Introducing the 3DCityDB-Tools Plug-In for QGIS

Giorgio Agugiaro, Konstantinos Pantelios, Camilo León-Sánchez, Zhihang Yao, and Claus Nagel

Abstract This paper introduces a new plug-in for QGIS that allows to connect to the free and open-source 3D City Database to load CityGML data, structured as classic GIS layers, into QGIS. The user is therefore not required to be a CityGML specialist, or a SQL expert, as the plug-in takes care of hiding from the user most of the complexity in terms of underlying data model and database schema implementation. The user can therefore load CityGML thematic "layers" (e.g. for buildings, bridges, vegetation, terrain, etc.), explore their geometries in 2D and 3D and access and edit the associated attributes. At the same time, depending on the user privileges, it is possible to delete features from the database using either normal QGIS editing tools, or a "bulk delete" tool, also included. The plug-in is composed of two parts, a server-side one, which must be installed in the 3D City Database instance, and the client-side one, which runs as a QGIS plug-in in strict sense. A GUI-based tool is also provided for database administrators in order to install/uninstall the database-side part of the plug-in, and manage users and their privileges. All in all, the 3DCityDB-Tools plug-in facilitates the access to CityGML data for GIS practitioners from heterogeneous fields and expertise with the common denominator being the well-known QGIS environment.

This article was selected based on the results of a double-blind review of the full paper

G. Agugiaro (✉) · C. León-Sánchez
3D Geoinformation Group, Faculty of Architecture and the Built Environment, Department of Urbanism, Delft University of Technology, Julianalaan 134, 2628BL Delft, The Netherlands
e-mail: g.agugiaro@tudelft.nl

C. León-Sánchez
e-mail: c.a.leonsanchez@tudelft.nl

K. Pantelios
Coöperatieve Rabobank U.A. RNT DP Consumer Banking, Croeselaan 18, 3521CB Utrecht, The Netherlands

Z. Yao · C. Nagel
Virtualcitysystems GmbH, 7 B/C, 10789 Tauentzienstr, Berlin, Germany
e-mail: zyao@vc.systems

C. Nagel
e-mail: cnagel@vc.systems

T. H. Kolbe et al. (eds.), *Recent Advances in 3D Geoinformation Science*, Lecture Notes in Geoinformation and Cartography, https://doi.org/10.1007/978-3-031-43699-4_48

Keywords 3D city database · QGIS · CityGML · CityJSON · Plug-in

1 Introduction

Semantic 3D city models are being used more and more in a wide variety of applications (Biljecki et al. 2015) like energy (León-Sánchez et al. 2021; Monteiro et al. 2017; Rossknecht and Airaksinen 2020), flood simulation (Kilsedar et al. 2019), integration with BIM (Kolbe and Donaubauer 2021; Noardo et al. 2020), microclimate simulation (Chen et al. 2020), urban planning (Agugiaro et al. 2020), visibility analyses (Virtanen et al. 2021), traffic simulation (Ruhdorfer et al. 2018), etc.

Given the number and heterogeneity of urban objects to represent and the heterogeneity of data to describe them, adoption of standards is beneficial to facilitate usage and exchange of information depending on the application and the involved stakeholders. For this purpose, the Open Geospatial Consortium (OGC) has adopted CityGML (Gröger and Plümer 2012) as an international standard to store and exchange spatial and non-spatial data related to semantic 3D city models. CityGML comes as a UML-based conceptual data model and several encodings: two are file-based and rely on XML or JSON–the latter commonly known as CityJSON (Ledoux et al. 2019)–, while a third one is based on SQL and is called the 3D City Database or, for short, 3DCityDB (3DCityDB, 3D City Database Documentation 2023; Yao et al. 2018). The benefit of using a database encoding is that databases are built to handle and organise large amounts of data, which semantic 3D city models usually consist of.

The 3DCityDB is an open-source project currently developed for PostgreSQL, Oracle and PolarDB/Ganos databases. The database schema implements the CityGML standard with semantically rich and multi-scale urban objects. The 3DCityDB has been used in production and commercial environments for more than a decade already, as well as in academia and in several research projects related to 3D city models. Besides the database schema, the 3DCityDB normally ships with a suite of additional tools, collectively packaged as "3DCityDB Suite" (2023). The suite contains–among the rest–a Java-based Importer/Exporter, which allows to import and export XML-based CityGML and CityJSON files from/to the database, as well as to further export data in KML, COLLADA, and glTF formats for the visualisation for example in Google Earth and CesiumJS.

On the one hand, importing CityGML data into the 3DCityDB has the advantage to avoid direct XML or JSON file parsing by allowing the user to interact directly with data via "standard" tables and SQL commands. On the other hand, the structure of the database schema is rather complex and requires sometimes long SQL queries in order to extract data properly. For example, the current version 4.x of the 3DCityDB consists of a set of 66 tables used mostly to store feature data, but also to handle relations between them. Attributes referring to the same CityObject (e.g. a Building, a Bridge, a Road, etc.) are often stored in multiple linked tables. Additionally, CityGML allows for nested features (e.g. a Room is part of a Building), and

one feature can have multiple representations in terms of geometry: for any given CityObject there are multiple LoDs. In addition, also within the same LoD there can be different possibilities. For example, a building in LoD2 can be represented by means of a solid geometry, a multi-surface geometry, or by means of thematic surfaces (e.g. an aggregation of WallSurfaces, RoofSurfaces, GroundSurfaces, etc.). On top of that, in the same 3DCityDB instance there can co-exist several "copies" of a 3D city model. For the sake of simplicity, we will refer to them as "scenarios" in this paper, but in reality they are stored in different database schemas. The resulting complexity of the 3DCityDB reflects the rich structure of CityGML, but, eventually, the level of SQL knowledge required to interact with it might be indeed beyond that of a common GIS practitioner, de facto limiting access to the data stored therein. The query shown in Fig. 1 provides a simple example. It retrieves all building roofs built since 2015 from the 3DCityDB. As it can be seen, several tables must be joined and geometries must be collected. This implies that a rather advanced knowledge of SQL and, above all, of the 3DCityDB structure is required to write and successfully run such query.

```
SELECT
    ts.id AS roof_id,
    co_ts.gmlid AS roof_gmlid,
    b.id AS building_id,
    co.gmlid AS building_gmlid,
    b.year_of_construction,
    ST_Collect(sg.geometry) AS roof_geom
FROM
    citydb.thematic_surface AS ts
    INNER JOIN citydb.cityobject AS co_ts
        ON (co_ts.id = ts.id)
    INNER JOIN citydb.surface_geometry AS sg
        ON (ts.lod2_multi_surface_id = sg.root_id)
    INNER JOIN citydb.building AS b
        ON (b.id = ts.building_id)
    INNER JOIN citydb.cityobject AS co
        ON (co.id = b.id)
WHERE
    ts.objectclass_id = 33 AND -- roofsurfaces
    b.objectclass_id = 26 AND -- buildings
    b.year_of_construction >= '2015-01-01'::date
GROUP BY
    ts.id,
    co_ts.gmlid,
    b.id,
    co.gmlid,
    b.year_of_construction
ORDER BY
    b.id,
    ts.id;
```

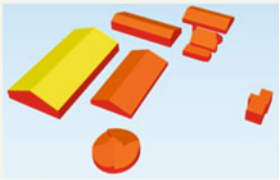


Fig. 1 Example of query to the 3DCityDB to extract the buildings roofs built since 2015. A simple visual example of the result is given in the image in the lower bottom

These limitations appear even stronger if one recalls the fact that QGIS (formerly: Quantum GIS) was originally developed in the early 2000s with the primary goal of accessing data stored in PostgreSQL/PostGIS.

Overcoming the above-mentioned limitations has been one of the main reasons behind the development of the 3DCityDB-Tools plug-in: the purpose is to let the plug-in "hide" and handle the complexity of the 3DCityDB schema(s) in the background, while providing a user-friendly, GUI-based interface directly from within QGIS.

Just two conceptually similar plug-ins were found during the preliminary research work. They were tested and analysed before starting with the development of the 3DCityDB-Tools plug-in, in order to collect information and learn from previous experiences. In general, both are designed to connect to PostgreSQL/PostGIS and both are still rather limited in the type and extents of offered functionalities, not documented or apparently not in active development anymore.

The 3DCityDB Explorer plug-in (2023), for example, is available in GitHub, but last updated in 2021. It allows to load data from the 3D City Database and modify the CityGML generic Attributes of the underlying geometries. Data are loaded into QGIS considering a combination between a maximum number of features (set by the user) and the current extents of the map. However, it only works for "Building" features represented in LoD2 geometries, and no dependent objects (e.g. BuildingInstallation, Room) can be loaded. Additionally, the layers do not contain the attributes. Finally, only one "scenario" can be accessed. Lastly, the plug-in doesn't seem to account for cases of multiple database users with different privileges.

The 3DCityDB Viewer plug-in 3DCityDB Viewer (2023) is also available in GitHub, although it seems that the project is not maintained anymore, the last commit being in 2021, too. The plug-in allows to load data from a 3DCityDB instance. The user can load "Building" features based on all LoDs, but no children objects. Features are however loaded only in terms of geometry, so no attributes are available at all. As with the "3DCityDB Explorer" mentioned before, only tables from the default citydb schema can be accessed. There is no check on the amount of data to load, which means that the plug-in attempts to import the entire database. This means that, in case of huge amounts of data, QGIS might crash. Finally, similarly to the previous plug-in, it doesn't seem to support multiple database users with different access privileges.

Although it does not interact with the 3DCityDB, the CityJSON-Loader plug-in (2023); Vitalis et al. 2020) was also evaluated, as it allows to load into QGIS and interact with CityGML data encoded as CityJSON files. The user can select which features to import, and this applies also to the different LoDs. It is important to note that the plug-in loads CityJSON files by converting the data into vector layers stored temporarily in memory. This means that changes happening in the QGIS environment (both in geometries or attributes) are not saved to the original CityJSON file. In order to save changes, users need to export the layers as a new file, however QGIS does not support a CityJSON driver for writing files. Consequently, the plug-in focuses mostly on loading data and does not allow for direct data modifications. Lastly, it is not possible to load data just for a particular area. For big CityJSON files, this could cause performance and stability issues in QGIS.

The results of the preliminary analyses were collected in order to help define the overall goal, the user and software requirements of the "3DCityDB-Tools" plug-in. In the remainder of the paper, the main characteristics and functionalities of the plug-in will be presented and described, followed by a discussion on the current development status, the current limitations and the planned future improvements.

2 User and Software Requirements

From the user point of view, the plug-in should allow to load data from the 3DCityDB as "layers", ideally following the Simple Features for SQL Model (2023). In simple words, this means that each layer contains a number of objects having each one a single geometry and one tuple of attributes. Probably, the most well-known example is represented by a shapefile. In the case of CityGML, this means that for each class–for example in the case of a Building–there might exist different layers in which the associated attributes are the same, but the geometries vary depending on the chosen LoD. As a matter of fact, the number of possible layers that can be generated following this approach is close to 600, therefore, from the software point of view, this requires some logic to deal with these numbers.

Just to exemplify this concept, a CityGML building object might be equivalent to several layers that follow the Simple Features for SQL Model. If the attributes can be considered to be always the same, this does not apply however to its geometries. A building could be represented by means of its footprint (i.e. as LoD0), or as a prismatic geometry (i.e. as LoD1), or as a geometry that includes the simplified shape of the roof (i.e. as LoD2), or a more detailed one (i.e. LoD3), etc. Additionally, a building might also be decomposed into its component surfaces, e.g. as a set of Ground-, Wall- and RoofSurfaces. Again, each thematic surface can be associated with geometries at different levels of detail. Allowing the user to easily choose which geometry to use and work with (also depending on the available ones in the database), has been one of the main challenges (and goals) of the 3DCityDB-Tools plug-in.

Additionally, a 3D city model can be extremely large and therefore it may be not possible to load it completely in memory without crashing the client. This implies that some logic is also required at software level to limit or control the amount of data that the user can select and load into QGIS. As a result, this has been another goal to pursue when developing the plug-in.

Furthermore, since data are already stored in the database, the user should be allowed to perform typical database operations, i.e. inserts, updates, and deletions. In the case of the 3DCityDB, and depending on the type of privileges granted to the user, it was decided that:

- No insert operations are allowed at all. Nevertheless, inserting new CityGML data can still be carried out as usual and at any time by means of the existing Importer/Exporter tool;

- Attributes can be edited and updates are then stored back to the database, but geometries cannot be modified;
- Features can be deleted.

Finally, unlike other existing plug-ins, the user should be given the possibility to access different citydb schemas (aka "scenarios").

This set of above-mentioned user requirements has contributed to the definition of the software requirements, as well as the overall structure of the plug-in. First and foremost, the plug-in has been developed based on the 3DCityDB version 4.x for PostgreSQL/PostGIS, but avoiding–as far as possible–technological lock-ins in case it might be extended in future to support Oracle or other databases. The 3DCityDB 4.x supports CityGML v. 1.0 and 2.0, but not the recently released CityGML 3.0 (Kutzner et al. 2020).

Regarding QGIS, the Long Term Release v. 3.22 (first released in autumn 2021) has been chosen, given its maturity–although the plug-in works also on the latest QGIS LTR v. 3.28 (as of summer 2023) without problems. The reason for choosing QGIS as the target front-end of the plugin is due to the fact that it is a well-known and established open-source software, widely used by a heterogeneous and steadily growing community. In addition, since it's very first releases in the early 2000s it has a native support for PostgreSQL/PostGIS. Furthermore, it has a strong 2D and some 3D visualisation functionalities, although the latter ones are still a bit unstable. Last but not least, it can be extended with Python-based plug-ins, for which several freely available examples and a fairly good documentation already exist.

3 The 3DCityDB-Tools Plug-In

In general, the 3DCityDB-Tools plug-in is composed of two main parts. The first is the server-side one (also called "QGIS Package") and it is written completely in PL/pgSQL, the procedural language of PostgreSQL. The "QGIS Package" must be installed on top of a 3DCityDB database instance in order to enable the communication and data exchange between the 3DCityDB and QGIS.

The second, client-side part is written in Python 3.9 and uses the QGIS Python-based API in order to communicate with the host application. The Qt library is used for all user interface elements, while the dialogs are designed in Qt Creator (Qt 2023), which is also shipped with QGIS. Except for the server-side part, there are no additional software requirements for the client-side to work, as all aforementioned libraries are provided by the QGIS installation. The overall structure is represented in Fig. 2.

More in detail, the "QGIS Package" is responsible to create and manage the layers, the users and their privileges. In particular, each layer consists in a database view that links all necessary tables containing the feature attributes with a materialized view containing the geometries for the selected LoD. A simplified example is given in Fig. 3, based on the class "Building". In the 3DCityDB, the attributes are stored in

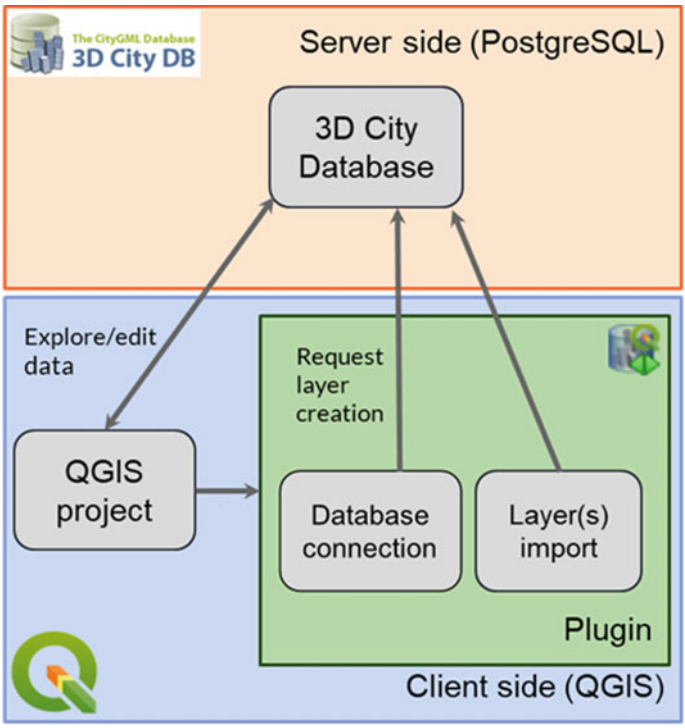


Fig. 2 Overall structure of the 3DCityDB-Tools plug-in, with a server-side part for PostgreSQL and a client-side one for QGIS

tables CITYOBJECT and BUILDING. These two tables are linked together using as primary/foreign key the ID of the "Building" object and then linked to the respective materialized view containing a specific LoD geometry. A set of naming conventions for prefixes and suffixes has been defined and implemented that allows to identify each layer uniquely within the same 3DCityDB instance. At the same time, triggers and trigger functions have been developed to make each view updatable (as far as the attributes are concerned).

The reason for choosing materialized views for the geometries is due to the complexity of how geometries are decomposed and stored in the 3DCityDB. A detailed explanation of how the CityGML geometry model is mapped to the 3DCityDB is beyond the scope of this paper, so the reader is invited to refer to the online documentation for a better understanding. In short, regardless of the nature of the original GML geometry (e.g. solid or multi-surface), all geometries are decomposed and stored as simple 3D polygons, whereas information about their hierarchy and aggregation is also preserved. Preliminary performance tests have shown that querying (and thus aggregating again) the 3D polygons directly from the corresponding 3DCityDB table is rather time-consuming. As a consequence, the materialized views offer instead a good compromise to allow for a better user experience

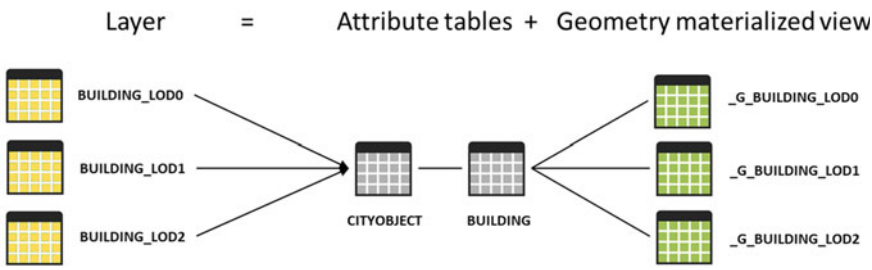


Fig. 3 Simplified representation of how layers are composed for the 3DCityDB-Tools plug-in in the case of Building objects. Tables containing building attributes (here: CITYOBJECT and BUILDING) are linked to the corresponding materialized views with specific LoD geometries

at the cost of some storage space and the time needed to generate/refresh them upon layer creation. In addition, another advantage of relying on materialized views is that also implicit geometries can be created in advance and used as "normal" geometries. In CityGML, implicit geometries can be used to represent features (e.g. a street lamp, a traffic sign, a bus stop, etc.) by means of "template geometries" that must then be instantiated, roto-translated and scaled by means of a 3D affine transformation.

In order to cope with the large number of layers that may result from all possible combinations, a number of checks has been added and implemented. First, layer creation functions are grouped according to the CityGML modules (Building, Bridge, Vegetation, Transportation, Terrain, etc.) and can be invoked individually. Therefore, if a user is only interested in working with building data, only those layers will be created. Second, during the layer creation process, a check is performed to count the number of existing features for that layer. If, for example, there are no data at all regarding Rooms or BuildingInstallations in the database, then those layers will not be generated. The same goes also for LoDs: layers are created only if data (here: geometries) are available in a specific LoD.

Finally, the user can define the size of the area for which the layers will be generated. This means that, especially with very large city models, it is not necessary to generate materialized views of the whole city model, but only within a user-selected, smaller area. This has the benefit of reducing the storage space consumed by the materialized view and, more importantly, the time needed to refresh the materialized views.

The client-side part of the plug-in allows the user to interact with the "QGIS Package" on the server via a set of GUI-based dialogs, and to interact with the data in QGIS itself. Currently, as of version 0.8.2, the client-side part offers three GUI-based tools:

- The "**QGIS Package Administration**" is used to install the server-side part of the plug-in, as well as to set up database user access and user privileges;
- The "**Layer Loader**" allows the user to load and interact with layers in the 3D City Database directly from QGIS

- The "**Bulk Deleter**" can be used to bulk delete features from the database, either all at once, or by means of spatial and feature-related filters.

The "QGIS Package Administration" is meant to be used only by a database administrator. Figures 4, 5 and 6 serve as a visual reference as the following text will reference certain parts of the GUI dialog indicated by letters. Once the connection is set up (a), the database administrator can perform different operations. The first time, the installation of the "QGIS Package" must be carried out (b). For every action, status information is provided in the Connection status box (c). Once the server-side part is installed, the user installation box is activated. It is now possible to choose which database users are allowed to connect to the selected 3DCityDB database instance from the plug-in. This is achieved by adding users to a specific database user group (d). For each group member, the database administrator has to set up the server-side configuration by creating a user schema (e) which will contain the user's layers and settings. Finally, database privileges (read-only or read and write) (f) can be granted or revoked for each user and for each existing citydb schema (or "scenario") (g). Once the database administrator has completed the setup, the GUI dialog can be closed. The client-side plug-in can now be used–provided the user is entitled to.

The "Layer Loader" GUI dialog can be loaded by any user. Again, Figs. 7 and 8 will serve as a visual reference. Once the user has connected to a 3DCityDB instance, the list of citydb schemas (or "scenarios") is shown, and information is provided regarding the access privileges (read-only: "ro", or read and write: "rw") (h). Once the user selects the citydb schema to work with, the extents of the whole dataset are displayed in the map canvas. It is possible to set the area extents for which the layers will be generated (i), those are then displayed by means of the blue bounding box. Additionally, the user can choose whether the layers have to be created for all CityGML modules, or only for some of them (j). Afterwards, the next operation is the creation of the layers. Materialized views and the updatable views needed for each layer are set up, and materialized views can be subsequently refreshed. If all requirements are met (as indicated in the Connection status box at the bottom of the dialog), the user can move on to the next tab, where layers will be eventually loaded into QGIS.

For the Layers tab, Fig. 9 will provide visual guidance to the reader when it comes to the principal operation to be carried out. First, the user is given the chance to further reduce the extents of the data to be imported into QGIS (l). These extents are represented by means of a green bounding box. By default, the green bounding box coincides with the blue one defined before, but the user is given the option to change it. Successively, the user filters the layers to be selected by first defining the CityGML module and the LoD (m). These drop down menus are updated depending on the actual data available within the green bounding box. Finally, the filtered list of available layers is generated, from which the user can check which layer(s) will be eventually loaded into QGIS (n).

Once the layers are imported into the QGIS main window, the user can interact with them as "normal" GIS layers and perform the usual set of operations. A hierarchical

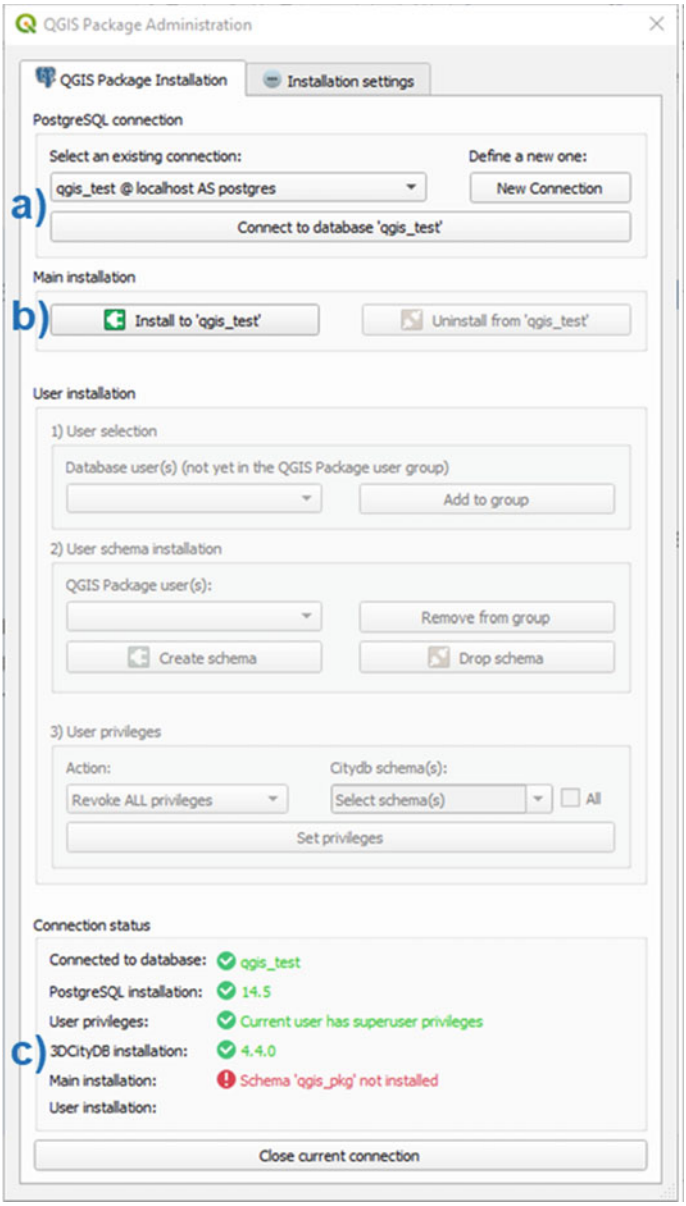


Fig. 4 Overview of the connection and installation operations within the "QGIS Package Administration" GUI dialog in the client-side part of the plug-in

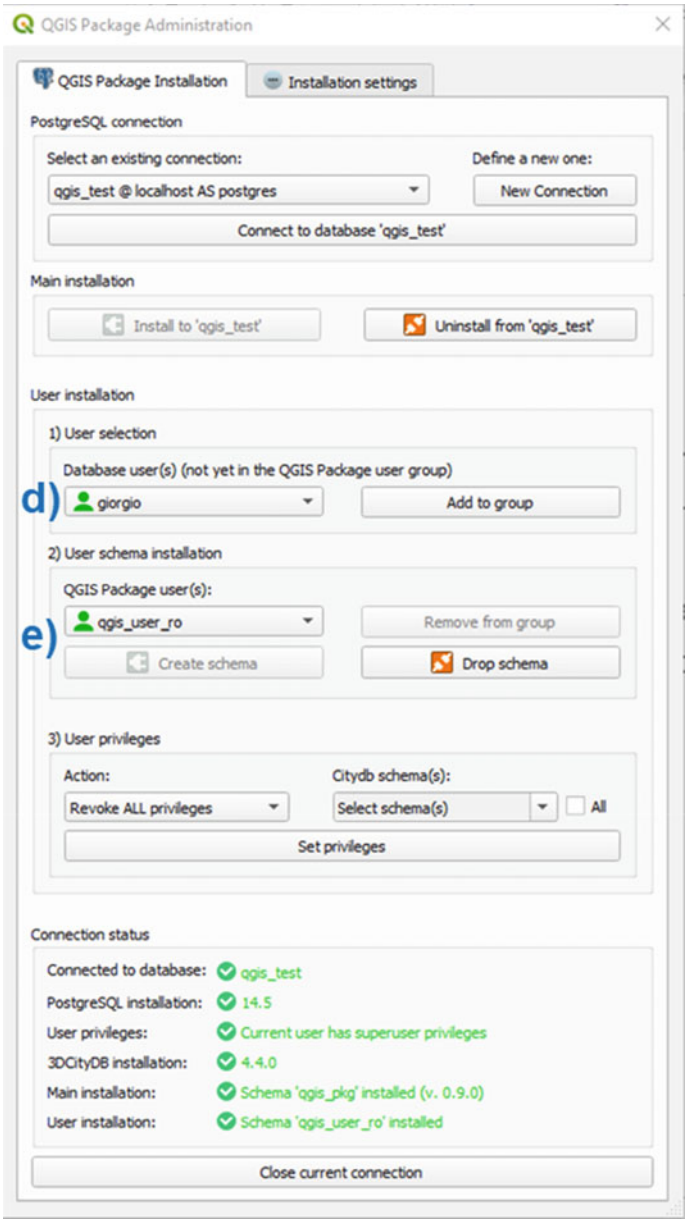


Fig. 5 Overview of the user management operations within the "QGIS Package Administration" GUI dialog in the client-side part of the plug-in

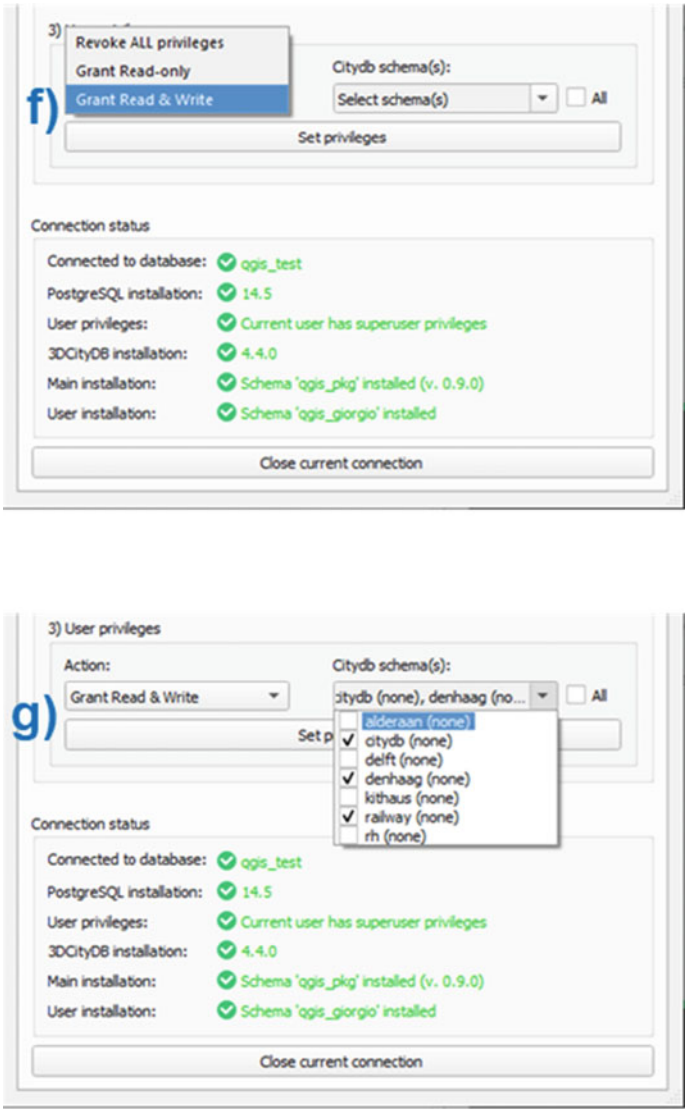


Fig. 6 Overview of the user privileges operations within the "QGIS Package Administration" GUI dialog in the client-side part of the plug-in

Table of Contents (Layers tab) is generated and updated upon each layer import, it offers an overview of the loaded layers ordered according to the CityGML module and LoD. An example is provided in Fig. 10. The user can select features, and access their attributes, either via the table view, or by means of customised attribute forms that present the feature's attributes, but also contain nested tables related to CityGML generic attributes, addresses and external references. If the user is allowed to edit

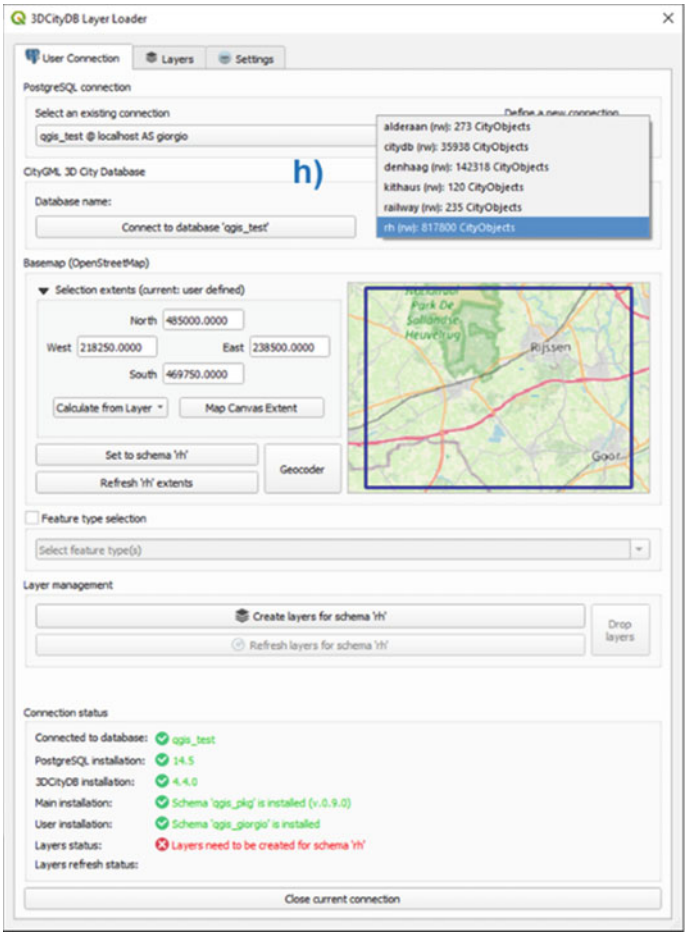


Fig. 7 Overview of the connection operations within the "User connection" tab of the "Layer Loader" GUI dialog in the client-side part of the plug-in

data, the attribute forms perform a series of checks to reduce errors during data entry. For example, enumeration values are converted to drop down menus and, in case of invalid input, the user is informed also visually. For example, in Fig. 11 an error is shown for the number of storeys below ground, as the value must be a positive integer, and in the case of the storey height value, the units of measure are missing. Finally, the user can visualise the layers in 3D using either the QGIS 3D Map or the plug-in Qgis2threejs Exporter (Qgis2threejs 2023), as shown in Fig. 12. It must be said however that 3D visualisation support is still rather unstable in QGIS, especially in the case of the 3D Map which might crash while loading the 3D layers and rendering the scene. The Qgis2threejs Exporter is a bit more stable, however

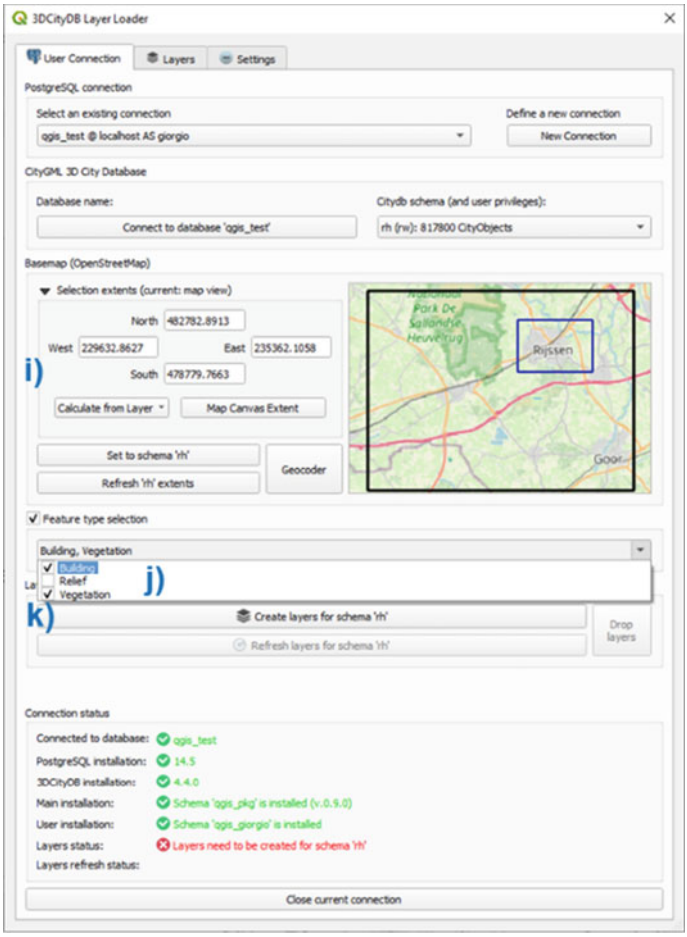


Fig. 8 Overview of the layer generation operations within the "User connection" tab of the "Layer Loader" GUI dialog in the client-side part of the plug-in

artefacts are sometimes still visible with some 3D geometries. These limitations will be discussed later on in the paper.

The third and last available GUI dialog of the 3DCityDB-Tools plug-in is the "Bulk Deleter". It is meant to be used by a user having at least read-and-write privileges. Figures 13 and 14 provide a visual reference of the following text. Once the user has connected to a 3DCityDB instance, the list of available citydb schemas (or "scenarios") is shown in a similar way as seen before, however only those citydb schemas are listed for which read-and-write privileges are granted. Once the citydb schema is selected (o), the user can delete features in two ways. The first is to clean up the whole citydb schema (p). In other words all tables are truncated and the schema is completely emptied. Alternatively, the user can perform a selection on the type

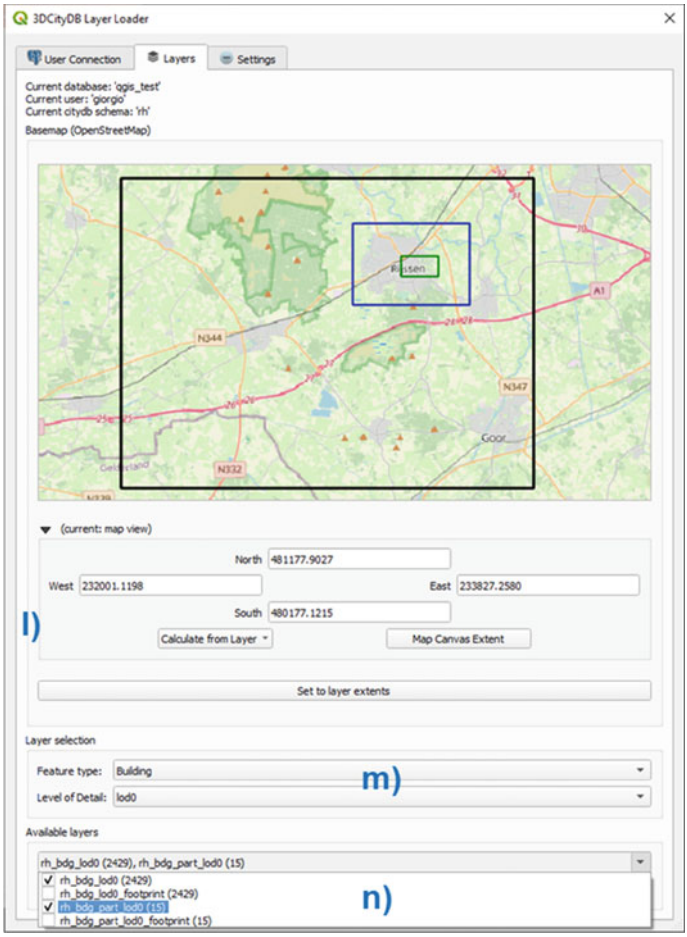


Fig. 9 Overview of the operations within the "Layers" tab of the "Layer Loader" GUI dialog in the client-side part of the plug-in

of features to delete, choosing either from a list of available CityObject modules, or from a list of available CityGML top-class features (q). Additionally, it is possible to set the area extents inside which features will be deleted. Such extents are represented by a red bounding box. The information presented in the drop down menus is updated dynamically.

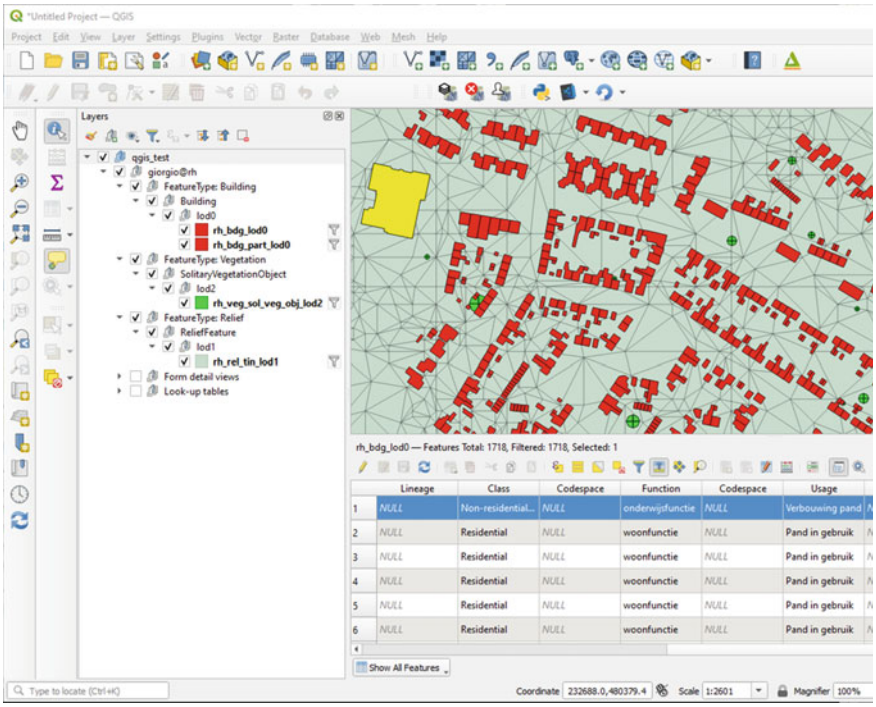


Fig. 10 Example of layers loaded in QGIS. The hierarchical Table of Contents (Layers tab on the left) is generated and updated automatically upon layer import. Attributes can be accessed via the standard attribute table view

4 Tests and Current Limitations

The plug-in has been tested with different datasets varying both in terms of size (geographical extents and number of features) and in terms of available CityGML modules and classes. Table 1 contains an overview of the main characteristics of such test datasets. All datasets have been previously imported into a 3DCityDB instance using the Importer/Exporter. Tests have been conducted to check whether layers can be generated correctly, the attributes can be retrieved, displayed and updated, the geometries can be correctly visualised in 2D and in 3D, the features can be deleted. At the same time, the plug-in was tested on a variety of software configurations (Windows, Linux, MacOS) and different PostgreSQL/PostGIS versions.

Tests conducted so far show that, in general, the back-end part of the plug-in works as expected with all datasets used. Of course, the larger the area is for which the layers are created, the longer the time will be to refresh the materialized views associated to the layers. To test the most time-consuming use case, all layers were generated for all datasets for the whole dataset (e.g. maximum extents). As results show in Table 2, both operations to generate the layers and to refresh the underlying

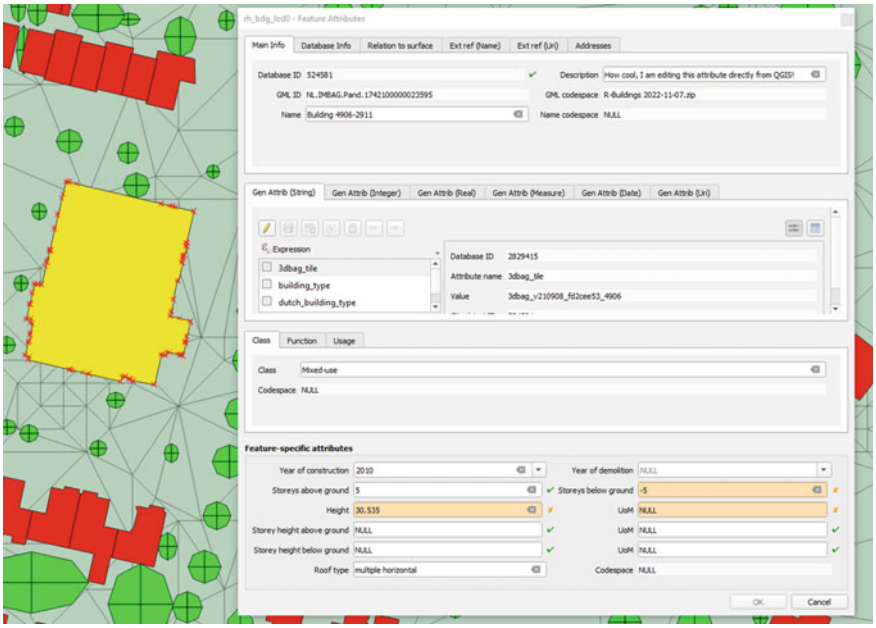


Fig. 11 Example of a customised attribute form used in QGIS to present all attributes of a feature. If the user is allowed to edit the attributes, checks are carried out interactively to avoid errors during data input. The user is notified also visually. External references, addresses and generic attributes are also supported and can be added, deleted or updated

materialized views could always be carried out in acceptable times from the user perspective. Obviously, selecting smaller extents leads to shorter times.

When it comes to the front-end, regardless of the extent of the layers (i.e. the blue bounding box), the ability to further reduce the size of the data to load into QGIS (i.e. the green bounding box) is useful to guarantee a good user experience. Using the front-end part of the plug-in in QGIS, the user can therefore:

- Visualize and interactively define an area of interest for which layers will be created;
- Choose and select for which CityGML module(s) the layers will be created;
- Further select for which features and for which LoD layers can be selected and loaded into QGIS;
- Visualize data in 2D and 3D using QGIS (or the Qgis2threejs plug-in);
- Access feature attributes and eventually edit them, if sufficient privileges are granted.

Usability tests have been carried out with heterogeneous beta testers and the feedback collected has been so far positive, proving that the user interaction with the 3DCityDB has become indeed easier than before. Some feedback suggestions have also been used to improve the design of the GUI dialogs. There are however still

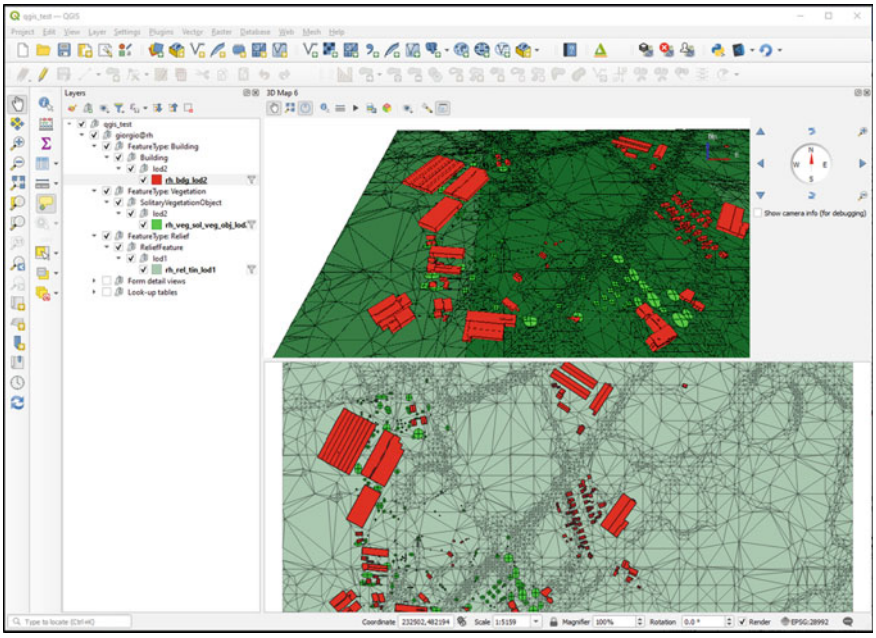


Fig. 12 Example of 2D and 3D visualisation (using QGIS 3D Map View)

some limitations that currently apply to the 3DCityDB-Tools plug-in. For example no raster-based layers from the 3DCityDB are supported. This applies basically to the Relief module, and more specifically to the RasterRelief class. Additionally, the CityGML module CityObjectGroup is not supported by the "Layer Loader", although it is supported by the "Bulk Deleter". Currently, no CityGML appearances are supported, i.e. neither colours nor textures can be read from the 3DCityDB and applied to the loaded features. This is however in part due to the limitations imposed by the capabilities of QGIS, which only supports simple or rule-based coloured surfaces for polygon layers. Finally, no CityGML Application Domain Extensions (ADEs) are supported at the moment.

Although it is not a limitation related to the plug-in itself in a strict sense, a major drawback experienced so far by the users is due to some 3D visualisation problems in QGIS which affect the possibility to visualize CityGML data in 3D. Therefore–for the sake of completeness–they will be briefly reported here. As a matter of fact, trying to visualise layers in 3D may lead, from time to time, to visualisation artefacts (e.g. geometries showing spikes) or to crashes of QGIS altogether. This happens when using both the default QGIS 3D Map and–less frequently–the Qgis2threejs plug-in. Some tests were carried out to check the validity of the geometries. As shown for example in Fig. 15, while QGIS shows artefacts (1) (please note the spikes in the WallSurface geometries, shown in the upper part of the image; the roofs are shown only in the lower part), the same (valid) WallSurface geometries can be successfully

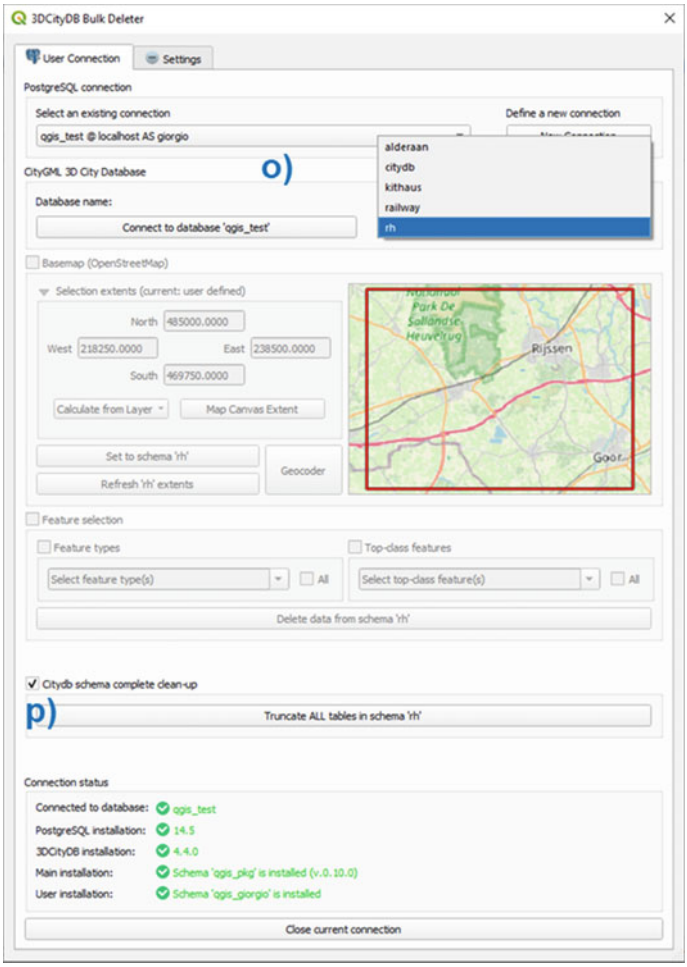


Fig. 13 Overview of the connection and database clean-up operations within the "Bulk Deleter" GUI dialog in the client-side part of the plug-in

rendered in Safe Software's FME Data Inspector (2) or Google Earth (3). Further tests were conducted exporting the datasets to other formats, such as GeoPackage. In QGIS, the same issues still apply, no matter whether data come from PostGIS or from a GeoPackage file. The reason for such problems is not clear at the moment and needs therefore further investigation, however a first hypothesis is that the reason could be how geometries stored as well-known binary are then converted before triangulation and rendering, with possible loss of precision in the geometry coordinates happening during the process.

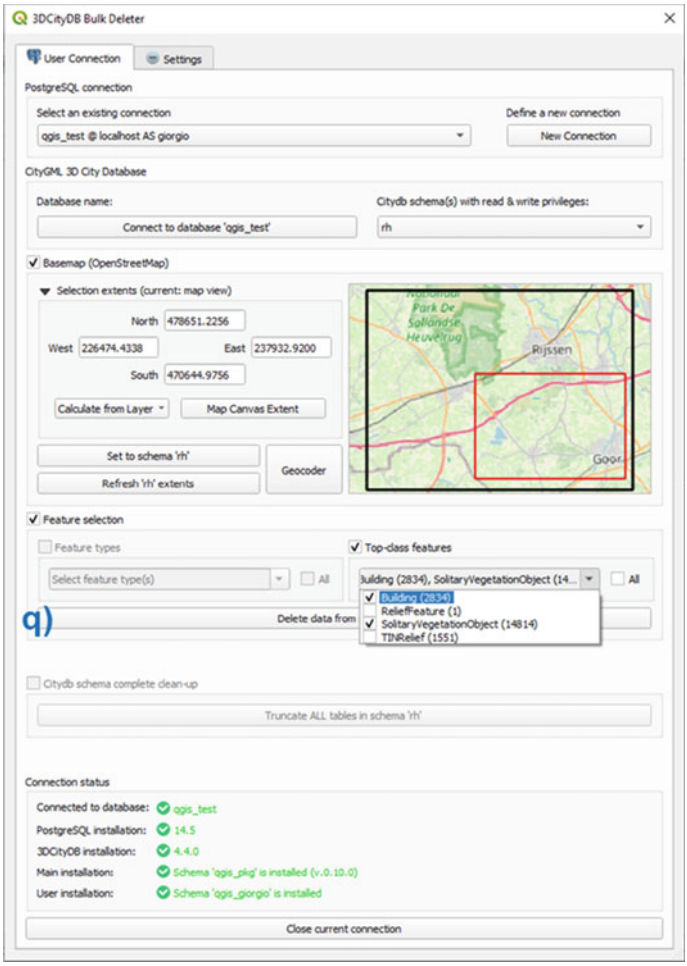


Fig. 14 Overview of the feature selection operations within the "Bulk Deleter" GUI dialog in the client-side part of the plug-in

5 Conclusions and Outlook

This article has presented and described the design and the main functionalities of the 3DCityDB-Tools plug-in for QGIS. The plug-in is already available on GitHub (2023) and on the official QGIS plug-ins repository (2023). The plug-in was conceived and developed to facilitate management and visualisation of data stored in the 3D City Database, which currently supports CityGML v. 1.0 and 2.0. As semantic 3D city models tend to be huge datasets and are generally best managed in spatial databases, the main idea behind the development of this plug-in is to facilitate access and use of CityGML/CityJSON data for those practitioners that lack a

Table 1 List of datasets used to test the "3DCityDB-Tools" plug-in, and their main properties

Dataset [source]	Number of CityObjects	Database schema size	CityGML modules	LoDs
FZK Haus (2023)	120	5 MB	Building	0, 1, 2, 3, 4
Railway (2023)	235	23 MB	Bridge, building, CityFurniture, generics, relief, transportation, tunnel, vegetation, WaterBody	3
Delft (2023)	287,242	2.4 GB	Building	0, 1, 2
Amsterdam (2023)	319,117	4 GB	Bridge, buildings, generics, LandUse, transportation, vegetation, WaterBody	1
Rijssen-Holten (2023); León-Sánchez et al. (2022)	827,105	1.8 GB	Building, relief, vegetation	2
Den Haag	3,143,353	4 GB	Building	1, 2
Vienna (2023); Agugiaro (2016)	7,512,795	23 GB	Building, generics, LandUse, vegetation, relief	1, 2

Table 2 Time required to generate the layers and to refresh them, respectively. Times shown in this table are measured on a Dell Latitude 7490 compute with Core i7-8650U @ 1.90 GHz CPU, 32 GB RAM and 2 TB SSD, and Windows 10 21H2, QGIS v. 3.22 LTR, PostgreSQL v. 14 and PostGIS v. 3.2. Each time value is the average of three consecutive measurements

Dataset	Layer generation (s.s)	Layer refresh (mm:ss.s)
FZK Haus	2.3	00:00.4
Railway	10.4	00:01.4
Delft	6.5	01:12.1
Amsterdam	11.4	01:51.4
Rijssen-Holten	11.3	01:15.2
Den Haag	14.3	02:05.9
Vienna	33.5	13:38.7

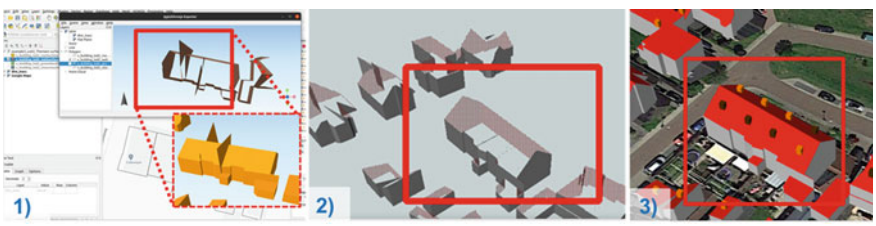


Fig. 15 Examples of artefacts generated during 3D visualisation in QGIS (via Qgis2threejs plug-in). The WallSurfaces of the building highlighted in **(1)** are shown with and without RoofSurfaces. Some artefacts while rendering the WallSurfaces can be noticed. Such artefacts are however not present when visualising the same WallSurface geometries in **(2)** (FME Data Inspector) and in **(3)** (Google Earth, after exporting to KML using the Importer/Exporter)

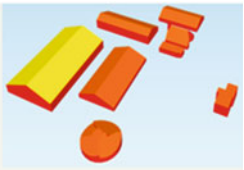
deep knowledge of the international standard OGC CityGML data model, and/or have limited experience with SQL/Spatial-RDBMSs in general. For this reason, a view/layer-based concept was conceived and implemented in order to bridge common GIS practitioners and complex 3D city models. At the same time, an approach to allow different users to access the 3D city model database was realised.

As a result, the plug-in allows to connect to local or remote instances of 3DCityDB for PostgreSQL/PostGIS and to load data as "classical" layers into QGIS. The user can then interact with them as usual, i.e. perform analyses, work with associated attributes, edit and update them, explore and visualise the data. Additionally, data in the database can be deleted, either using classical QGIS editing tools, or bulk-wise.

From the internal tests carried out so far, and from the feedback received from some early adopters and external testers, the plug-in seems to work well and, most importantly, it can be used not only by experts but also by the "target user" it was originally developed for: GIS practitioners without deep knowledge of CityGML or databases. The complexity of CityGML, and by reflection of the 3DCityDB structure, is sufficiently "hidden" to leverage access to CityGML/CityJSON data to a wider number of users interested in spatial urban analytics.

A simple visual example of the main contribution provided by the 3DCityDB-Tools plug-in is presented in Figs. 16 and 17: the same query shown in Fig. 1 is now run either using the "QGIS Package" (i.e. with circa 50% less SQL code lines), or directly from QGIS using only GUI tools (i.e. without any need to write SQL).

More in general, the added value of choosing QGIS to develop this plug-in is manifold. First and foremost, it is a free and open-source platform which is used by a large and steadily growing amount of heterogeneous users. If, on the one hand and as described in the previous sections, there exist still some issues mainly regarding the 3D visualisation and support for textures, on the other hand QGIS comes with a plethora of processing and data conversion tools "out of the box"–without counting the number of already available plug-ins developed by the community. For example,



```
SELECT
    rs.id AS roof_id,
    rs.gmlid AS roof_gmlid,
    rs.building_id AS bdg_id,
    b.gmlid AS bdg_gmlid,
    b.year_of_construction,
    rs.geom AS roof_geom
FROM
    qgis_user_ro.citydb_bdg_lod2_roofsurf AS rs
    INNER JOIN qgis_user_ro.citydb_bdg_lod2 AS b
        ON b.id = rs.building_id
WHERE
    b.year_of_construction >= '2015-01-01'::date
ORDER BY
    b.id,
    rs.id;
```

Fig. 16 Using the "QGIS Package", i.e. the server-side part of the 3DCityDB-Tools plug-in, the same query as in Fig. 1 can now be written in 16 lines (instead of 30)

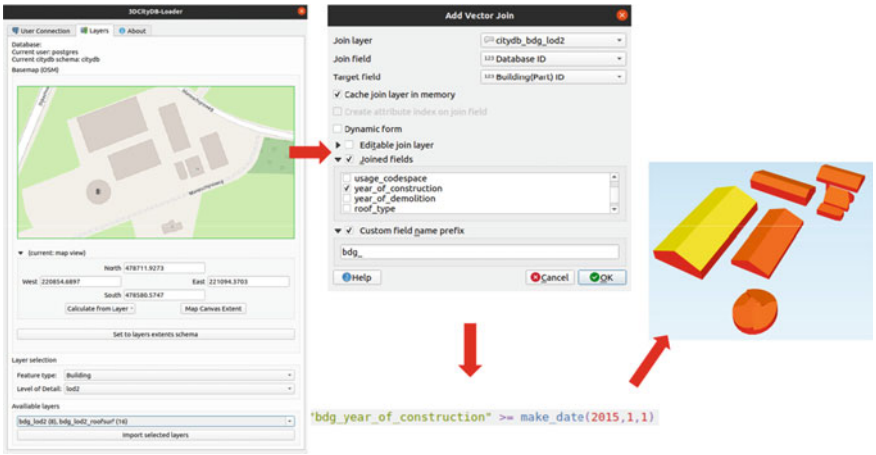


Fig. 17 Using the client-side part of the 3DCityDB-Tools plug-in, the same query as in Fig. 1 can now be constructed and run using the GUI of QGIS without writing any SQL

QGIS already incorporates SAGA GIS (2023) and GRASS GIS (2023) functions and the ability to further transform data formats through GDAL (2023).

Although the current development status of the plug-in already covers most of the originally envisioned characteristics, there are still a number of further improvements that are planned for the future. Besides further testing, we are looking forward to more user feedback so that the overall plug-in structure (functionalities, GUI dialogs, etc.) can be possibly improved. Looking further, inclusion of CityGML Application Domain Extension (ADE) support is planned despite probably being among the major challenging ones, as it will require the revision and extension of the current implementation for both the server-side and the client-side part in order to cope with new ADE classes, data types, and relations. This is however still subject of research at TU Delft with preliminary results expected during the second half of 2023. Finally, the upcoming version 5.0 of the 3DCityDB will support CityGML 3.0. Therefore it will be a valuable addition to investigate–and possibly integrate–support for it also in the "3DCityDB-Tools" plug-in.

Acknowledgements The authors would like to thank María Sánchez Aparicio for the thorough testing of the plug-in and the feedback provided to improve its usability, and Tendai Mbwanda for the contribution to the initial development of the "Bulk Deleter".

References

3DCityDB, 3D City Database Documentation. Retrieved from https://3dcitydb-docs.readthedocs.io. Accessed on 1 Jul 2023

3DCityDB Explorer. Retrieved from https://github.com/3dcitydb/3dcitydb-qgis-explorer. Accessed on 1 Jul 2023

3DCityDB Suite. Retrieved from https://github.com/3dcitydb/3dcitydb-suite. Accessed on 1 Jul 2023
3DCityDB-Tools. Retrieved from https://github.com/tudelft3d/3DCityDB-Tools-for-QGIS. Accessed on 1 Jul 2023
3DCityDB Viewer. Retrieved from https://github.com/aberhamchristomus/3DCityDB-Viewer. Accessed on 1 Jul 2023
Agugiaro G (2016) First steps towards an integrated CityGML-based 3D model of Vienna. In: ISPRS annals of the photogrammetry, remote sensing and spatial information sciences, 2016 XXIII ISPRS congress, Prague, Czech Republic, vol III-4, pp 139–146
Agugiaro G, González FGG, Cavallo R (2020) The city of tomorrow from… the data of today. ISPRS Int J Geo-Inf 9(9, 554):42
Amsterdam Dataset. Retrieved from https://3d.bk.tudelft.nl/projects/geobim-benchmark/amsterdamgml.html. Accessed on 1 Jul 2023
Biljecki F, Stoter J, Ledoux H, Zlatanova S, Çöltekin A (2015) Applications of 3D city models: state of the art review. ISPRS Int J Geo–inf 4:2842–2889
Chen S, Zhang W, Wong NH, Ignatius M (2020) Combining CityGML files and data-driven models for microclimate simulations in a tropical city. Build Environ 185:107314. https://doi.org/10.1016/j.buildenv.2020.107314
CityJSON Loader. Retrieved from https://github.com/cityjson/cityjson-qgis-plugin. Accessed on 1 Jul 2023
Delft Dataset. Retrieved from https://3dbag.nl/en/download. Accessed on 1 Jul 2023
Den Haag Dataset. Retrieved from https://ckan.dataplatform.nl/dataset/3d-stadsmodel-den-haag-2021-citygml
FZK Haus Dataset. Retrieved from https://www.citygmlwiki.org/index.php?title=KIT_CityGML_Examples. Accessed on 1 Jul 2023
GDAL. Retrieved from https://gdal.org. Accessed on 1 Jul 2023
GRASS. Retrieved from https://grass.osgeo.org. Accessed on 1 Jul 2023
Gröger G, Plümer L (2012) CityGML–interoperable semantic 3D city models. ISPRS J Photogramm Remote Sens 71:12–33. https://doi.org/10.1016/j.isprsjprs.2012.04.004
Kilsedar CE, Fissore F, Pirotti F, Brovelli MA (2019) Extraction and visualization of 3D building models in urban areas for flood simulation. In: The international archives of the photogrammetry, remote sensing and spatial information sciences, 2019 GEORES 2019–2nd international conference of geomatics and restoration, Milan, Italy, vol XLII-2/W11, pp 669–673. https://doi.org/10.5194/isprs-archives-XLII-2-W11-669-2019
Kolbe TH, Donaubauer A (2021) Semantic 3D city modeling and BIM. In: Shi W, Goodchild MF, Batty M, Kwan MP, Zhang A (eds) Urban informatics. The Urban Book Series. Springer, Singapore, pp 609–639. https://doi.org/10.1007/978-981-15-8983-6_34
Kutzner T, Chaturvedi K, Kolbe TH (2020) CityGML 3.0: new functions open up new applications. PFG 88:43–61. https://doi.org/10.1007/s41064-020-00095-z
Ledoux H, Ohori KA, Kumar K, Dukai B, Labetski A, Vitalis S (2019) CityJSON: a compact and easy-to-use encoding of the CityGML data model. Open Geospatial Data Softw Stand 4(4):1–12
León-Sánchez C, Giannelli D, Agugiaro G, Stoter J (2021) Testing the new 3D bag dataset for energy demand estimation of residential buildings. In: The international archives of the photogrammetry, remote sensing and spatial information sciences, 6th international conference on smart data and smart cities, Stuttgart, Germany, vol XLVI-4/W1-2021, pp 69–76
León-Sánchez C, Agugiaro G, Stoter J (2022) Creation of a CityGML-based 3D city model testbed for energy-related applications. In: The international archives of the photogrammetry, remote sensing and spatial information sciences, 7th international conference on smart data and smart cities (SDSC), Sydney, Australia, vol XLVIII-4/W5-2022, pp 97–103
Noardo F, Ellul C, Harrie L, Overland I, Shariat M, Ohori KA, Stoter J (2020) Opportunities and challenges for GeoBIM in Europe: developing a building permits use-case to raise awareness and examine technical interoperability challenges. J Spat Sci 65(2):209–233. https://doi.org/10.1080/14498596.2019.1627253

QGIS Plugins. Retrieved from https://plugins.qgis.org/plugins/citydb-tools/. Accessed on 1 Jul 2023
Qgis2threejs. Retrieved from https://github.com/minorua/Qgis2threejs. Accessed on 1 Jul 2023
Qt. Retrieved from https://www.qt.io/product/development-tools. Accessed on 1 Jul 2023
Railway Dataset. Retrieved from https://nervous-ptolemy-d29bcd.netlify.app/samplefiles/. Accessed on 1 Jul 2023
Rijssen-Holten Dataset. Retrieved from https://github.com/tudelft3d/Testbed4UBEM. Accessed on 1 Jul 2023
Rossknecht M, Airaksinen E (2020) Concept and evaluation of heating demand prediction based on 3D city models and the CityGML energy ADE—case study Helsinki. ISPRS Int J Geo–inf 9(602):1–19. https://doi.org/10.3390/ijgi9100602
Ruhdorfer R, Willenborg B, Sindram M (2018) Coupling of traffic simulations and semantic 3D city models. gis.Science, vol 3
SAGA. Retrieved from https://saga-gis.sourceforge.io. Accessed on 1 Jul 2023
SFS, Simple Feature for SQL Model. Retrieved from https://www.ogc.org/standard/sfs/. Accessed on 1 Jul 2023
Monteiro CS, Pina A, Cerezo C, Reinhart C, Ferrão P (2017) The use of multi-detail building archetypes in urban energy modelling. Energy Procedia 111:817–825. ISSN 1876–6102
Vienna dataset. Retrieved from https://www.wien.gv.at/stadtentwicklung/stadtvermessung/geodaten/viewer/geodatendownload.html. Accessed on 1 Jul 2023
Virtanen J-P, Jaalama K, Puustinen T, Julin A, Hyyppä J, Hyyppä H (2021) Near real-time semantic view analysis of 3D city models in web browser. ISPRS Int J Geo-Inf 10(138):1–23. https://doi.org/10.3390/ijgi10030138
Vitalis S, Ohori KA, Stoter J (2020) CityJSON in QGIS: development of an open-source plugin. Trans GIS 24(5):1147–1164. https://doi.org/10.1111/tgis.12657
Yao Z, Nagel C, Kunde F, Hudra G, Willkomm P, Donaubauer A, Adolphi T, Kolbe TH (2018) 3DCityDB–a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML. Open Geospatial Data Softw Stand 3(5):1–26. https://doi.org/10.1186/s40965-018-0046-7