGIS User Guide. Available: http://www.qgis.org/en/docs/user_manual/index.html

[2] GeoServer Documentation: User Manual. GeoServer [online]. Release 2.4.1. 2013. Available: http://docs.geoserver.org/

[3] MapServer Documentation: MapServer. [online]. Release 6.4.0. 2013. Available: http://mapserver.org/documentation.html

[4] OGC Standards. Open Geospatial Consortium. 2013. [online]. Available: http://www.opengeospatial.org/standards/is

[5] GeoRaptor Documentation. GeoRaptor [online]. Release 3.0.1. 2011. Available: http://sourceforge.net/apps/mediawiki/georaptor/index.php?title=Main_Page

[6] LI, Zhilin. Algorithmic foundation of multi-scale spatial representation. Boca Raton, FL: CRC Press, c2007, 280 p. ISBN 978-084-9390-722.

[7] Google Inc. KML Documentation. 2012. [online]. Available: https://developers.google.com/kml/documentation/

[8] Refractions Research Inc. PostGIS 1.5 Manual. SVN Revision (11757). 2012. [online]. Available: http://postgis.net/docs/manual-1.5/index.html

[9] Visigoth Software Society. FreeMarker Manual. Version 2.3.20. 2013. [online]. Available: http://freemarker.org/

[10] Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Chapter 5: Representational State Transfer (REST). Doctoral dissertation, University of California, Irvine, 2000. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[11] MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. In Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability Vol. 1. University of California Press. 1967. pp. 281–297. Available: http://projecteuclid.org/euclid.bsmsp/1200512992

[12] Hartigan, J. A. and Wong, M. A. Algorithm AS 136: A K-Means Clustering Algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), Vol. 28, No. 1 1979. pp. 100-108. Available: http://www.jstor.org/stable/2346830

[13] Harada Hitoshi. Kmeans 1.1.0. Plugin for PostgreSQL. Release 22. 7. 2011. [online]. Available: http://pgxn.org/dist/kmeans/

[14] Skála, Jiří. Algorithms for manipulating large geometric data. Ph.D. dissertation. Fakulta aplikovaných věd, Západočeská univerzita v Plzni, Plzeň. 2012.

[15] Marvalová, Jindra. Správa a vizualizace časoprostorových bodových dat. Bachelor thesis. Fakulta aplikovaných věd, Západočeská univerzita v Plzni, Plzeň. 2013.

[16] Niemeyer, Gustavo. Geohash. [online]. Available: http://geohash.org/

[17] GNIEMEYER. Geohash. In: Wikipedia: the free encyclopedia. [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-09-19]. Available: http://en.wikipedia.org/wiki/Geohash

[18] Van Tulder, Gijs. Storing Hierarchical Data in a Database. Sitepoint. 30.4.2013. [online]. Available: http://www.sitepoint.com/hierarchical-data-database/