INTEGRATION OF URBAN SPATIAL DATA MANAGEMENT AND VISUALIZATION WITH ENTERPRISE APPLICATIONS USING OPEN-SOURCE SOFTWARE

H. Ostadabbas[1], H. Merz[2], H. Weippert[3]

[1] die STEG GmbH, Olgastraße 54, 70182 Stuttgart, Germany – hamidreza.ostadabbas@steg.de
[2] cyberconcepts IT-Consulting, Am Steinigen Graben 8, 86911 Dießen, Germany – h.merz@cyberconcepts.de
[3] Koch Immobilienbewertung GmbH, 73730 Esslingen am Neckar, Germany – heike.weippert@dr-koch-immo.de

KEY WORDS: QGIS, PostgreSQL, PostGIS, EAI, ALKIS data, SQL, Python, Lizmap

ABSTRACT:
In recent years, efficient management of urban spatial data has played a major role in improving urban planning projects both in terms of cost and time savings. Since urban planning projects involve various disciplines like city planning and architecture as well as working with different spatial data, one of the main challenges is how to integrate and manage these multimodal data for a proper workflow. Currently, the involved companies are using project management and accounting systems, so called Enterprise Resource Planning (ERP) systems to handle these complex urban projects - that partly handle the same data objects as stored in urban spatial databases but without any spatial reference. Embedded in the application example of an urban redevelopment area, which according to the German Urban Development Promotion Act aims at financially promoting urban districts in need of renewal, project-related spatial and non-spatial data that were previously kept separate are linked and integrated. Therefore, our work presented here bridges the gap between these two types of application systems, the non-spatial accounting system called Finanz Management System (FMS) and the urban spatial databases. FMS manages information related to parcels, buildings, property owners, as well as the legally required payments connected to urban development, while an urban spatial database manages the geodata. We describe the prerequisites, procedures, and software development steps for coupling different types of applications by providing an example of the Enterprise Application Integration System (EAI). Our innovative integration process aims at making information from the spatial database available in FMS and vice versa, and allows updating the corresponding databases. Our work shows the potential of open-source software for cadastral data processing and visualization as well as accounting procedures for urban planning projects.

1. INTRODUTION

1.1. Technical Background

In 1971, the Urban Development Promotion Law (Städtebauförderungsgesetz StBauFG) was passed in Germany. Until now, more than five thousand urban redevelopment areas were designated with the aim to maintain, modernize, and revitalize the urban structure and to improve the living conditions for their inhabitants (BauGB, 2017; Ostadabbas et al., 2020). The ongoing urban redevelopment process can take place more than a decade and need to be closely monitored, both financially and spatially regarding the complex redevelopment measures ranging from building demolition and construction to new transportation infrastructure to be build. For the monetary aspects accounting systems like the FMS were developed to fulfill the specific needs of statutory funding. For the surveying aspects, geo spatial databases handle the changes of the urban structure.

To combine and integrate these two mentioned aspects is, concerning the whole procedure of managing an urban redevelopment area, a unique German task and legal invention. Therefore a specific case comparison is hard to be carried out and will not meet the application task to be aimed at. The following sections will explain firstly the two systems, their content and input data separately and secondly in 1.4 the need, the benefit and the innovation of such application integration will be pointed out.

1.2. Urban Spatial Database

The content of a spatial database consists of space information that is necessary for applications, where there is need to monitor the dynamic spatial position of an object or event. Therefore, spatial databases describe the fundamental representation of objects derived from spatial or geographic entities. They support aspects of space and offer spatial data types in its data model and query language (Samson et al, 2017).

In the following work a spatial database (PostgreSQL with the extension PostGIS) was set up especially for urban planning and development applications. PostGIS is a spatial database extender for the PostgreSQL object-relational database, which contain spatial information and follows the Simple Features for SQL Specifications of the Open Geospatial Consortium (OGC) (Obe et al, 2011, PostGIS, 2021).The spatial objects stored in the database represent all aspects to document and monitor changes in the urban structure. Corresponding objects are mainly urban quarters, parcels, buildings and the infrastructure network. Additionally many object specific attribute like e.g. the usage of a building are stored in the database.

To guarantee the complexity of urban structure changes the database allows enhancing information towards each spatial object. Official land cadastral data (ALKIS) present the main input dataset for the urban spatial database. ALKIS®, established in all federal states of Germany in 2015, combines and integrates data of the former cadastral map as well as the former property registry. In ALKIS®, for the first time, spatial

and non-spatial related data were kept together systematically redundancies-free. The consortium of the Surveying Authorities of the States of the Federal Republic of Germany (AdV) developed a functional design to manage all basic geo data of the official surveying and mapping. All federal states agreed upon a unique data model using the UML standard (AdV, 2009, Seifert 2005). The conceptual design of the ALKIS® data defines different kinds of relations between the object types encoded in GML format (Lake et al. 2004, Portele 2007) and has been re-modeled in this project for the needs of the accounting procedure (Ostadabbas et al.,2019).

1.3. FMS Accounting Application

The FMS (Finanz Management System) application is an accounting and project management application used for managing urban redevelopment projects. It provides services for cities and smaller communities for managing statutory funded projects, recording all corresponding expenses and earnings and preparing funding requests.

The core objects managed by the application system and stored in the database are individual funding tasks that are usually associated with real estate, e.g. selling or buying properties, demolition, construction, or refurbishment works, and corresponding planning activities. While these tasks are closely related to information managed in spatial data management systems the FMS application does only provide a static image export for a specific task object but not an automatic map generation.

The FMS application is browser-based, thus allowing easy access via the web without the need for installing special client software. It is written in Python and builds upon a web application framework (Zope, 2021; Zope toolkit, 2021) and the knowledge management framework loops (cyberconcepts loops, 2021). The primary storage of FMS is an object oriented database (ZODB, 2020) but it provides also access to SQL databases for data storage and retrieval. As the standard SQL database used for FMS is PostgreSQL FMS can readily access spatial data stored in a PostgreSQL database (which has the PostGIS extension installed).

1.4. Application Integration Tasks

Initially the FMS accounting application and the spatial database systems were strictly separated, in spite of considerable overlap of data stored in both systems. Thus, for example, information about building usage, quality and state of buildings, type and state of the refurbishment tasks were stored in both systems and had to be edited twice. This not only leads to additional work but presents also a source of potential data inconsistencies between the systems and delayed updates.

As the FMS does not contain any geometric information the maps for each property or each redevelopment area have been created manually in BricsCAD (commercial software) and uploaded as a static image export in FMS. If there are changes in the data, these map images have to be recreated and uploaded again.

It is necessary to generate maps on the fly in FMS, showing the current state of the geometries such as land parcels and buildings and their attributional information visualized by corresponding layout symbols.

Therefore the main objectives of the EAI (Enterprise Application Integration) work are:

- provide spatial representations on the fly in FMS,
- show relevant information simultaneously in both systems with updated and consistent data,
- allow automatic update of relevant information in both directions.

As FMS can be easily extended and adopted to specific needs, the most feasible solution was to access the spatial database directly from FMS. The main prerequisite for this kind of connection is a set of strictly standardized tables in the spatial database which in the following will be defined as *generic representation*, (see section 1.5). This step was challenging as the database tables for the projects are initially created independently via QGIS layers in separate database schemas. Thus it was necessary to provide an easy and safe way to fill this *generic representation* with data from each project schema.

The *generic representation* of spatial data and related information can then additionally be used for displaying dynamically generated maps in the web browser using QGIS server together with the Lizmap rendering engine. As FMS is a browser-based application these maps can also be included in the appropriate pages (e.g. those related to redevelopment tasks) in FMS.

1.5. Generic Representation for Spatial Data and Object Attributes

For unified access by application programs and the display of maps that represent a variety of properties of spatial data objects a *generic representation* for all data with a simple set of tables was created. This structure allows the representation of all kinds of data for all cities/communities and projects: spatial data (different types of geometries: parcels, buildings, other areas, lines, points; see table *geoitems*) and an open set of additional attributes. For use cases where additional attributes must be included, the separate table for storing the attributes of the geometric object (see table *geoattrs*) allows a full SQL definition of the *generic representation* (see Figure 1) (cyberconcepts geostore, 2021).

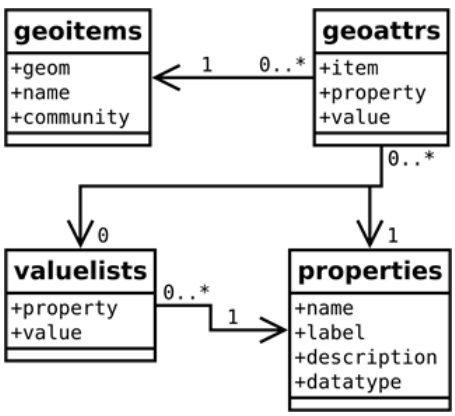

Figure 1: Simplified table structure of the *generic representation*

The table properties contains metadata related to the attributes stored in *geoattrs*; only attributes with a corresponding entry in the properties table can be stored in the *geoattrs* table.

As there are a lot of properties that have a fixed set of possible values (usually short texts, e.g. for building usage or quality, or project state) an additional table *valuelists* that contains predefined lists of values for these properties was defined. In this case the value column of the properties table contains only the identifier of the value from the *valuelists* table.

Inserting attributes in a table separate from the objects themselves provides maximum flexibility: new properties or additional values can be introduced any time without any changes to the existing table structure, so no tables or columns will have to be added to the *generic representation*, yielding high stability for all kinds of access.

On the other hand the distribution of related data across multiple tables makes the accessing of data and especially the displaying of maps with tools like QGIS more complicated. This complexity is addressed by a database view *vgeoattrs* that provides a full list of all spatial objects with their attributes, taken from the *valuelists* table when necessary (see Figure 2), thus the access by QGIS is greatly simplified (client and server).

```
create or replace view vgeoattrs as
  select i.id as item_id, i.geom, i.type, i.name as item_name,
         i.description as item_description, i.community,
         a.id as attr_id, p.id as prop_id,
         p.name as prop_name, p.label as prop_label,
         p.description as prop_description, p.datatype,
         p.domain as prop_domain,
         case when p.datatype = 25 then vl.value
              else a.txtvalue
         end as txtvalue,
         a.intvalue
    from geoitems i
      left join geoattrs a on a.item = i.id
      left join properties p on p.id = a.property
      left join valuelists vl on
        p.datatype = 25 and vl.property = p.id and vl.id = a.intvalue;
```

Figure 2: SQL code for creating the view *vgeoattrs* to allow easy access to the *generic representation*

Based on this view, QGIS layers can be easily created by using the filter functionality of QGIS.

2. METHODOLOGY

2.1. Integration Workflow

The workflow for each project (usually a city or community) to be managed by the system basically consists of three steps (see Figure 1):

- Initial set up of an *import representation* using cadastral data from ALKIS data and field data taken from Excel sheets or imported from spatial data collection tools like QField
- Load the *generic representation* from the database schema set up in the first step using SQL scripts, typically run via the PosgreSQL command line tool *pgsql*
- Access the *generic representation* from QGIS, the web interface (using QGIS Server and Lizmap), and the FMS accounting application

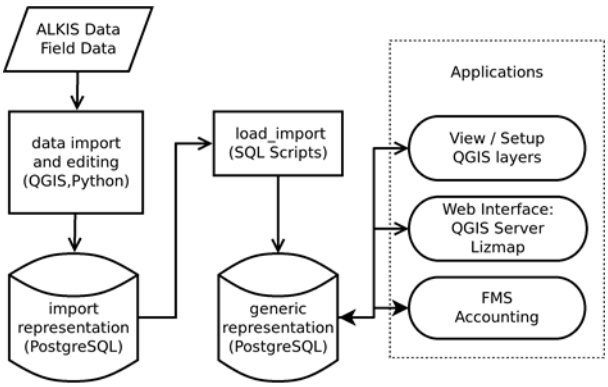

Figure 3: Components of the integration workflow

For each city/community the basic geometries of parcels, buildings and streets are set up and stored in the PostgreSQL database using the corresponding ALKIS dataset. Additional data are supplied using QGIS (see also section 2.2).

The *import representation* corresponds to one schema per project /community that contains a fixed set of tables for different types of objects (parcels, buildings, other areas, lines, points) with columns for all properties of the respective type. These tables are created in each schema via a predefined SQL script, so that in all database schemas there is the same set of tables with the same set of table columns. If necessary, additional columns could be added to the tables to store additional properties.

For the visualization of spatial data and related information in FMS one needs a strongly standardized data structure, the *generic representation*, with only one database schema (generic) that contains all relevant data from all communities and projects (see section 1.5 for a detailed description). This *generic representation* has to be filled (for each community or each project) via an SQL script (load_import.sql) that collects data from the tables and columns of an import representation and updates the *geoitems* and *geoattrs* tables in the *generic representation*.

The FMS application can thus retrieve and display data from the *generic representation* by direct access to the spatial database; maps created on the fly via the web interface can easily be embedded in the data pages of FMS.

2.2. Python Script for Map and Project Configuration

According to the integration workflow described in the methodology chapter, Python scripting will be applied for ALKIS data import and editing. Moreover, it will be utilized for viewing and setting up the QGIS layers. Starting from 0.9 release, QGIS supports Python language. It was chosen for the software development because it is one of the most common programming languages (PyQGIS Developer Cookbook, 2020). PyQGIS is a Python environment inside QGIS with a set of QGIS libraries, including Python tools with enables access to libraries such as Pandas, Numpy, and Scikit-learn (Montaya, 2017).

The Python coding in the project mainly focuses on configuring attribute forms for widget types of different fields that had been created in PostgreSQL (data import and editing). This configuration for instance is compulsory for image acquisition on-site (see Figure 4).

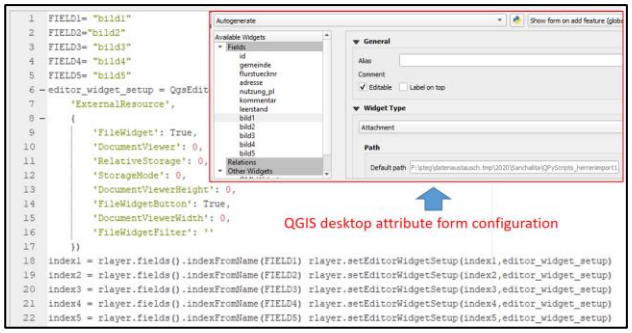

Figure 4: Python script for attribute form configuration for widget type "attachment".

According to predefined attributes for urban planning purposes in the *import representation* schema, such as e.g. usage of a building, year of construction, heat demand etc., the capability of choosing these different attributes has to be activated. Figure 5 shows the configuration result and feature attribute by running the python code.

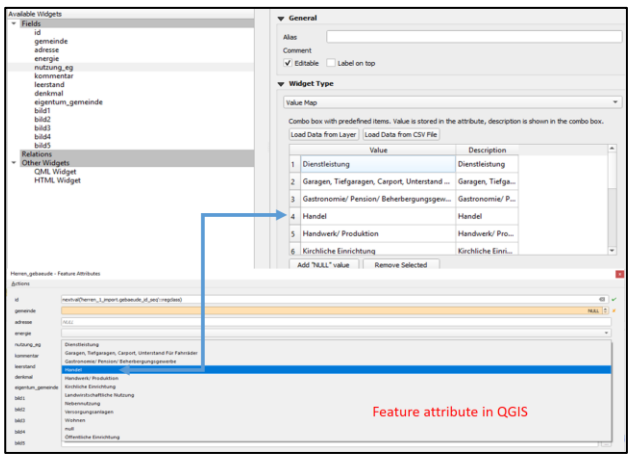

Figure 5: Widget type configuration in attribute form for feature attribute in QGIS desktop.

In the second phase, Python programming is used for the visualization of the QGIS layers. By creating a styling button as a new icon in QGIS desktop, the process of assigning predefined symbols and different layout styles is much faster and easier. Furthermore it allows the accessing of difficult symbols which have been created in drawing software such as BricsCAD and Illustrator (see Figure 6).

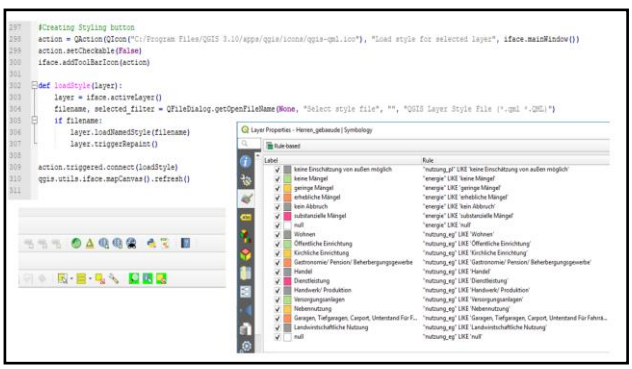

Figure 6: Python script for style configuration.

The mentioned Python scripts will be updated and configured for all the important layers for the *import representation* schema in the PostgreSQL database.

2.3. Web Presentation with QGIS Server and Lizmap

As requested by municipalities and internal by the involved company itself the maps created in QGIS desktop should be accessible for the public in the web. Therefore, QGIS server which employs QGIS as a backend for GIS functions and map rendering, and Qt library will be used for graphical interface. "Qt is a free and open-source widget toolkit for creating graphical user interface which runs on different software and hardware platforms". (Qt – About US, 2020).

By using the same graphical library (Qt) in QGIS desktop and QGIS server, the maps that are created in the desktop will have the same appearance in the web. (QGIS Documentation, 2020). Lizmap as a web client is the open source software that is founded by a French company "3Liz".

As a requirement on the client side, Lizmap is installed in QGIS desktop as a plugin. It provides the possibility to add tools needed in the web map user interface (see figure 7). On the server side the visualization is based on the JavaScript library OpenLayers which handles the WMS requests to the QGIS server (Ostadabbas et al., 2019).

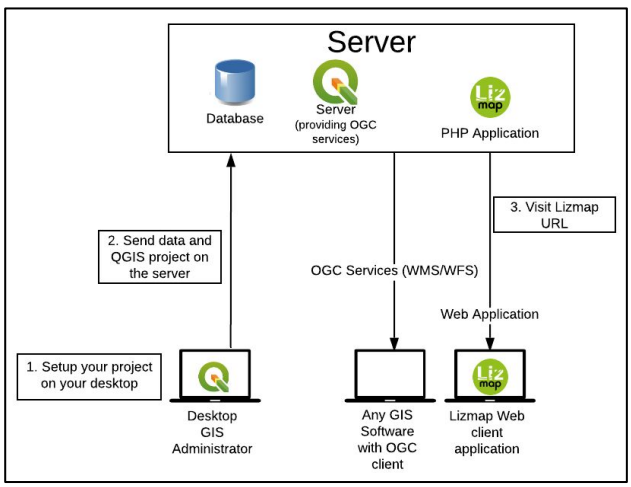

Figure 7: Lizmap Architecture (Lizmap Documentation, 2020)

Lizmap has some favorable predefined functionality for map tools such as print, measure tool, zoom history, address search and automatic geolocation. In addition, Lizmap provides "Lizmap search" in PostgreSQL tables. For activation "Locating by layer" and "Layer editing" function, the WFS capabilities in the QGIS server should be activated. (see Figure 8). WFS capabilities provides the selection of layers for publishing and specifying if they have permission for updating, inserting and deleting operations.(QGIS Documentation, 2020).

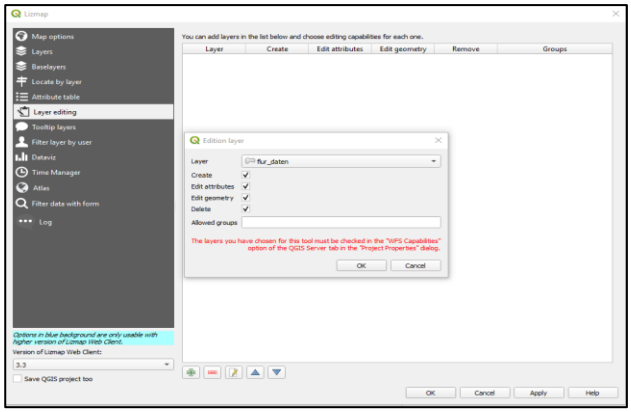

Figure 8: Activation of "Layer editing" in Lizmap plugin.

Moreover, Lizmap allows the user to perform some operations in the web interface such as: create, edit attributes, edit geometry and delete action. Those functions will be saved directly in the database and will be updated by assigned the defined symbology according to the map legend (see Figure 9, 10).

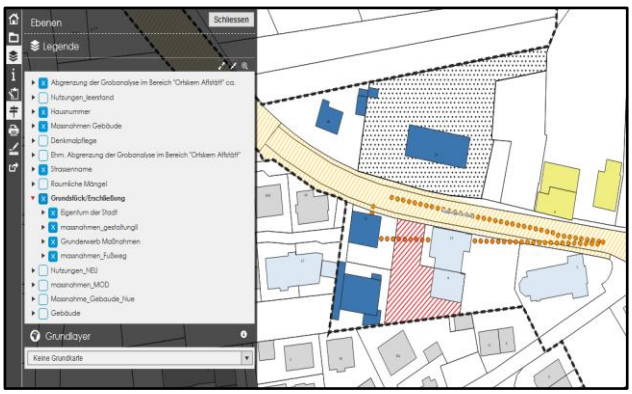

Figure 9: Lizmap web client application with different layer categories and symbologies.

Figure 10: Creating a new geometry and automatic map updating according to defined symbols in the map legend.

By assigning credits and restrictions to the administrators of the company and the municipalities, these changes can be done fast without installing GIS desktop software. Finally, the updated maps in PDF format and attributional information in CSV format are accessible to all users.

3. CONCLUSIONS

The described workflow represents an innovative integration process for the specific use case of administering and managing complex urban redevelopment areas extending over many years. The municipalities carrying out the various urban planning measures and being responsible for the legal accounting procedures could highly benefit from the developed Enterprise Application Integration System (EAI). All urban measures and financial transactions are related to spatial objects, whether a building or a parcel. Based upon this fact the non-spatial database as used in accounting and project management systems will be coupled with the spatial database where all the cadastral geometries and attributes are stored. This integration shows the great potential of open source software in respect to data aggregation and retrieving as well as automatic map generation and visualization for a better documentation. A future application of the developed integration lies in connecting devices for data acquisition on-site, which could update the database on the fly and automatically generate Web GIS maps.

An important prerequisite for the integration of data from different application systems is the definition of a common generic data structure that collects data from disjoint data sources. It allows providing updated maps via web interfaces which can be viewed not only in GIS desktop application but in other web-based applications (like accounting and project management applications). In addition, information from other applications like accounting systems may be provided in the spatial representation.

4. FUTURE STEPS

Further research could be done regarding the connection of the developed application with a three-dimensional open source database like 3D City Database (3DCityDB) for 3D city modelling (Coors et al., 2016). This software allows the import, management, analysis, visualization and export of virtual 3D city models through a database schema for relational database management systems. For visualizing, querying and performing analysis on the 3D model, a JavaScript code could be written with CesiumJS graphical user interface.

Another important future development is related to the way application systems access the spatial data: While directly accessing the spatial database is very efficient and fairly simple to be implemented, this is only possible if the application can be extended and thus requires the availability of the source code of the application. Therefore, in EAI projects the applications usually have to be much more decoupled from one another: Instead of directly accessing the database of the other system an application has to "talk" to this other system via a standardized interface, typically using a messaging-based integration service (see e.g. cyberconcepts integrator, 2021) providing a standard REST/JSON interface (JSON, Wikipedia, 2020). Then only the integration service would need access to the spatial database (the *generic representation*) and other applications could be connected to the spatial database via this service.

The described development would open the gate to other possibilities of integrating data from a larger set of applications and data sources: Using the Semantic Web standards, the *generic representation* with its open data model can be used as an RDF-compliant data storage (RDF, Wikipedia, 2020). This allows the representation of data provided in RDF format and thus the integration of official open-data sources as well as publicly available information from services like Wikidata

(Wikidata, 2021). And vice versa, using this Web standard, information from internal spatial database can be made accessible in mapping services like OpenStreetMap (OpenStreetMap, 2021).

REFERENCES

3DCityDB, 2020. About citygml, https://www.3dcitydb.org/3dcitydb/ (20 December 2020).

AdV, 2009. Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens (GeoInfoDok), Hauptdokument, Version 6.0.1, Stand: 01.07.2009, http://www.adv-online.de/GeoInfoDok/binarywriterservlet?imgUid=8f830072-8de8-9221-d5ad-8f138a438ad1&uBasVariant=11111111-1111-1111-1111-111111111111 (03 May 2020).

BauGB, 2017. Baugesetzbuch in der Fassung der Bekanntmachung vom 3. November 2017 (BGBl. I S. 3634). https://www.gesetze-im-internet.de/bbaug/ (03 January 2021).

Coors, V., Andrae, C. und Böhm, K.-H., 2016. 3D-Stadtmodelle – Konzepte und Anwendungen mit CityGML, Wichmann

cyberconcepts geostore, 2021. git.sr.ht/~cco/storage-common/tree/master/item/pgsql/geostore (4 January 2021).

cyberconcepts integrator, 2021. www.cyberconcepts.org/posts/integrator (13 January 2021).

cyberconcepts loops, 2021. www.cyberconcepts.org/posts/loops-km (13 January 2021).

JSON Wikipedia, 2020. en.wikipedia.org/wiki/JSON (31 December 2020).

Lake, R., Burggraf, D., Trninic, S. & Rae L., 2004. GML – Geography Mark-Up Language: Foundation for Geo-Web. Wiley & Sons, Hoboken.

Lizmap Documentation, 2021. Introduction, https://docs.lizmap.com/current/en/introduction.html#lizmap-architecture.(25 Januar 2021).

Montaya. S., 2017. Introduction to PyQGIS, the python environment in QGIS, online access at: https://www.hatarilabs.com/ih-en/introduction-to-pyqgis-the-python-environment-in-qgis (30 November 2020).

Obe, R., Hsu, L., & Ramsey, P., 2011. PostGIS IN ACTION. Manning publication Co. Shelter Island, NY11964. pp 312-350. OpenStreetMap 2021. www.openstreetmap.org (4 January 2021).

Ostadabbas, H., Weippert, H., Behr F-J., 2019. Database Transformation, Cadastre Automatic data processing in QGIS and Implementation in WebGIS. The international Archive of the Photogrammetry, Remote sensing and Spatial Information Sciences, Volume XlII-4/W14, 2019.

Ostadabbas, H., Weippert, H., Behr F-J., 2020. Using synergy of QFIELD for collecting data on-site and QGIS for interactive map creation by ALKIS data extraction and implementation in PostgreSQL for urban planning process. The international Archive of the Photogrammetry, Remote sensing and Spatial Information Sciences, Volume XLIII-B4-2020.

PostGIS, 2018. What is post GIS, https://en.wikipedia.org/wiki/PostGIS (26 January 2021).

PyQGIS Developer Cookbook, 2020. QGIS Documenation. https://docs.qgis.org/3.16/en/docs/pyqgis_developer_cookbook/intro.html (12 November 2020).

QGIS Documenation, 2020. QGIS server Guide, https://docs.qgis.org/3.16/en/docs/server_manual/introduction.html. (22 December 2020).

Qt – About Us, 2020. https://web.archive.org/web/20170222172844/https://www.qt.io/about-us/ (16 December 2020).

RDF Wikipedia, 2020. en.wikipedia.org/wiki/Resource_Description_Framework (30 November 2020).

Samson, G. L., Lu, J., Usman, M. M. and Xu, Q., 2017. Spatial databases: an overview, in 'Ontologies and Big Data Consideration for Effective Intelligence', IGI Global, Abstract. https://www.igi-global.com/chapter/spatial-databases/177391 (27 January 2021).

Seifert, M., 2005. Das AFIS-ALKIS-ATKIS-Anwendungsschema als Komponente einer Geodateninfrastruktur. Zeitschrift für Geodäsie, Geoinformation und Landmanagement (zfv) 130:77–81.

WikiData, 2021. www.wikidata.org (4 January 2021).

ZODB, 2020. pypi.org/project/ZODB (11 June 2020).

Zope toolkit, 2021. zopetoolkit.readthedocs.io (4 January 2021).

Zope, 2021. Documentation, www.zope.org (4 January 2021).