Yao et al. Open Geospatial Data, Software and Standards (2018) 3:5
https://doi.org/10.1186/s40965-018-0046-7

Open Geospatial Data, Software and Standards

SOFTWARE Open Access

3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML

Zhihang Yao[1*], Claus Nagel[2], Felix Kunde[3], György Hudra[4], Philipp Willkomm[4], Andreas Donaubauer[1], Thomas Adolphi[2] and Thomas H. Kolbe[1]

Abstract

Over the last decade, more and more cities and even countries worldwide are creating semantic 3D city models of their physical environment based on the international CityGML standard issued by the Open Geospatial Consortium (OGC). CityGML is an open data model and XML-based data exchange format describing the most relevant urban and landscape objects along with their spatial and non-spatial attributes, relations, and their complex hierarchical structures in five levels of detail. 3D city models, which are structured according to CityGML, are often used for various complex GIS simulation and analysis tasks, which go far beyond pure 3D visualization. Due to the large size and complexity of the sometimes country-wide 3D geospatial data, the GIS software vendors and service providers face many challenges when building 3D spatial data infrastructures for realizing the efficient storage, analysis, management, interaction, and visualization of the 3D city models based on the CityGML standard. Hence, there has been strong demand for an open and comprehensive software solution that can provide full support of the aforementioned functionalities. The '3D City Database' (3DCityDB) is a free 3D geo-database solution for CityGML-based 3D city models. 3DCityDB has been developed as an Open Source and platform-independent software suite to facilitate the development and deployment of 3D city model applications. The 3DCityDB software package consists of a database schema for spatially enhanced relational database management systems (ORACLE Spatial or PostgreSQL/PostGIS) with a set of database procedures and software tools allowing to import, manage, analyze, visualize, and export virtual 3D city models according to the CityGML standard. Within this paper, the software suite is illustrated and explained in detail with respect to the related technical implementations and the underlying conceptual software design. Moreover, the utilization of 3DCityDB in different projects and practical application fields are also presented in this paper.

Keywords: GIS, GML, CityGML, 3D City modelling, relational database modelling, spatial database, 3D web visualization, Cloud computing

* Correspondence: zhihang.yao@tum.de
[1]Chair of Geoinformatics, Technische Universität München, Arcisstraße 21, 80333 Munich, Germany
Full list of author information is available at the end of the article

Introduction

Virtual 3D city models are used today for a wide range of applications like urban planning, environmental and training simulations, navigation, disaster management, energy assessment, and many more. In their paper Biljecki et al. [6] present a systematic overview about the different application fields and their respective requirements on the 3D city models. It was shown that most of the applications do not just need data about the 3D geometry and graphical characteristics, but also require semantic information like object types, thematic attributes as well as spatial and semantic interrelationships. Semantic 3D city models, hence, are virtual models of the physical urban environment, i.e. datasets representing the entities of the physical reality like buildings, streets, trees, bridges, and the terrain. In contrast to 3D models used in Computer Graphics they are structured (e.g. subdivided and attributed), according to thematic and logical criteria and not according to graphical or rendering considerations. The objects of a semantic 3D city model represent the respective real-world objects with their thematic, geometrical, topological, and appearance properties (cf. [21]).

In order to support the interoperable exchange and mutual usage of 3D city models over different applications as well as software systems, the Open Geospatial Consortium (OGC) has issued the international standard City Geography Markup Language (CityGML, cf. [17]). Many cities worldwide and even entire countries today have created and are maintaining CityGML-based 3D city models. CityGML defines an object-oriented data model of the most relevant urban objects like buildings, vegetation, roads, water bodies, terrain etc. The structural and spatial complexity of CityGML-based 3D city models can range from very simple to complex and nested entities. Each object can be spatially represented by multiple geometries of different types in 3D space (e.g. polygons, meshes, solids) in different levels of detail (LOD). While on the one hand applications and users benefit from the rich data model in many ways (cf. [43]), it puts high demands on storing, managing, and analyzing the complex structured data on the other hand. Above, 3D city models can be very large and single CityGML files for a bigger city or region can have from tens over hundreds gigabytes in size. However, applications and users need efficient tools to query, visualize, and update the 3D city model.

This paper presents a free 3D geo-database solution called '3D City Database (3DCityDB)', which especially addresses the challenges named above. 3DCityDB is an Open Source software suite allowing to import, manage, analyze, visualize, and export virtual 3D city models according to the CityGML standard, supporting both versions 2.0 and 1.0.

3DCityDB is not completely new. In fact, the development of 3DCityDB was started by the last author of this paper back in 2003 at the Institute for Cartography and Geoinformation at the University of Bonn. During the first phase the companies 'lat-lon' and '3D Geo' also contributed to the developments. Between 2006 and 2012 the development was continued by the Institute for Geodesy and Geoinformation Science at Technical University of Berlin. In 2012, the developer team at TU Berlin received the *ORACLE Spatial Excellence Award for Education and Research* from *ORACLE USA* for the work on 3DCityDB. Since 2013 the 3DCityDB and its tools are being further developed at the Chair of Geoinformatics of TU Munich (TUMGI) in collaboration with the companies 'virtualcitySYSTEMS GmbH' (VCS) and 'M.O.S.S. Computer Grafik Systeme GmbH' (MOSS) on the basis of a cooperation agreement. In order to simplify the inclusion and adoption of 3DCityDB within third-party commercial and Open Source products, the developers decided to switch from the LGPL3 license to the Apache 2.0 license in 2016. Today, third-party developers and software vendors from different domains are not only using the 3DCityDB, but they also contribute to the improvement of the functionalities and quality of the software tools and they create their own extensions.

While some basic concepts of the 3DCityDB have already been explained in an earlier paper [39], much has happened since then. Therefore, this paper gives more details on the one hand, and especially puts focus on the changes, new features, tools, and application examples that have been developed over the last ten years on the other hand.

The rest of this paper is structured as follows: Section 2 gives a brief introduction to the international standard CityGML. In addition, the essential aspects and approaches for realizing the efficient management using spatially-enhanced relational database management system (SRDBMS) are discussed in order to provide the foundation for designing a compact CityGML-compliant relational database schema for 3DCityDB. Section 3 presents the 3DCityDB software tools with details about the conceptual design and technical implementations. Section 4 at first highlights some application areas, use cases, research projects, and users currently employing 3DCityDB. Then it is shown, how 3DCityDB is being used to manage a large 3D city model of entire New York City, which has been created from Open Data. It is also shown, which role 3DCityDB plays as a technological core component in commercial software products of the two companies VCS and MOSS who have contributed to the 3DCityDB development for many years now. The last section draws the conclusions about the presented work and outlines the relevant aspects of our future research and development tasks.

Managing 3D city models within a 3D geodatabase

3D city modelling and CityGML

The City Geography Markup Language (CityGML) is an international standard for the interoperable representation and exchange of virtual 3D city and landscape models. CityGML defines a conceptual schema for the most relevant entities of the urban space like buildings, roads, railways, tunnels, bridges, city furniture, water bodies, vegetation, and the terrain. The conceptual schema specifies how and into which parts and pieces physical objects of the real world should be decomposed and classified. All objects can be represented with respect to their semantics, 3D geometry, 3D topology, and appearances in five predefined levels of detail (LOD 0–4). CityGML is formally specified using UML class diagrams, explanations of the object classes and attributes, and an XML schema for the file exchange format. CityGML is issued by the Open Geospatial Consortium (OGC). The first official version of CityGML was released in the year 2008 and the current version 2.0.0 was published in 2012 (cf. [17]).

In CityGML, all classes and data types are grouped into a number of thematic modules. The modules and their relationships are shown in the UML package diagram in Fig. 1. The *Core* module defines the basic CityGML components and is, hence, a mandatory package that must always be referenced by the packages of the other modules including *Building*, *Bridge*, *Transportation*, *CityObjectGroup*, *Appearance*, *Generic*, *CityFurniture*, *Relief*, *Vegetation*, *Tunnel*, *LandUse*, *and WaterBody*. Since CityGML is based on OGC's Geography Markup Language (GML) in version 3.1.1, the *Core* module has a dependency of the GML3 schema which must always be imported into the CityGML schemas. Another mandatory package is the Extensible Address Language (xAL) issued by OASIS, which maps the address formats of different countries onto a unified XML schema for encoding the address information of a building object in a standardised XML structure.

The geometric-topological model of CityGML is realized using a subset of the GML3 geometry model, which is based on the ISO 19107 standard 'Spatial Schema' for representing the spatial properties of real-world objects. Supported geometric primitives include *Point*, *Curve*, *Surface*, and *Solid,* which allow to represent spatial properties of city objects in different dimensions ranging from zero to three. Volumetric geometries are modeled using the well-known boundary representation (B-Rep, cf. [13]), where each *Solid* geometry object is defined by a closed outer shell (composed of individual *Surface* objects) and an arbitrary number of inner shells (representing any inclosures). The orientation of surfaces can be specified explicitly when using the geometry type *OrientableSurface.*

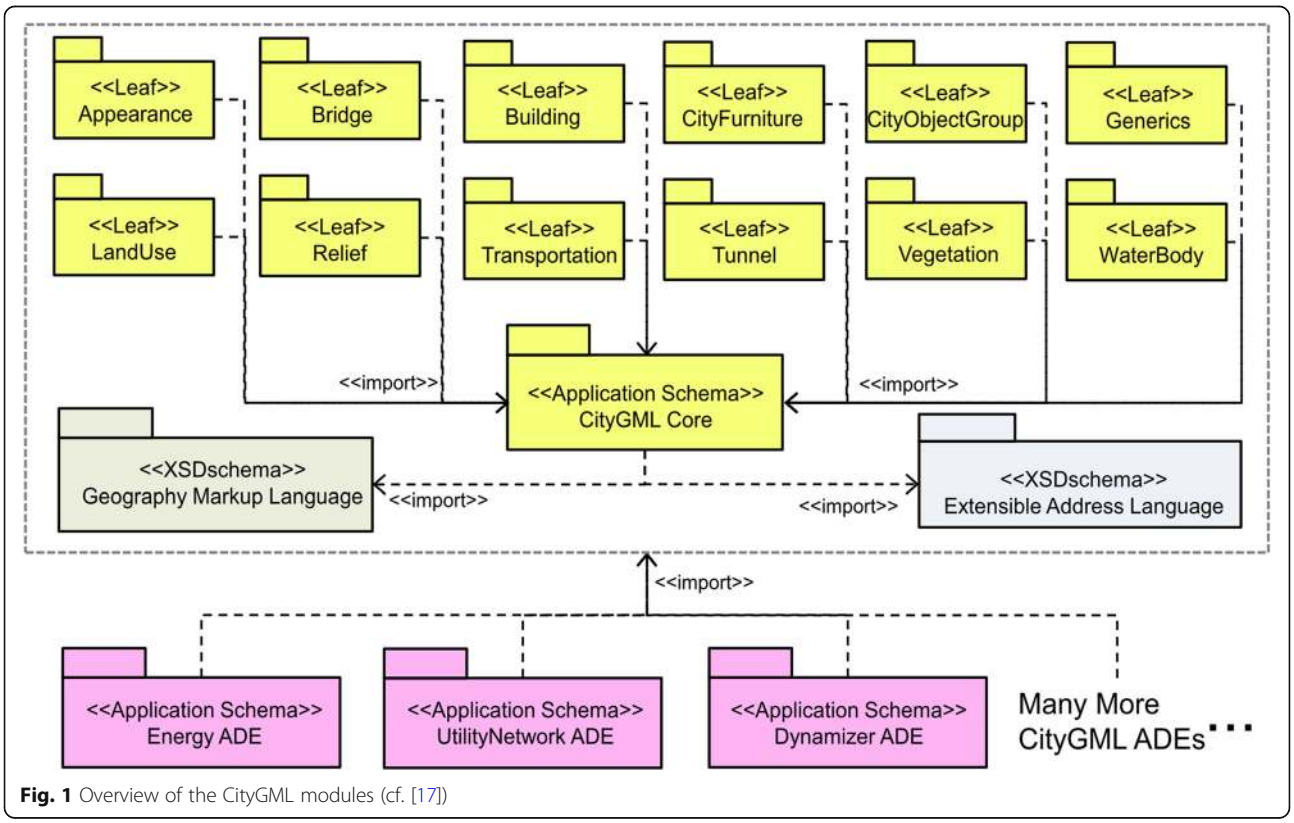

Fig. 1 Overview of the CityGML modules (cf. [17])

For each geometry type, more complex geometries with composite or aggregated hierarchies (cf. Fig. 2) can be constructed. The difference between aggregate and composite geometries lies in the topological relationships between the respective geometry components. For aggregate geometries such as *MultiCurve*, *MultiSurface*, and *MultiSolid*, the spatial relationships between components are not restricted and primitives can, hence, overlap, touch, or be disjoint. In contrast, a composite geometry like *CompositeCurve*, *CompositeSurface*, or *CompositeSolid* is a special case of the aggregate geometry which must be isomorphic to a single respective geometric primitive. This implies that the underlying elements must be topologically connected along their boundaries. In addition, the GML geometry type *GeometryComplex* can be used to represent a complex consisting of geometric primitives of different types (e.g. *Point* and *Curve*). The members of a geometric complex must not overlap and can touch at their boundaries only. *GeometryComplex* is being used in CityGML to represent the geometric network of streets and railways.

CityGML allows to assign appearances to individual surfaces (like *Polygons*), composite, and aggregate surfaces. Appearances can be specified by colours or textures, and each surface can be assigned any number of appearances. Textures are represented by raster images.

In order to represent topological relationships between geometries, CityGML utilizes the XLink concept according to the GML specification. Each geometry object can have a unique identifier and can form a shared part of different aggregate or composite geometries. For example, one polygon may be member of the outer shells of two solids in order to explicitly express that the two solids are touching along one side. The shared polygon is then not represented redundantly, but is referenced from the outer shell of the second solid by an XLink to the shared polygon.

CityGML is very flexible regarding the expression of spatial properties of semantic objects. For example, the geometry of a *Building* object may be given in any of the LODs 0, 1, 2, 3, and 4 (also simultaneously). Most LODs

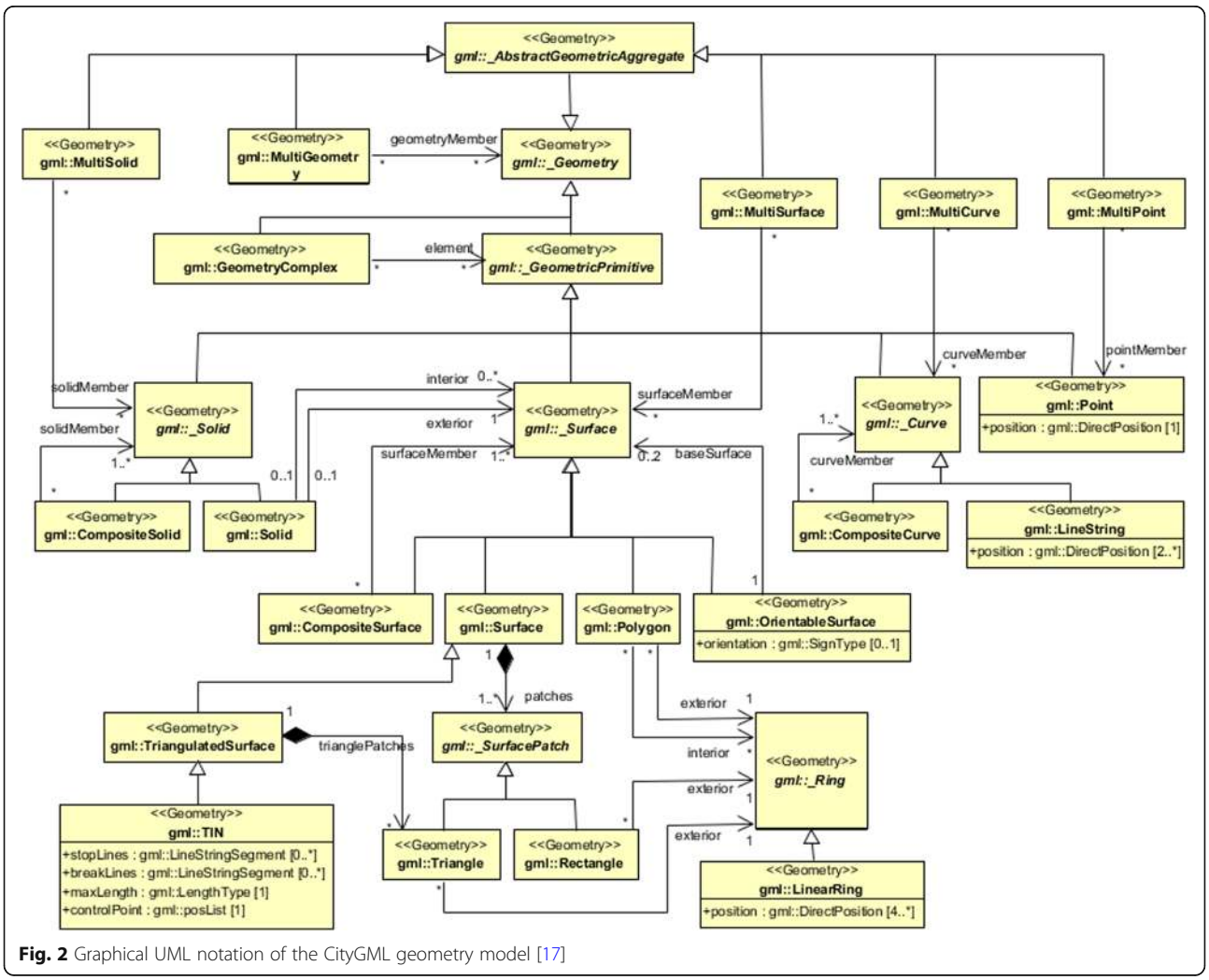

Fig. 2 Graphical UML notation of the CityGML geometry model [17]

allow representing the geometry by a *Solid*, a *MultiSurface*, or a combination of both. Besides geometrical decompositions, CityGML can also decompose objects semantically into parts. For example, a building can consist of building parts, which again can consist of roof, wall, and ground surfaces etc. When objects are decomposed in the same way regarding their semantic as well as their spatial structure, they are considered to be spatio-semantically coherent. This is illustrated for a building model in Fig. 3. Most of the LOD2 CityGML building models available today are semantically describing the wall, roof, and ground surfaces and additionally provide a solid geometry for the geometric representation of the building hull and its 3D shape. The semantic objects are usually used to query and analyse the building components and their thematic attributes, whereas the solid geometry represents the whole body and is useful for geometric calculations such as the building volume and surface areas. Both aspects of describing the building are complementary and provide a very flexible modelling structure ranging from simple geometric models to semantically rich models.

A 3D geodatabase for CityGML must be able to cope with all the aspects presented above. This means, each semantic object like a building or a tunnel can be decomposed into parts and subparts. Each semantic object can have a number of geometric properties of different geometry types and LODs. Some geometry elements can be shared from different aggregate geometries. Also semantic objects can be part of multiple semantic aggregate objects. Each surface can be assigned an arbitrary number of appearances. The geodatabase must also be able to handle appearance data like individual surface textures, which typically are given in binary image file formats (e.g. JPEG or PNG). All semantic objects have predefined thematic attributes and, in addition, can have an arbitrary number of generic attributes. Finally, since 3D city models cover large areas up to entire countries, the geodatabase must be able to manage the large data volumes and provide efficient access to the stored data for thematic and spatial queries.

Database solutions for CityGML

Besides *3DCityDB*, several other database solutions support the management of CityGML data. In the following, a selection of these software packages are listed, along with their major characteristics with respect to CityGML support. The Open Source software frameworks *deegree*[1] and *GDAL/OGR*[2] as well as the commercial software packages *CPA SupportGIS*[3] and *Snowflake GO LOADER*

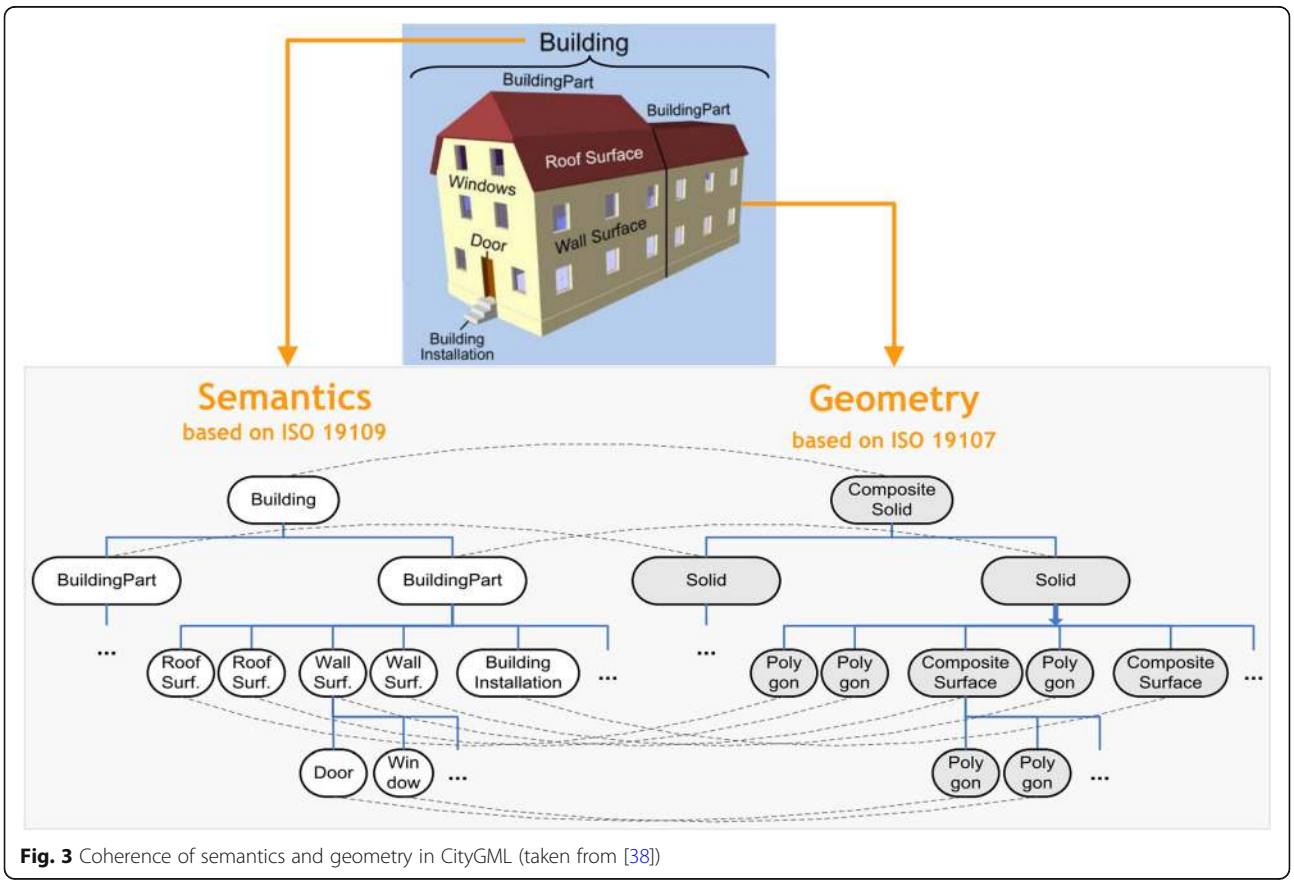

Fig. 3 Coherence of semantics and geometry in CityGML (taken from [38])

/ GO PUBLISHER[4] offer generic support for GML application schemas. Since CityGML is a GML application schema, these software systems are able to automatically create database schemas for storing CityGML data for various database management systems like *ORACLE Spatial* or *PostgreSQL/ PostGIS*, using the CityGML XML Schema definition files. For importing and exporting CityGML data sets into/from the database, *deegree* and *SupportGIS* offer an OGC Web Feature Service (WFS) interface whereas *Snowflake GO Loader* provides a desktop tool.

The named systems all extract CityGML data from CityGML files and insert the data into tables of spatially-extended relational database management systems. But, in recent years researchers have also examined different NoSQL solutions (de Souza Baptista et al. [11]). Document stores such as *BaseX*[5] for XML or *MongoDB*[6] for JSON seem like an obvious choice for storing instance documents of CityGML [31]. In fact, data ingest and retrieval is a lot faster than with RDBMS due to the smaller serialization effort [20]. While documents stores have their weakness with more complex queries including joins and spatial operations, they can be a great choice for web application backends sitting in between the RDBMS and a client. *GeoRocket,*[7] for example, decomposes CityGML XML files and stores the XML fragments in a (distributed) file system like *Amazon S3* or *MongoDB*. *GeoRocket* is available in an Open Source and a commercial version. Furthermore, solutions for storing CityGML data using the graph database *Neo4j*[8] have been presented by Agoub et al. [1] as well as by Nguyen et al. [33]. The latter software has been made available as Open Source software on *Github.*[9]

Relational database modelling for CityGML

There are strong reasons to employ spatially-extended relational database management systems (SRDBMS) to store and manage complex 3D city models. First, SRDBMS support all required geometry types and provide means for proper spatial indexing as well as for geometric and topological analyses. Second, SRDBMS can directly be used by most geoinformation systems (GIS) or spatially enabled ETL (Extract, Transform, Load) tools. As described above there exists a variety of non-relational databases like object-oriented databases, document-oriented databases, and graph databases, which are increasingly investigated and employed in many application fields (cf. [35]). However, they are currently still more or less limited in their capabilities and performance regarding spatial operations and coordinate transformations, which are of great importance for the enterprise use in GIS applications (cf. [1]). Therefore, SRDBMS such as the commercial software ORACLE Spatial/Locator and the Open Source software PostgreSQL with PostGIS extension play a major role for GIS due to their extensive capabilities in handling 3D spatial data.

The conceptual solution for handling object-oriented data models like CityGML in SRDBMS can be abstracted to solving the problem of mapping the object-oriented data model onto a relational data model. This has been extensively studied and discussed in literature over the past 25 years. Golobisky & Vecchietti [16] summarized the fundamental concepts for deriving relational database schemas using different mapping rules according to the source UML class structures. For example, a class shall be mapped onto one table where each row should represent an instanced object of the respective class. Thus, the mapped table shall have at least one primary key column which can be named as “ID” and defined with the long integer data type for storing the object identifier which must be unique within the table. Additional columns can also be added to the mapped table for storing the spatial and non-spatial attribute values of the respective class objects. To handle the class associations in relational models, a foreign key constraint or an associative table in case of M:N relationship shall be utilized to link the tables mapped from the associated classes. Moreover, the inheritance relationship between two classes can either be implemented using a foreign key constraint to link the subclass and superclass tables by joining their primary keys or mapped to a table that represents the two inherited classes at the same time. Further discussions and comparison of, among others, the aforementioned mapping rules are given in [19].

However, although these mapping rules from the literature allow to map CityGML data model onto a relational database model, they may easily lead to a large number of database tables with many join relations. An analysis of the existing relational database systems indicated that a more compact database schema is much more efficient for querying and processing of large and complex-structured data to facilitate good performance when interacting with the database in a real-time application (cf. [39]). To reach this purpose, the CityGML database schema shall result from a careful manual process by identifying and simplifying the complex CityGML classes and data types and mapping them onto fewer tables with respect to the database complexity, operating performance, and semantic interoperability. Concerning this requirement, [24] proposed a set of fine-grained mapping rules, which have been successfully adopted for designing the 3DCityDB database schema and are briefly reviewed in the following subsections.

Mapping an inheritance hierarchy onto one table

With this approach, multiple CityGML classes belonging to an inheritance hierarchy can be mapped onto one

single table. For example, a table named CITYOBJECT can be used for the instance objects and their attribute values of the GML class *_GML*, and *_Feature* as well as the CityGML class *_CityObject* (cf. Fig. 4). For each CityGML top-level class like *AbstractBuilding*, *AbstractBridge* and *AbstractTunnel* etc. a separate table associated with the CITYOBJECT table shall be created to hold the feature attributes. This way, the CITYOBJECT table can be used as a central registry of all the CityGML top-level features and allows for rapidly retrieving a list of CityObjects through a query on their attributes like spatial extent via a user-selected bounding box.

Mapping classes at the same inheritance hierarchy level onto one table

This mapping approach utilizes only one table to represent multiple classes which are subtyped from a common class and at the same time belong to the same inheritance hierarchy level (cf. Fig. 5). This way, the subclasses are logically mapped onto the super class table, such that the retrieval of data contents of all subclasses just needs to perform only one query on the table in order to avoid multiple table joins for speeding up the overall performance. To distinguish the different types of instance objects stored in the table, an additional column *OBJECTCLASS_ID* is required which can store a numeric value in each row for representing the respective class type. This type information is static and can be well documented in an additional table OBJECTCLASS whose primary key values are used for enumerating the object class IDs and referenced by the OBJECTCLASS_ID columns of the class tables. Moreover, additional columns for describing the meta-information like class name and parent class name etc. of each feature class can be added to the OBJECTCLASS table which allows third-party applications to directly retrieve the class information from the database for interpreting the queried feature objects.

Note that this mapping approach is not generally applicable since it also has its own usage limitations in some particular cases. For example, if the subclasses have very different attributes or associations to other classes, a large number of empty cells will occur in the database table and can result in a lower storage efficiency, especially when the number of subclasses is increased. Considering this situation, the utilization of this mapping approach shall satisfy some certain conditions regarding the model definitions and structures which may typically have the following characteristics:

- The super class shall be an abstract class that holds all attributes and associations which will be inherited by the concrete subclasses.
- Every of the subclasses shall not have any further attributes or associated with other classes.

With these conditions, the storage efficiency can be retained to the highest degree, because only one additional column e.g. OBJECTCLASS_ID storing the class type information needs to be added to the table. An analysis of the CityGML model structure shows that this mapping

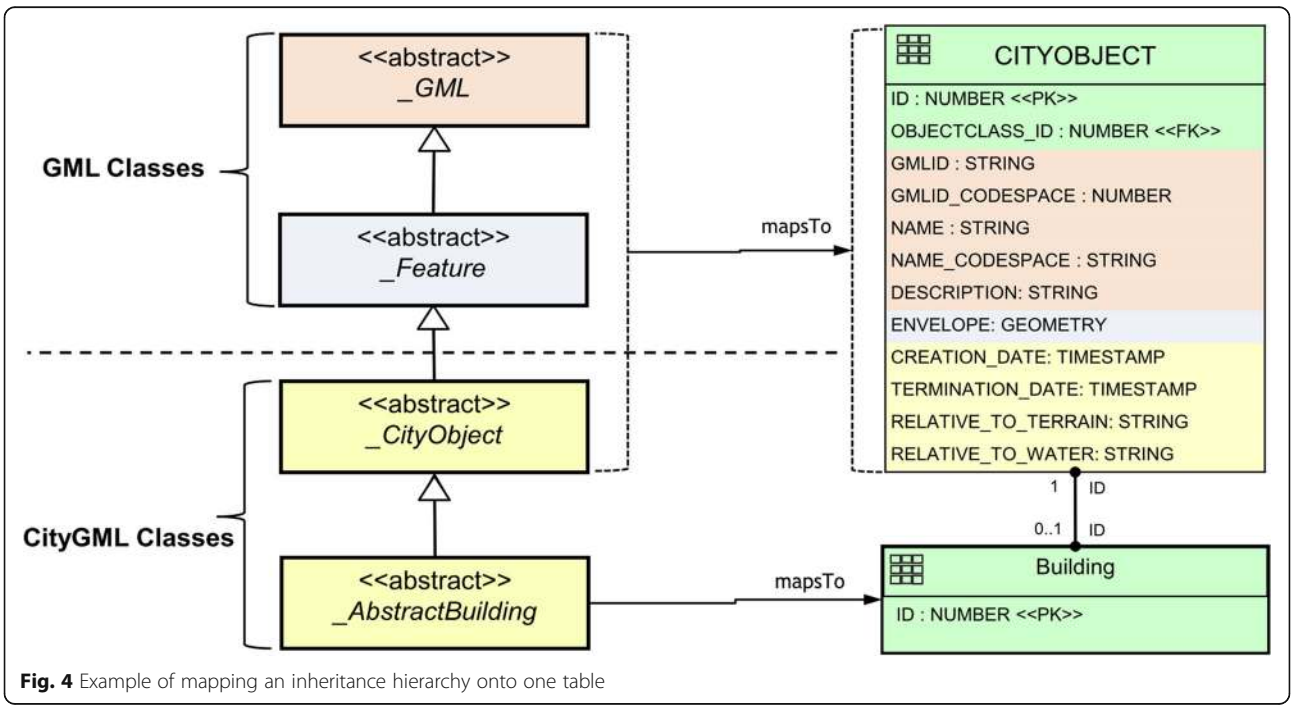

Fig. 4 Example of mapping an inheritance hierarchy onto one table

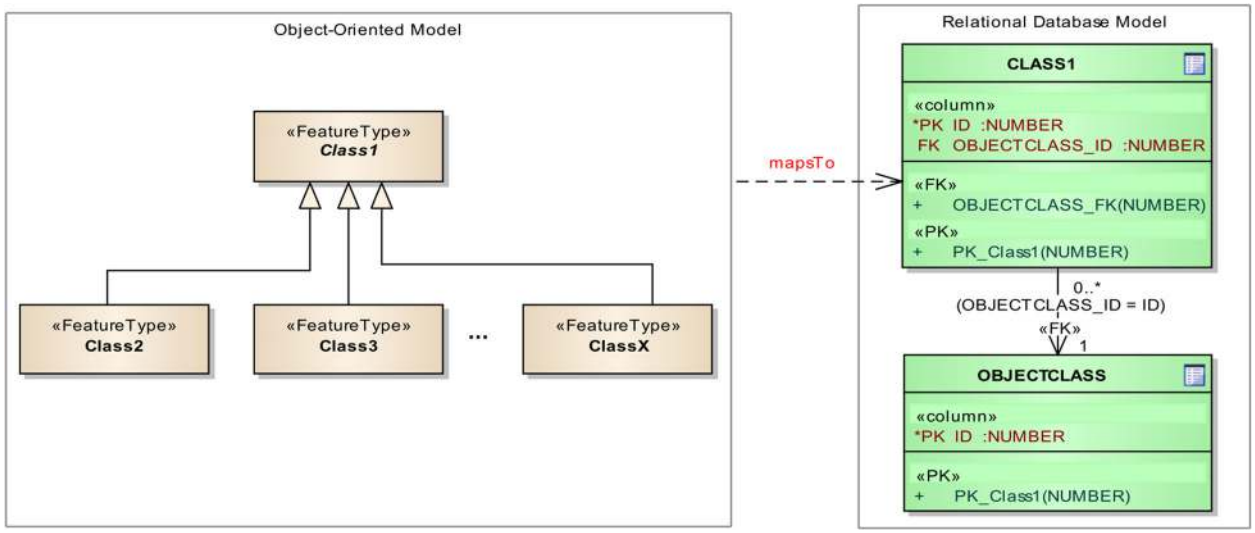

Fig. 5 Example of mapping multiple classes onto one table

approach can be well applied to the relational database modelling for CityGML to improve the overall database performance and efficiency. For example, the thematic surfaces like wall surfaces, roof surfaces, and ground surfaces etc. of each feature type like *Building*, *Tunnel*, and *Bridge* are abstracted to an abstract class called *_BoundarySurface* which holds the relevant attributes and association information. For each type of thematic surface, a concrete class i.e. *WallSurface*, *RoofSurface*, and *GroundSurface* etc. being a subtype of the class *BoundarySurface* is defined individually. This model definition exactly satisfies the afore-outlined conditions of this mapping approach allowing for realizing the fast data retrieval. For example, a typical query being usually applied is the export of a semantically rich building (LOD > = 2) to a 3D graphics format. In this case, the thematic surfaces like roof and wall surfaces forming the outer shell of the building object can be directly queried by joining the surface table with the building table instead of using multiple database joins.

Mapping aggregations and compositions onto one table

In objected-oriented data models, recursive aggregation relations of features can be properly modelled by means of a well-known design pattern called '*Composite Pattern*' (cf. [14]) which typically uses three interrelated classes (cf. Fig. 6) for constructing a tree-like data structure. According to the concept of this design pattern, each instance of the class *CompositeObject* can contain an arbitrary number of, but at least one instance of the class *BasicObject* or *CompositeObject*. The *BasicObject* corresponds to the leaf in the aggregation hierarchy and

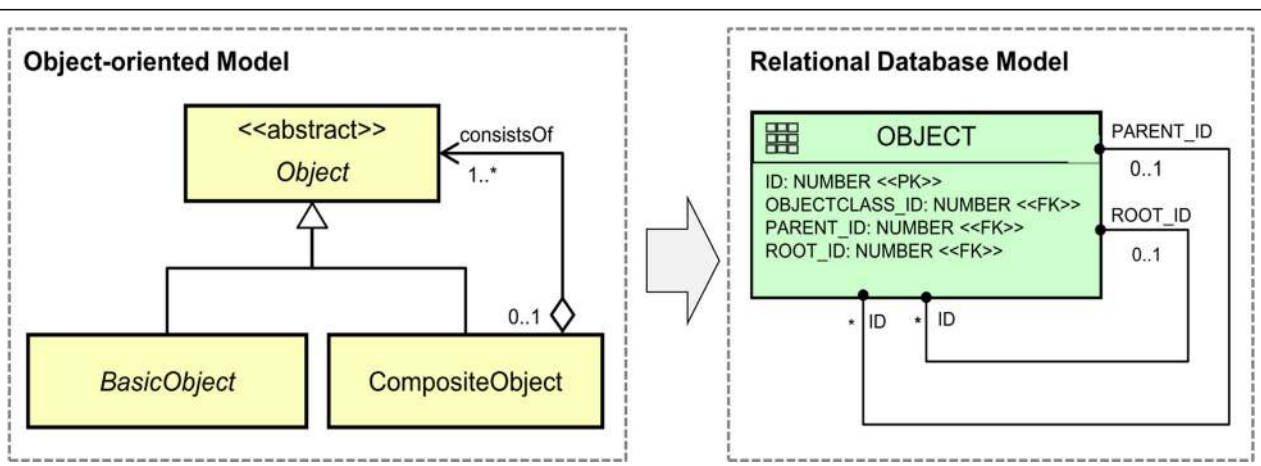

Fig. 6 General idea for the mapping of an object-oriented model with the composite pattern onto an efficient relational database model

shall not have child components. The conventional solution for the mapping of such data model onto relational structure is to use a foreign key for joining each object with its parent object to querying all the aggregated objects. In this case, recursive database queries must be performed which may cause high performance cost, especially if the recursion depth is unknown.

In order to achieve good performance when retrieving the elements of a tree of objects, a specific optimization approach has been developed. The key idea of the database design is to utilize a single database table for the mapping of all the involved feature classes along with their inheritance relationships. A foreign key column PARENT_ID is used for representing the composition relationship. Additionally, this database table receives a foreign key column ROOT_ID which holds the ID of the root element of each composite hierarchy and hence allows for fast retrieval of all its child elements by querying on the attribute ROOT_ID in order to avoid time-costly recursive database joins. Moreover, since three classes are mapped onto one table, an additional column OBJECTCLASS_ID is required for supporting the automatic determination of class affiliation information. This mapping approach can benefit the relational database modelling for the CityGML data modules like *Building*, *Bridge*, and *Tunnel*.

Mapping CityGML's B-rep geometries onto a single table

The optimization approach for the mapping of composite pattern can also be applied for the handling of complex data types like the B-Rep geometries such as aggregated/composite surfaces and solids (cf. Fig. 7).

With this optimization step, all surface-based geometry types can be represented in a simplified data model according to the composite pattern (cf. the previous subsection) and consequently mapped onto a compact table allowing for high-performance database query of all the geometry elements of an aggregation hierarchy. Instead of using a class ID column, the class affiliation is realized using a number of flag columns for characterizing the different types of geometry and aggregation. For example, the IS_SOLID distinguishes between surface and solid geometry, and the IS_COMPOSITE can be used to determine whether this is an aggregate (e.g. MultiSolid, MultiSurface) or a composite (e.g., CompositeSolid, CompositeSurface) geometry element. This approach offers semantic clarity of the table structure and also allows to manage the surface and solid geometries within a single table at the same time. Consequently, the interaction and query of the geometry data from this table becomes much simpler. For example, if a feature object owns a *MultiSurface* or *MultiSolid* property, a foreign key column can be added to the class table referencing to the primary key column of the geometry table to access the geometry data. Furthermore, since each surface geometry element is explicitly stored in a tuple, it can be easily augmented with appearance information like texture images, colors, or materials by associating the geometry table with the appearance data table via the corresponding row ID.

Implementation of the 3DCityDB and its tools

The relational database schema is the core component of the 3D City Database (3DCityDB). The mapping rules introduced in the previous section were used for a manual mapping of the object-oriented data model of CityGML onto a relational database schema. The database schema employs spatial datatypes to represent geometric properties of CityGML objects. Currently, 3DCityDB supports two different spatial relational database management system (SRDBMS), the first is the commercial SRDBMS ORACLE Spatial/Locator and the second is the Open Source SRDBMS PostgreSQL with the PostGIS extension.

Compared to other systems discussed in subsection 2.2 the resulting database schema is more compact in the sense that 3DCityDB requires fewer tables with respect to those systems that fully automatically generate the database schema. The manual mapping also ensures that table and attribute names are identical (or at least similar) to the respective class and property names of the CityGML data model. The entire relational schema is explained in full detail in the 3DCityDB documentation. This makes it easy for users who directly need to interact with the tables in the database. In this way, large and complex CityGML datasets can be not only efficiently managed, analyzed, and queried within a central data repository using the database language SQL, but they can also easily be accessed by external GIS and ETL software applications to e.g. enrich a 3D city model by adding information to the corresponding database tables.

In addition to the database schema, stored procedures are provided in PL/SQL (ORACLE) or PL/pgSQL, respectively, to perform basic tasks like the computation of bounding volumes of 3D objects and the entire 3D city model, the deletion of objects, or the management of spatial indexes. Furthermore, 3DCityDB comes with a number of software tools (cf. Fig. 8), which allow for the flexible extraction of 3D city model data subject to thematic and spatial filter criteria in the CityGML format, in 3D visualization formats like KML, COLLADA, and glTF as well as in spreadsheet formats.

Database stored procedures

The first important 3DCityDB software toolkit are the stored procedures which are automatically installed on the database side during the setup procedure of a 3DCityDB database instance. They are written in the database

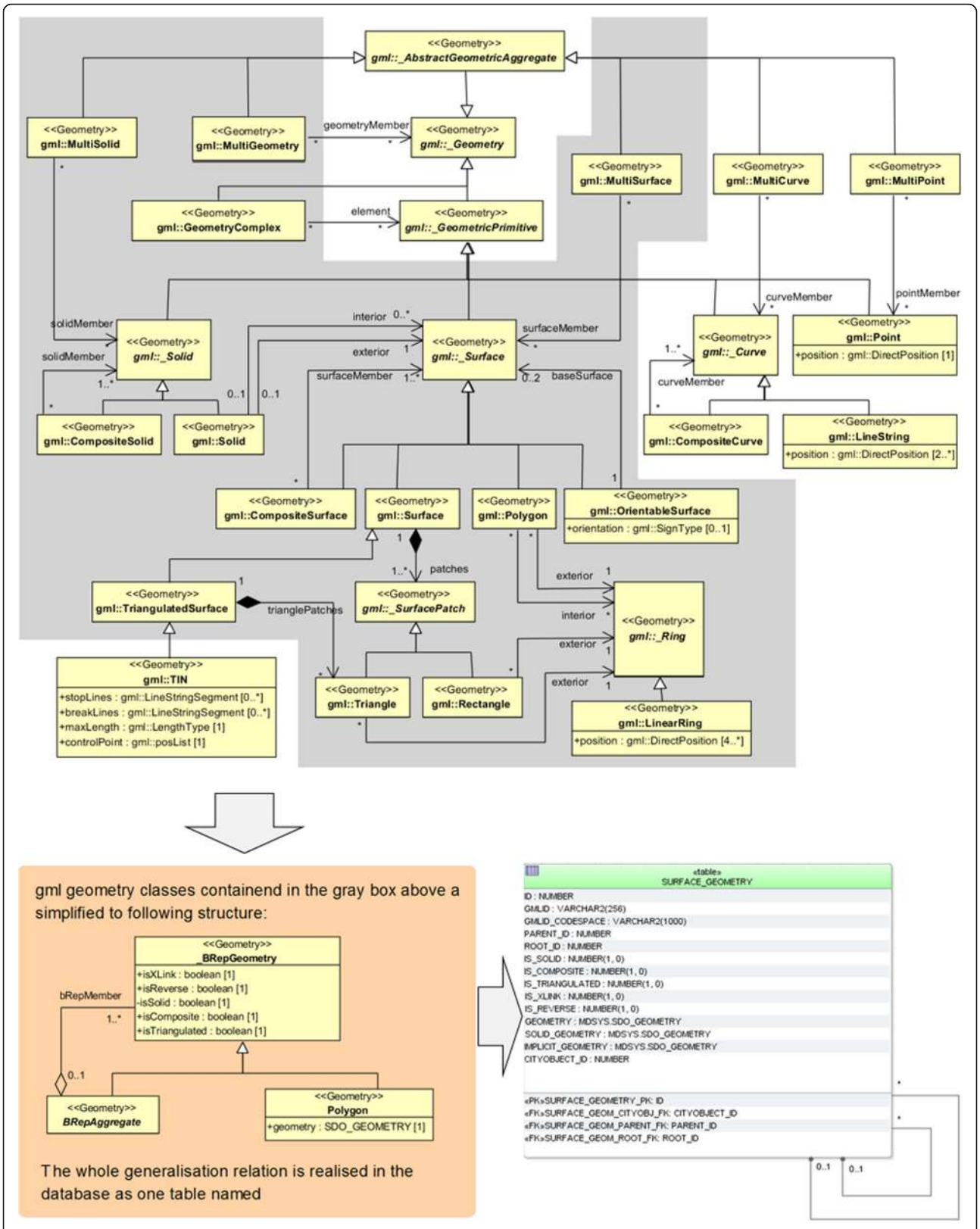

Fig. 7 GML geometry types being used in the CityGML standard (cf. [24])

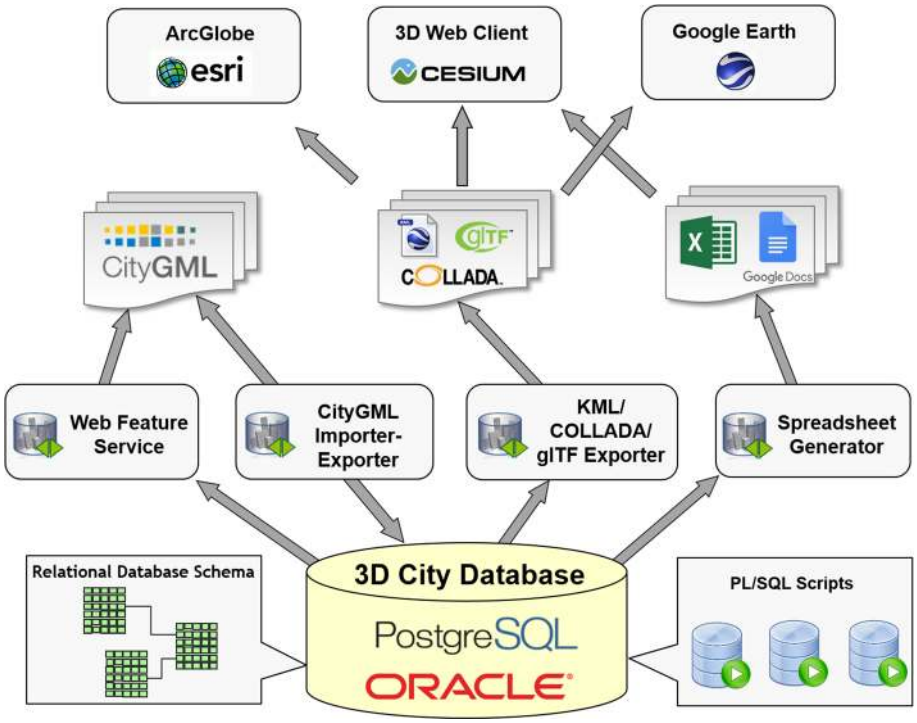

Fig. 8 Key components of the 3DCityDB Software Suite

scripting language PL/SQL for ORACLE and in PL/pgSQL for PostgreSQL. Since some of these procedures expose related functionalities, they are organized into six packages, namely `SRS`, `STAT`, `INDX`, `ENVELOPE`, `DELETE` and `UTIL` (cf. Fig. 9). The package `SRS` mainly provides a useful function allowing to transform the stored 3D spatial data into another coordinate system during the export process. The package `STAT` can be applied to count all entries in all data tables and generate a report listing the number of rows in the individual data tables. The package `DELETE` consists of several functions allowing to delete single or multiple city objects

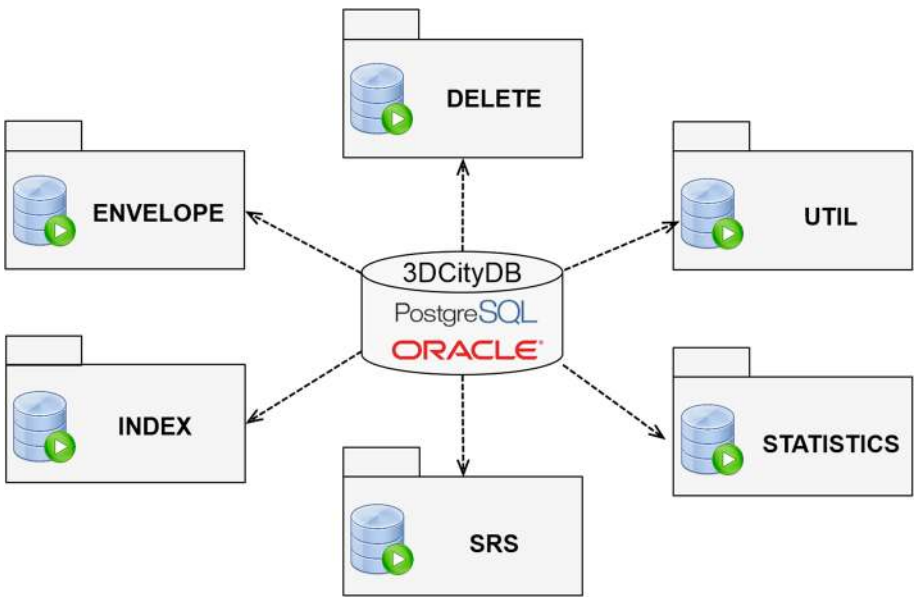

Fig. 9 Overview of the 3DCityDB database procedure packages

from the database according to the given row ID or an object class ID in the CITYOBJECT table. Each function automatically takes care of the integrity constraints between the related tables to properly clean up the corresponding data contents. The package `ENVELOPE` provides functions for calculating the maximum 3D bounding volume of a city object according to its geometry contents and also allows for updating the ENVELOPE attribute of the respective city object with the calculated value. In order to ensure data consistency, it is hence very important to run this function whenever one of the geometry representations of a city object has been changed. The package `INDEX` contains the function for activating and deactivating the ordinary indexes as well as the spatial indexes on those columns that are frequently used for performing queries. This allows to deactivate the spatial indexes before running a CityGML import in case of very big datasets and to reactive the spatial indexes afterwards. This way, the import process is able to run much faster than with enabled spatial indexes. The last package `UTIL` offers various utility functions i.e. checking the database version information, performing affine transformation on the 3D coordinates, determination of the mapping relationships between 3DCityDB tables and CityGML classes etc. All functions are explained in detail within the 3DCityDB documentation.

CityGML import/export tool

One of the major software tools included in the 3DCityDB software package is the CityGML Importer/Exporter, which is a Java-based desktop application serving as a front-end for the 3DCityDB database with a graphical user interface (cf. Fig. 10). The Importer/Exporter allows for high-performance reading and writing of large CityGML datasets with arbitrary file sizes. For reading and writing CityGML documents, a low-level Java API called citygml4j[10] is employed which provides a convenient way to process and validate CityGML datasets against the CityGML and xAL schema definition files. This is realized by using the Java XML Schema binding compiler (xjc) included in the Java™ Architecture for XML Binding (JAXB[11]) to compile the CityGML, GML, and OASIS xAL models to a set of corresponding Java

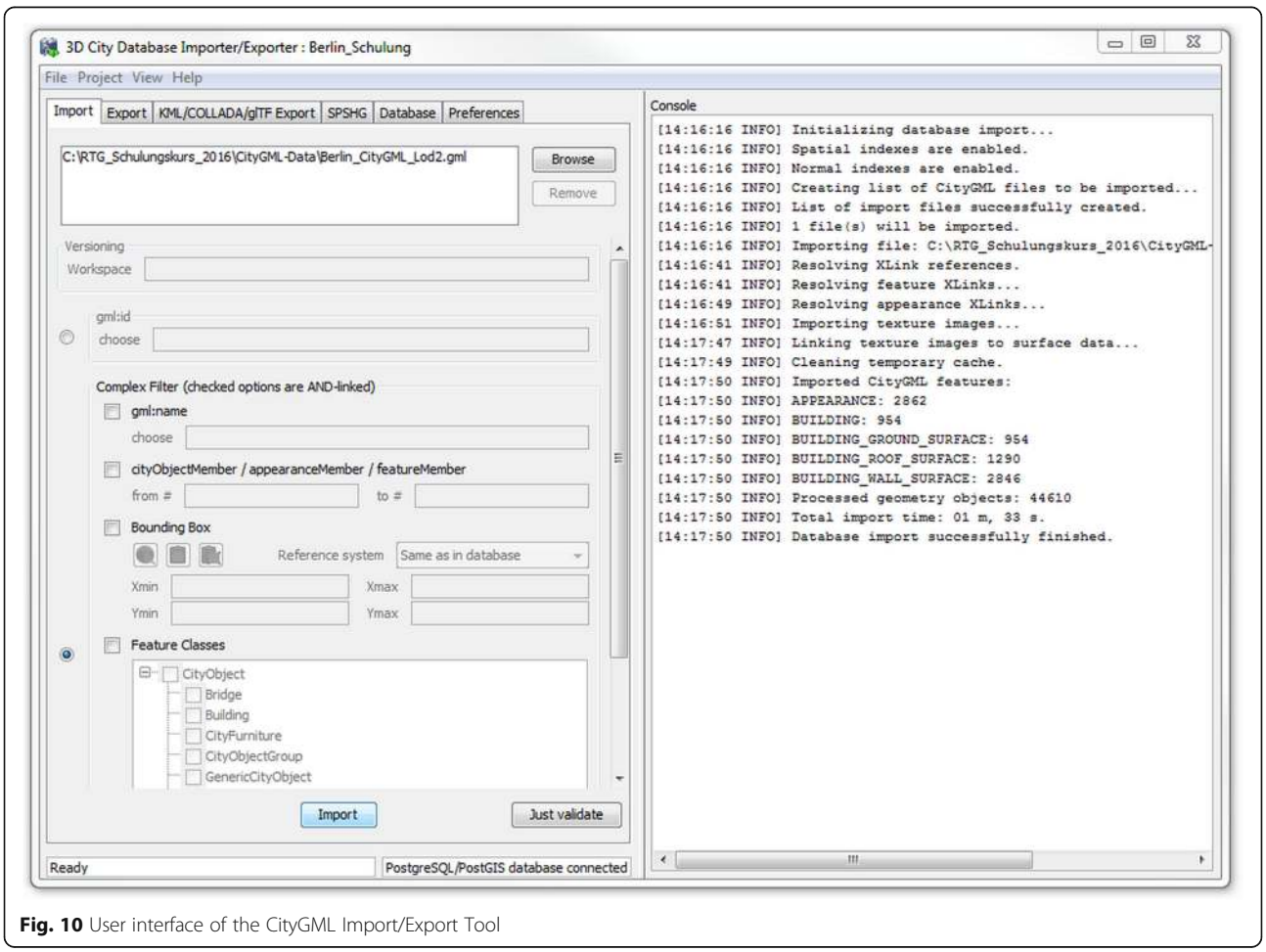

Fig. 10 User interface of the CityGML Import/Export Tool

classes which are kept static and provide an object-oriented view for handling CityGML features along with their properties in Java during runtime. The import of CityGML datasets works chunk-wise. Each CityGML top-level feature element (e.g. a Building, WaterBody, Street object) is read individually and automatically transformed to a Java object according to the corresponding class definition in the citygml4j context. All these Java objects are organized within a buffered queue and are successively imported into the database concurrently by means of a multi-threaded approach to increase the overall processing performance. In order to make full use of multiple CPU cores and to avoid the thread life-cycle overhead, a thread pool is employed for dynamically managing and controlling the number of the threads according to the number of the available processors of the hardware being used.

An important step of the import process is the resolution of XLinks in the CityGML datasets. This addresses the issue that a CityGML feature or geometry could be referenced by other ones using GML's XLink mechanism. Since some CityGML objects in the beginning of a CityGML file can point to objects located at the end of the same CityGML file, resolving the XLinks usually requires reading the entire CityGML dataset into main memory. This, however, will eventually cause a memory overflow when dealing with very large CityGML datasets (> > 4GB), which is not so uncommon today. For example, the CityGML file representing the one million building models from the New York City dataset presented in section 4.1 has a size of 32 GB. To overcome this problem, CityGML features and geometries are first read and imported, neglecting all the XLink reference information which, however, is temporarily stored into the database. When the first import process is done, the XLink reference information stored in the database will be resolved again and written into the corresponding CityGML data tables to complete the entire CityGML import process. This is illustrated in the center and upper part of Fig. 11.

During the export process, a list of GMLIDs of the top-level features satisfying the user-defined filter criteria i.e. feature class types and geographic bounding box etc. are first queried from the database. In the subsequent step, a worker pool containing a number worker threads is constructed and each GMLID is processed by a worker thread for creating a citygml4j object from the CityGML feature content queried from the respective database tables. In the last step, the citygml4j objects are marshalled to XML elements and written to a CityGML

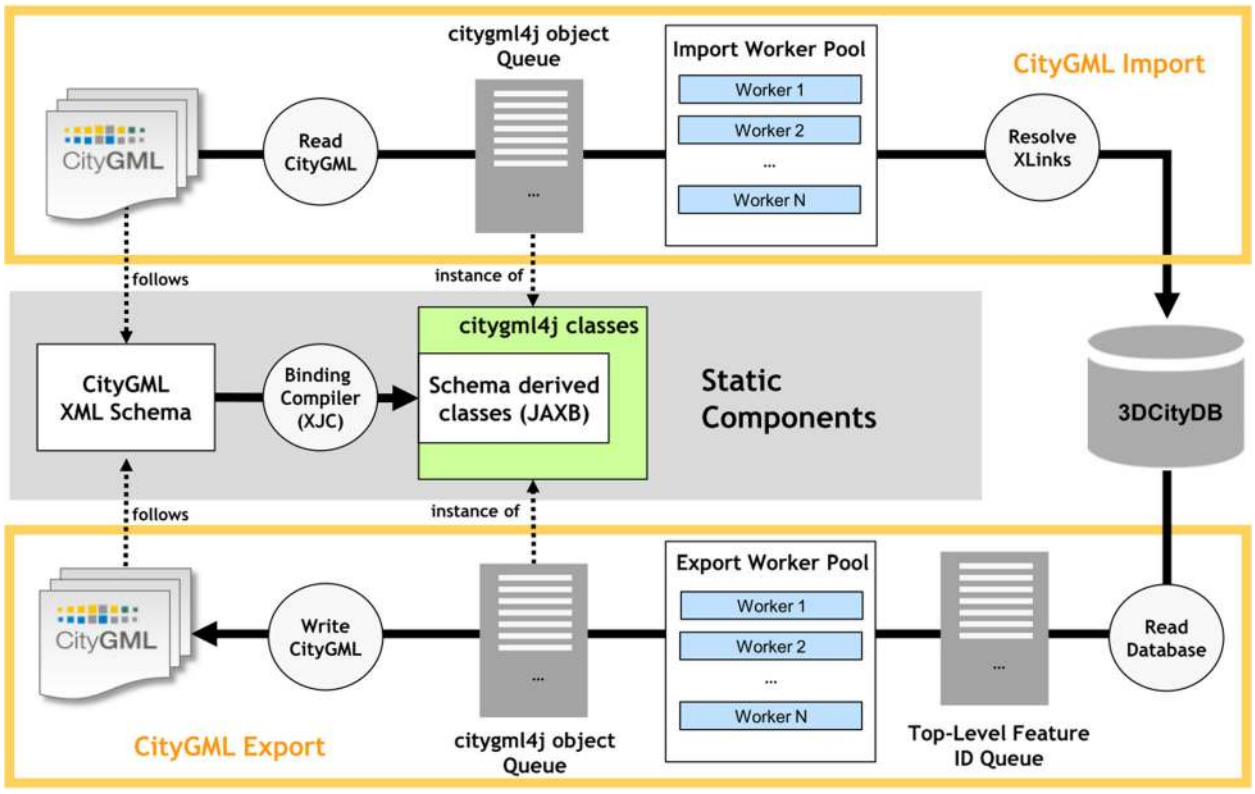

Fig. 11 Software structure of the CityGML Import/Export Tool (cf. [39])

instance document. This is illustrated in the center and lower part of Fig. 11.

The 3DCityDB Importer/Exporter allows just to validate CityGML datasets against the respective XML schemas. If datasets should also be validated regarding their semantic and geometrical correctness or regarding their geometric-topological consistency, external tools like CityDoctor[12] (cf. [42]) or val3dity[13] (cf. [28]) must be used.

Web feature service

Although the CityGML Importer/Exporter offers extensive functionalities for reading and writing CityGML documents from the database, it is only applicable in a desktop environment. To overcome this limitation, 3DCityDB comes with a web service implementation extending the core modules of the CityGML Import/Export tool and allowing for web-based access to the 3D city objects stored in the database. The 3DCityDB web service implements the OGC Web Feature Service 2.0 (WFS 2.0) Interface Standard, which provides a standardized and open interface for requesting geographic features across the Web using platform-independent requests using a simple HTTP call. Thus, the 3DCityDB users are no longer restricted to use the Importer/Exporter tool for the data retrieval only but can also directly use the WFS interface via a web browser or a WFS-aware applications. Please note, that the Open Source version of the 3DCityDB WFS implements the WFS 2.0 Simple Profile only, which is enough to retrieve objects by their GMLID. A full, transactional WFS-T is commercially available, though.

The 3DCityDB WFS is implemented as a *Java web application* based on the *Java Servlet* technology and must, hence, be run in a Java servlet container like Apache Tomcat on a web server. The workflow of executing a WFS procedure is illustrated in Fig. 12. When sending a request to the WFS server to retrieve certain CityGML features, the 3DCityDB WFS servlet will first capture and parse the request information and translate it to a corresponding database query to obtain a list of GMLIDs of the those CityGML top-level features that satisfy the filter criteria encoded in the WFS request messages. These feature IDs will be then handed over to the CityGML Import/Export module which utilizes its pre-complied citygml4j/JAXB classes as well as the multi-threading API for efficiently querying and generating the corresponding CityGML XML elements. Finally, these XML datasets will be returned as a response of the WFS request and can be directly downloaded or used by a WFS client.

KML/COLLADA/glTF exporter

The CityGML Import/Export tool has an extensive plugin API enabling developers to create plugins for the Import/Export tool that dynamically extend the existing functionalities. A plugin can be easily installed by copying its mandatory files – including the compiled JAR file and related libraries etc. – into a specific subfolder of the Importer/Exporter installation directory. The plugin will be automatically launched when starting up the Importer/Exporter. In addition to the functional aspects, a plugin may also have its own GUI that can be embedded and rendered in the main operations window of the CityGML Import/Export tool. Per default, the 3DCityDB comes with a plugin called KML/COLLADA/glTF Exporter. Using this plugin, the spatial contents of CityGML features can be directly exported to the 3D visualization formats KML, COLLADA, and glTF. These formats were chosen, because they allow to directly view the visualization models with virtual digital 3D globes like Google Earth, ESRI ArcGlobe & ArcGIS Pro, NASA Worldwind, and CesiumJS. In order to be able to render

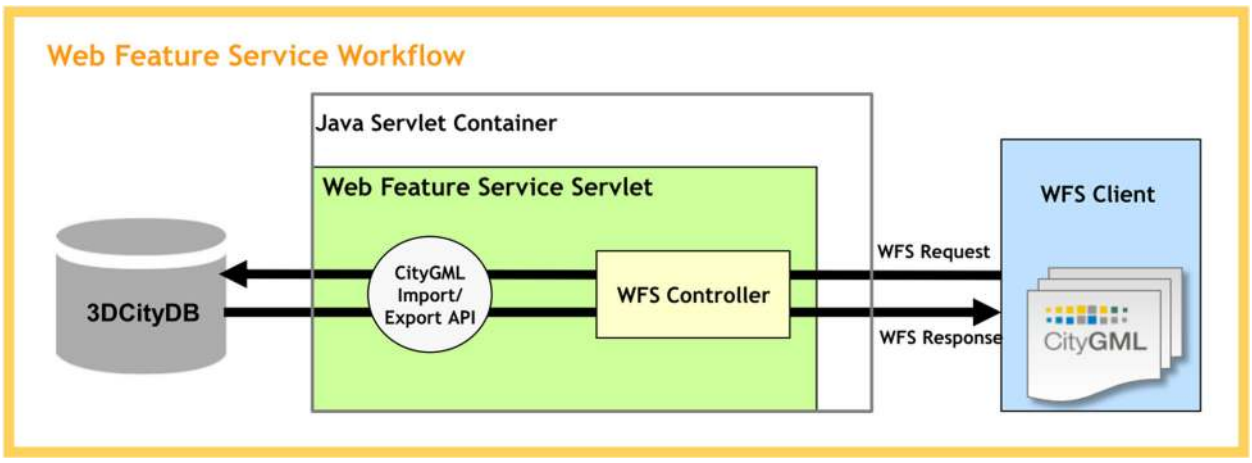

Fig. 12 Implementation of the 3DCityDB Web Feature Service

3D city models with acceptable frame rates for interactive exploration, larger datasets need to be geographically subdivided into tiles. 3DCityDB supports tiling as well as the generation of some master file (in KML or JSON), which loads the respective tiles according to the current camera perspective on demand. 3DCityDB can also adjust the base heights of individual 3D objects during export in order to align them with a digital elevation model (DEM). This works automatically with the DEM used in Google Earth. Users can choose which LOD should be exported and which appearance should be used. For 3D objects that do not have appearance information attached, users can specify simple colour and styling rules. Also information balloons containing thematic data for each 3D object can be generated.

The export process (cf. Fig. 13) follows a similar logic to that of the CityGML exports. In the first step, the GMLIDs of the features are first queried from the database according to the user-defined filter criteria and then passed to a worker pool which is implemented using the multi-threading API of the Import/Export tool. Depending on the hardware being used, this worker pool is able to dynamically create an optimal number of worker threads each of which is responsible for taking one GMLID after another from the waiting queue and querying the respective spatial data contents from the database for creating a KML/COLLADA java object. The class definition of the java object is pre-generated by means of the XML Schema binding compiler (xjc) for compiling the XML schema definition files of KML and COLLADA to the corresponding Java classes and allowing for directly marshalling the created java objects to the corresponding XML elements using the JAXB library. Furthermore, the creation of the glTF models can be done in a subsequent processing step, which utilizes a third-party software tool called *Collada2glTF* for converting the COLLADA models to glTF.

Spreadsheet generator plugin

The 3DCityDB offers another plugin called *Spreadsheet Generator,* which can be installed optionally. This plugin can be used to generate simple reports about objects of the stored 3D city model. It exports the thematic contents of a 3D city model from a 3DCityDB instance to a simple table format, either to a CSV or a Microsoft Excel file, where the first column lists the unique identifiers (GMLIDs) of the exported city objects each of which refers to one spreadsheet row. The generated spreadsheets can be opened using a spreadsheet application like Microsoft Excel and Open Office Calc etc. or uploaded to a Cloud-based online spreadsheet service like Google Spreadsheets or Microsoft OneDrive, which allows for interactive, collaborative web access and easy data exploration by multiple users. The 3DCityDB-Webmap-Client can link 3D visualization models created by the KML/COLLADA/glTF Exporter with Google Spreadsheets or Google Fusion Tables for interactive display of object attributes (see section 3.6 below for more details).

The spreadsheet generation process is similar to the workflow of the KML/COLLADA/glTF export and is also implemented based on a multi-threading programming by means of the concurrency API of the Importer/Exporter tool (see Fig. 14). Each thread in the worker pool is dedicated to query the thematic contents of a top-level feature and map the results onto a table row

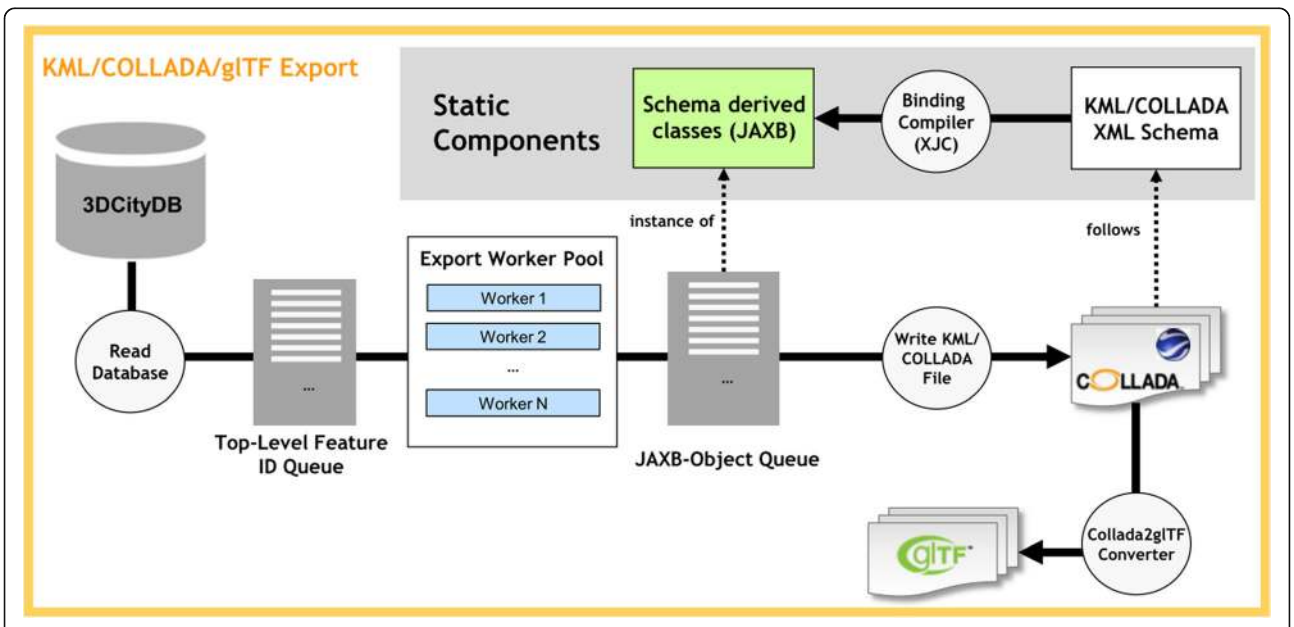

Fig. 13 Workflow of generating KML/COLLADA/glTF visualization models

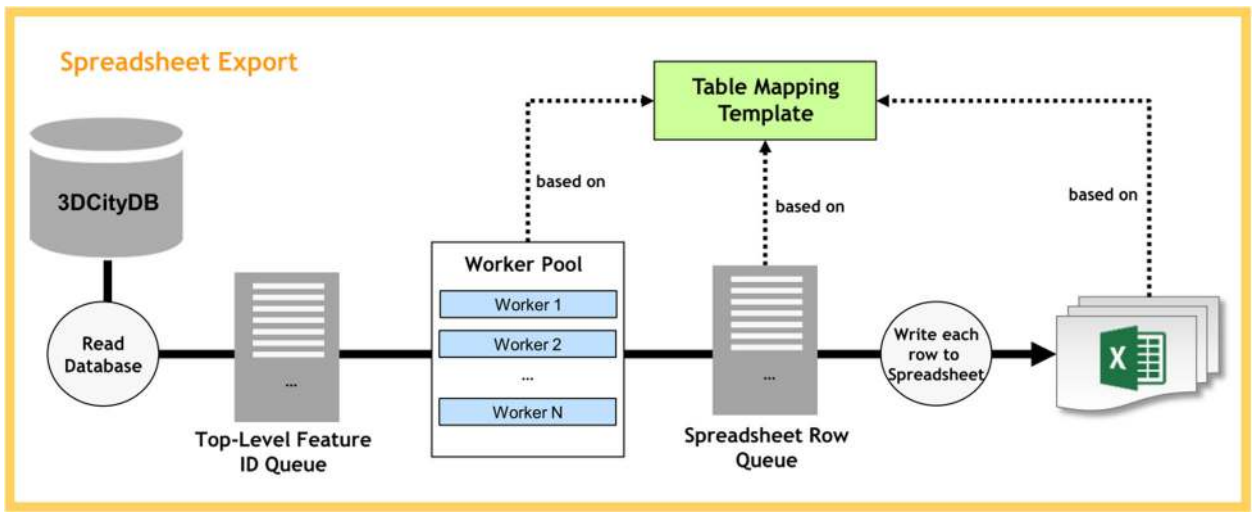

Fig. 14 Workflow of generating spreadsheet from 3DCityDB

based on a so-called table mapping template which can be freely defined by the users. This template is text-based file and comprises a set of key-value pairs (kvp) each of which can be seen as a column definition: The "key" of a kvp specifies an expression which can be directly translated to an SQL statement for fetching the data from a column of a specific 3DCityDB table, whereas the "value" specifies the column name in the output spreadsheet. With this template file, the value of each spreadsheet column can be dynamically queried from the database for every city object and written to the spreadsheet during export. In case that more than one value is returned, it is possible to select the first or the last one of the returned values or simply group them as a comma separated string value which can be passed into the corresponding spreadsheet cell.

3DCityDB web-map-client

Starting from version 3.3.0, the 3DCityDB software package comes with a new tool called *3DCityDB Web-Map-Client* or simply called *3D web client* acting as a web-based front-end for 3D visualization and interactive exploration of arbitrarily large semantic 3D city models. Basically, the 3D web client is an extension of the CesiumJS WebGL virtual globe. The latter is an Open Source software offering high-performance and cross-platform visualization and exploration of 3D geographic contents on the web without the need to install additional web browser plugins. The 3D web client implements various extensions to the CesiumJS virtual globe as shown in Fig. 15. The major new functionality is the support and handling of configurable data layers allowing to create visualization mashups consisting of digital terrain models, imagery data as well as of (possibly large tiled) 3D visualization models in the formats like KML, CZML, glTF, 3DTiles, point. Users can interactively add and remove, enable and disable their selected data layers on the 3D map. In addition, user interaction with the 3D models is also supported, e.g. highlighting of 3D objects on mouse-over and mouse-click, hiding and showing of the selected 3D objects as well as the exploration from different view perspectives using third-party mapping services like Microsoft Bing Maps with oblique view, Google Streetview, and a combined version using DualMaps. Moreover, the 3D web client implements a Cloud-based online spreadsheet API, i.e. the Google Fusion Table API, to query thematic information for clicked 3D objects stored in a tabular form. These functions allow to link the 3D visualization models generated from the KML/COLLADA/glTF exporter plugin with the tabular thematic data exported from the Spreadsheet Generator plugin within a 3D web client project. When a user clicks onto a 3D object, the linked Google Fusion Table is queried for the respective row and all its attributes are displayed then (cf. [18, 45]). In the past four years, this 3D web client has been successfully used in many research projects. It was awarded with the first price in the 'Best Students Contribution' of the 'Web3D city modelling competition' at the ACM SIGGRAPH Web3D Conference in 2015.

The graphical user interface of the 3D web client is shown in Fig. 16. Users are able to control the visibility of the selected data layers by deactivating their checkboxes or clicking on the *Remove selected layer* button to completely remove it from the 3D web client. In the example in Fig. 16, three data layers are loaded into the web client. Depending on the distance between the camera and the individual building objects, one of the three geometry representations are automatically chosen by the 3D web client for the display and dynamically

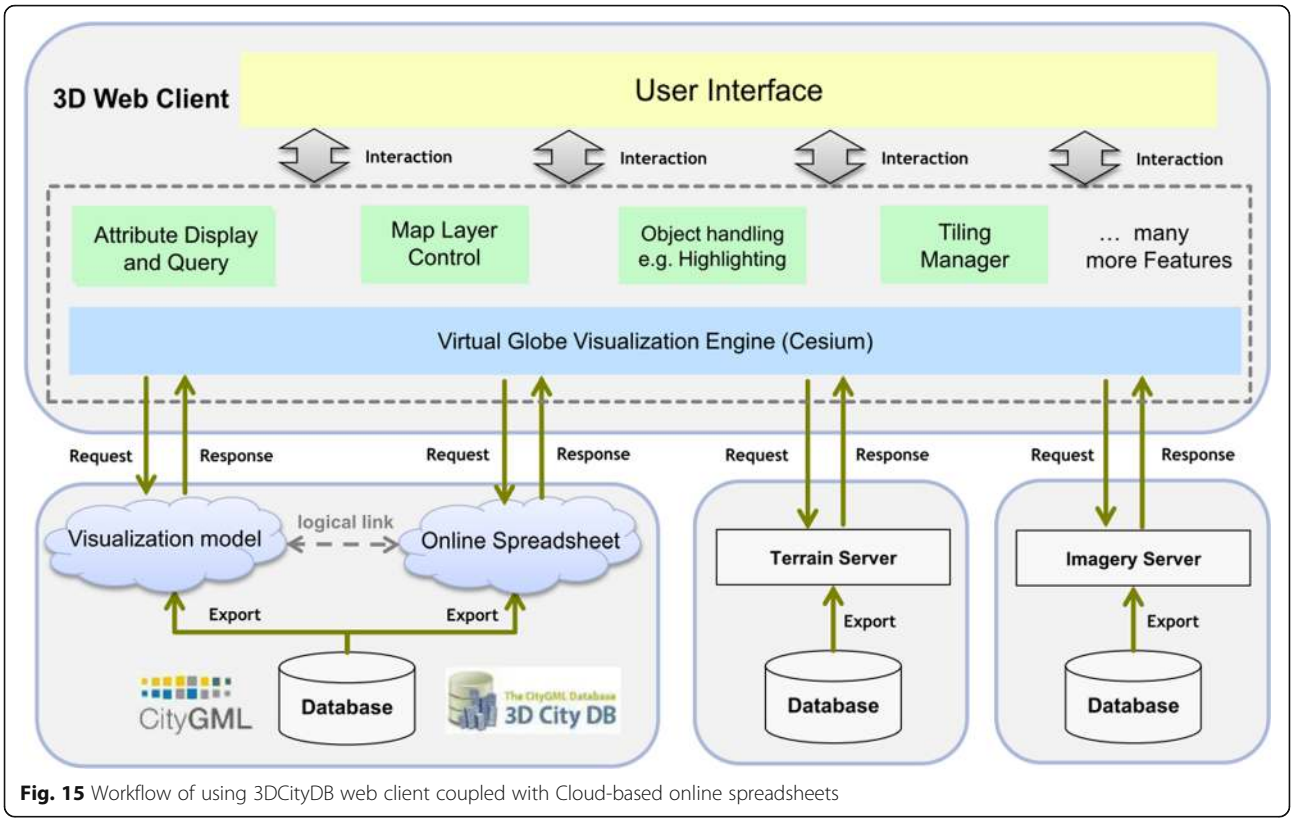

Fig. 15 Workflow of using 3DCityDB web client coupled with Cloud-based online spreadsheets

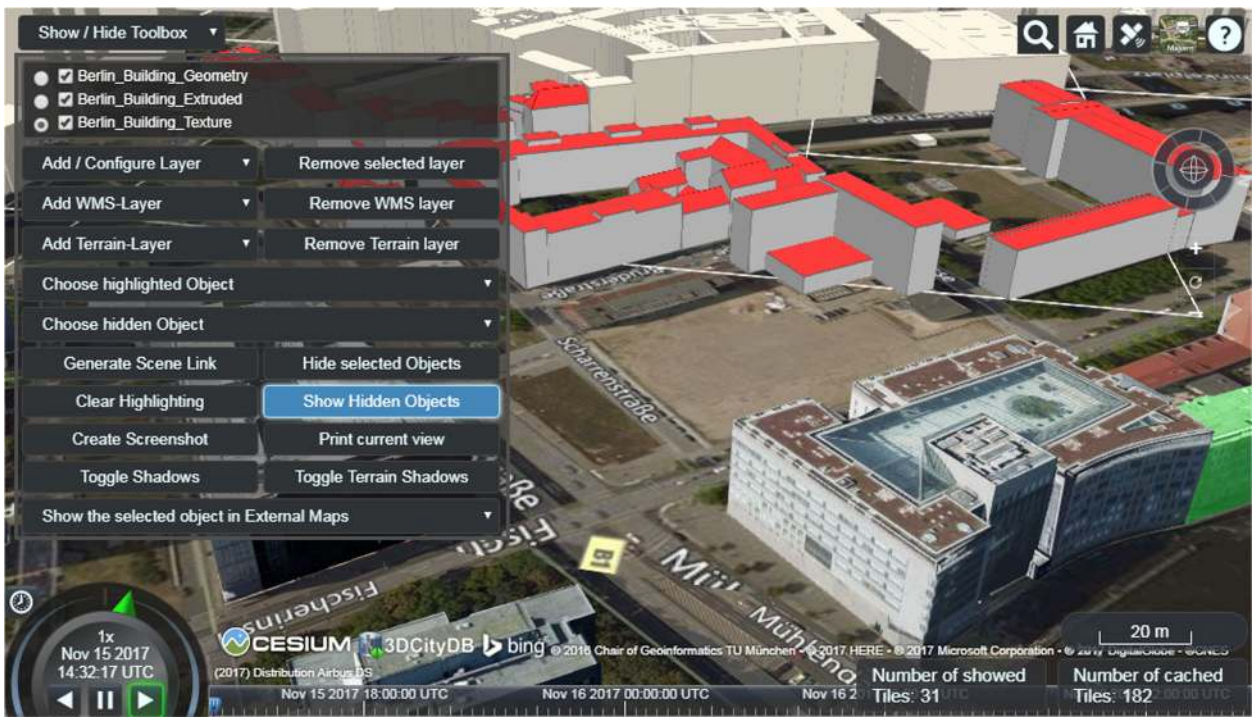

Fig. 16 Example of displaying multiple data layers on the 3D web client

switched during the runtime according to the Level of Detail mechanism: The buildings being far away from the camera are simply rendered as extruded or untextured geometries, while the textured buildings with higher details will be rendered when viewing the building objects from a short distance. In this way, the 3D visualization of large 3D city models with higher level of details (LOD ≥ 2) can be efficiently performed. The 3D web client automatically detects, if it is being executed on a mobile device like a smart phone or tablet computer. In this case, icons and text sizes are adjusted properly. When running in mobile mode, the client can also continuously align the 3D viewer with the user's position and heading.

Finally, it is possible to create a scene link saving the current configuration of the 3D web client by clicking on the *Generate Scene Link* button. This scene link encodes the information about the title of the 3D project, activation status of the shadow visualization, parameters of all layers, the current camera perspective etc. The created scene link can be stored as a browser bookmark or favourite and can also be sent e.g. by email to friends, colleagues, project partners etc. When they open the link, the same scene will open in their browsers. This encoding of the entire 3D client project within the URL is based on earlier work on public cooperative web maps as described in Kolbe et al. [23].

Examples for applications and the adoption of 3DCityDB

The 3DCityDB software suite has been and is being employed by many cities as well as by mapping agencies on regional, state, and country level worldwide (concrete examples are named in subsections 4.1 to 4.3). It is also being used by companies as an embedded component of their products or to work with CityGML datasets in various application domains.

Apart from the adoption in administration and industry, 3DCityDB has been applied for storing and analysing semantic 3D City and Landscape Models in many research projects. For example, Konde and Saran [25] report their experience from using 3DCityDB in their research on spatio-temporal semantic analysis of traffic noise. 3DCityDB has been used by Santana et al. [36] in their research on mobile visualization of urban energy modelling and simulation results as well as by Tymkow et al. [40] for flood modelling. Agugiaro [2] describes using the 3DCityDB tools for creating an integrated CityGML-based 3D model of Vienna. 3DCityDB is also being used for Smart City projects, for example, the 'Future Cities Pilot 1' of the Open Geospatial Consortium (cf. [9]) and within the so-called 'Smart District Data Infrastructure (SDDI)' as presented by Moshrefzadeh et al. [32]. Due to the Open Source nature of the 3DCityDB, researchers also report on modifying specific parts of the software. For example, Ghassoun and Löwner [15] document their adaption and usage of the 3DCityDB migration SQL script for the calculation of building volumes and Blut et al. [7] describe their development of a simplified database schema – which is inspired by the 3DCityDB schema – for using CityGML models on smartphones.

In the following subsections, the three main contributors to the 3DCityDB Open Source project, namely the Chair of Geoinformatics at Technical University of Munich and the companies virtualcitySYSTEMS and M.O.S.S., describe practical use cases and how 3DCityDB is being employed as a part of commercial software systems. This illustrates the usability of the 3DCityDB software solution for projects in research, administration, and industry.

3D city model of new York City

The first example is the "Semantic 3D Model of New York City", which is a student project that was carried out in the context of the three master theses of Barbara Burger, Berit Cantzler, and Christof Beil within the master's program Geodesy and Geoinformation at TUM (cf. [5, 22]). The key objective of the project has been the creation of a homogenized and integrated semantic 3D city model of New York City (NYC) from existing public 2D, 2.5D, and 3D datasets provided in the NYC Open Data Portal. Different spatial and semantic transformations together with some photogrammetric analyses were investigated and performed using the ETL tool Feature Manipulation Engine (FME) from Safe Software. The resulting 3D city model integrates data from around 30 separate datasets from the NYC Open Data Store into a single CityGML dataset. It comprises a variety of 3D feature types including all NYC buildings, land parcels, roads, parks, the digital terrain model, and water bodies. The CityGML dataset has also been imported into a 3DCityDB instance for data management and exported to tiled KML/glTF models for Web-based 3D visualization. In order to interactively explore the resulting 3D city model in the 3DCityDB web client (see Fig. 17 and section 3.6), the generated 3D visualization models have been published on the 3DCityDB web server. The entire CityGML dataset as well as many online demos and videos are provided on the project homepage.[14]

The 3D city model of NYC and the 3DCityDB have been used already for different applications such as solar irradiation analysis and traffic simulations. In order to estimate the potentials of solar energy production for roofs and facades of all buildings in NYC a dedicated software tool called 'Solar Potential Analysis Tool' developed by TUM (cf. [43]) has been employed. This Java-based simulation tool works directly on the data tables of the 3DCityDB. It takes the 3D building models together with the calculated

Fig. 17 3D Visualization of the created 3D City Model of NYC within a Web browser

sun positions for each hour over an entire year as well as an approximated sky dome to estimate the direct, diffuse, and global solar irradiation using a ray casting approach, taking into account the shadowing effects of the surrounding 3D topographic objects and the digital elevation model. The estimated solar power values are then aggregated on a monthly basis at first for individual surfaces, then for all wall and roof surfaces of each building, and finally for each building. All aggregated data are attached to the 3D building models as CityGML generic attributes on Building, WallSurface, and RoofSurface objects within the 3DCityDB. Also, textures for each surface and month are generated and attached to the building models. In addition to the solar power, the Sky View Factor (SVF) which is a value ranging between 0 and 1 representing the visible fraction of the sky dome has also been calculated for each surface of the buildings.

A 3D visualization model (a tiled KML/glTF model) has been generated from the textured building models using the 3DCityDB KML/COLLADA/glTF Exporter. The thematic information (solar irradiation values) was exported on building level using the Spreadsheet Generator Plugin and the resulting file was uploaded to Google Fusion Tables. The 3DCityDB Webmap client can now be used to interactively explore the 3D city model with the simulation results as shown in Fig. 18. The same process has been applied to the 3D city model of Rennes, France, within the OGC Future Cities Pilot project (cf. [9, 10]).

virtualcitySUITE - a 3DCityDB-based software from VCS

virtualcitySYSTEMS[15] (VCS) is a key contributor to the 3DCityDB project and expert in the field of CityGML-based 2D/3D Web GIS solutions. With the virtualcitySUITE, VCS offers a modular software solution for the management, distribution, web-based visualization as well as analysis and simulation of massive 3D geodata, whose core component is the 3DCityDB. Being based on open standards, interfaces and APIs, the virtualcitySUITE ensures a reliable and scalable basis for innovative Smart City applications.

VCS has augmented the 3DCityDB with many advanced database functions and plugins for the Importer/Exporter that enable data management and maintenance processes such as data quality assessment, the integration of city objects with different LoDs and from different data sources into a clean and redundancy-free database representation, or the deletion of city objects using user-defined filter criteria. The Open Source WFS implementation has been extended to retrieve arbitrary subsets of the 3D city model based on thematic and spatial queries and to modify city objects using insert, update and delete operations. This allows for easily integrating 3DCityDB content into vendor-neutral applications and data maintenance workflows based on an open and standardized OGC web interface to the 3DCityDB.

Users of 3D city models often require the city objects to be provided in industry CAD or GIS formats other than CityGML to be able to seamlessly use the data in their software products and workflows. To tackle this requirement and to bring the 3D geodata into use, VCS has developed workspaces for the Feature Manipulation Engine (FME) from Safe Software to export 3DCityDB content into various target formats such as 3D Shape, Sketchup, DWG, DXF or 3D PDF. Together with

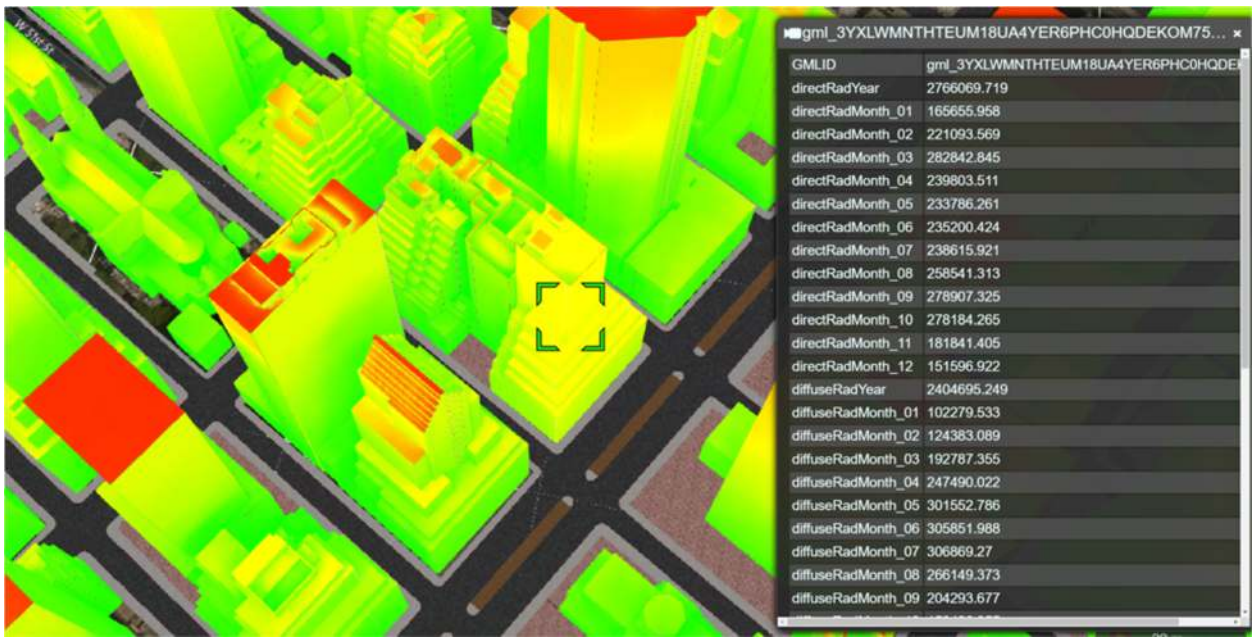

Fig. 18 3D visualization and exploration of the results of a solar irradiation estimation for buildings in central Manhattan

templates for processing and exchanging CityGML data, the workspaces have been published as *3D Solution Templates* on FME Hub.[16]

The virtualcitySUITE provides a web-based authoring tool for the creation of 3D web maps optimized for the high-performance and scalable visualization of 3D city models and geodata in modern web browsers and mobile devices without the need for additional plugins. Details and performance evaluations are given in [37]. User can choose any content from the 3DCityDB and have it automatically converted into a streaming dataset for the web. The Open Source web globe CesiumJS together with the candidate OGC community standard 3D Tiles is used as client-side rendering technology. The web maps are not limited to CityGML-based 3D models but also support massive 3D point clouds, 3D meshes, oblique imagery and legacy 2D map layers being seamlessly integrated into one application. They therefore serve as rich basis for building smart city applications on top of the 3D city model.

The virtualcitySUITE is in production use by cities and municipalities worldwide. Amongst others, the cities of Berlin, Hamburg, Helsinki, Rotterdam, Vienna, Singapore, Frankfurt, Dresden, Hannover, Potsdam, Salzburg, Falun, and Zurich as well as the German federal surveying agencies of Bavaria and Mecklenburg-Vorpommern rely on the 3DCityDB and the virtualcitySUITE for managing their 3D city model. The screenshot below shows the Berlin 3D city model consisting of more than 550,000 fully textured CityGML building models managed based on the virtualcitySUITE. It is used, amongst others, for the Berlin Economic Atlas,[17] a publicly accessible web map application that advertises Berlin as business location by combining 3D city assets with business and POI information. The Berlin 3D city model is available as open data and can be freely downloaded in different data formats from a web-based 3D geoportal[18] that has been realized with VCS technology on top of the 3DCityDB (Fig. 19).

Figure 20 shows further examples of CityGML-based 3D city models that are managed in the 3DCityDB and used in application scenarios realized with the virtualcitySUITE technology: Collaborative, web-based urban planning in the virtual urban space using CityGML models, BIM designs and 3D drawings (top left), linking results of a solar irradiation analysis on roofs and building façades with the city objects (top right), WFS-based search for buildings affected by a flooding scenario (lower left), and wind field and turbulence simulation of newly planned buildings in the built environment (lower right).

novaFACTORY – A 3DCityDB-based Software from M.O.S.S.

M.O.S.S. Computer Grafik Systeme GmbH[19] is another key contributor to the 3DCityDB project and offers a range of advanced geospatial data management and processing solutions. It owns an extensive commercial product suite called novaFACTORY which utilizes the 3DCityDB database schema as its core component for handling and processing CityGML datasets within a 3D geospatial data infrastructure (3D GDI). Based on the

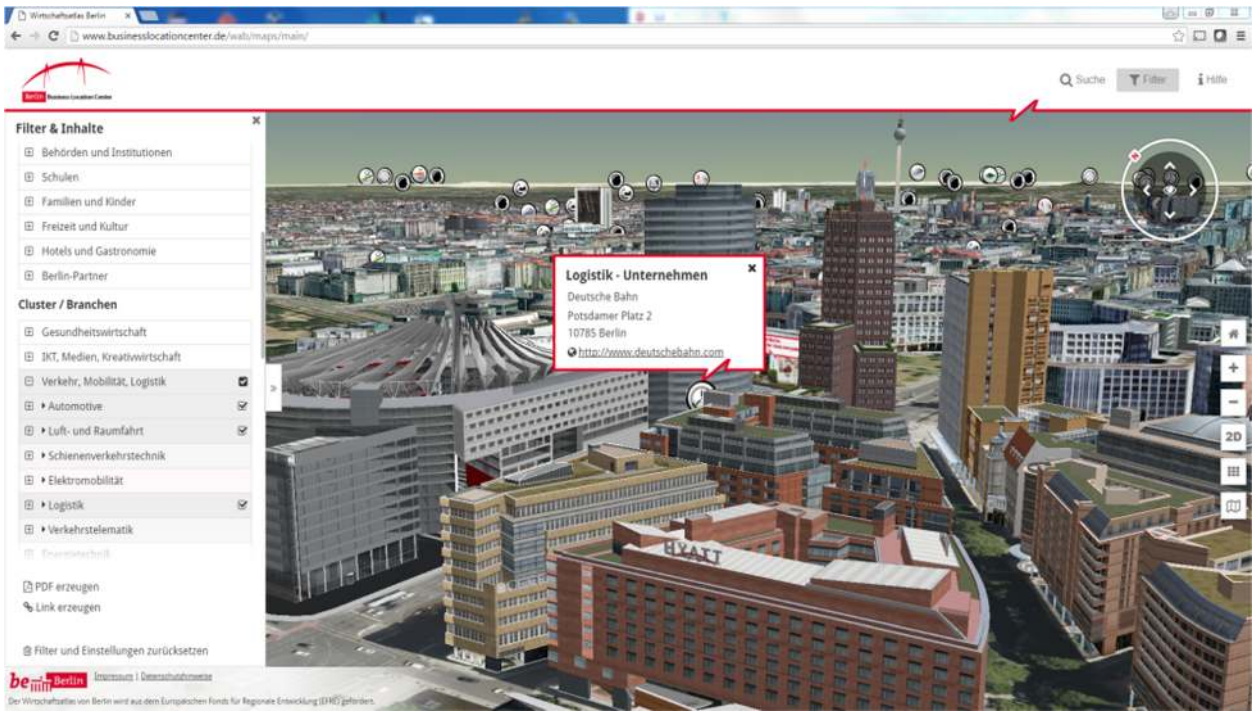

Fig. 19 The Berlin Economic Atlas realized as 3D web map application based on CityGML building models stored and managed in the 3DCityDB

3DCityDB, the novaFACTORY not only supports the efficient storage of 3D city models but also provide many additional data manipulation functionalities. For example, the CityGML data stored in the 3DCityDB can be automatically transferred into e.g. ESRI Geodatabase and allows to be easily interacted and explored using ArcGIS software like ArcMap. Besides, the CityGML data can also be disseminated in a variety of data formats, e.g. CityGML, KML/COLLADA, VRML, SKP, 3D Shape, 3D PDF and DXF etc. via a web-based service or graphical user interface. As an advanced module of the novaFACTORY suite, the 'novaFACTORY 3D Pro Module' additionally supports the automatic enrichment of the 3D city models based on an automatic process of recognizing the building roofs from photogrammetric data. Once the photogrammetric datasets have been successfully approved, the derived 3D city models will be automatically passed to the 3DCityDB and spread over the corresponding tables. In addition, the existing 3D city models stored in the database can also be updated in a user-defined regular time interval according to the actual changes of the real-world objects based on their unique model identifiers. The update process can be carried out using the software called "CityDoctor" (cf. [42]) for controlling the data quality regarding the semantic, topologic or geometric data errors which shall be fixed before importing the data into the 3DCityDB database. Moreover, the novaFACTORY also supports a number of data analysis and processing functions invoked during the export process. For instance, one of these functions allows to automatically detect the interior walls within a building or between two adjacent buildings. These walls could be marked as CityGML ClosureSurface structures and then used for distinguishing the outer and inner walls to facilitate performing heat demand analysis of buildings. In the near future, additional object types like towers, power poles, wind turbine generators or bridges will also be supported by the novaFACTORY software (Fig. 21).

In order to apply the developed software solution to a wide range of areas in Gemany, a user network called "M.O.S.S-Anwender 3D" has been initiated by M.O.S.S in 2012. The members of this network are from most German federal states who have submitted their 3D data as CityGML files and intended to create a national 3D landscape model for Germany. All these data have been successfully validated and compiled into an integrative dataset at the "Zentrale Stelle für Hauskoordinaten und Hausumringe (ZSHH)". The dataset is managed within a central 3DCityDB instance. Today, this nationwide database contains more than 40 million LoD1 building objects and has become one of the worldwide largest productive 3D city models based on the 3DCityDB and CityGML. With the help of the novaFACTORY solution, the efficient maintenance, quality assurance, and visualization of this large 3D dataset has been realized in

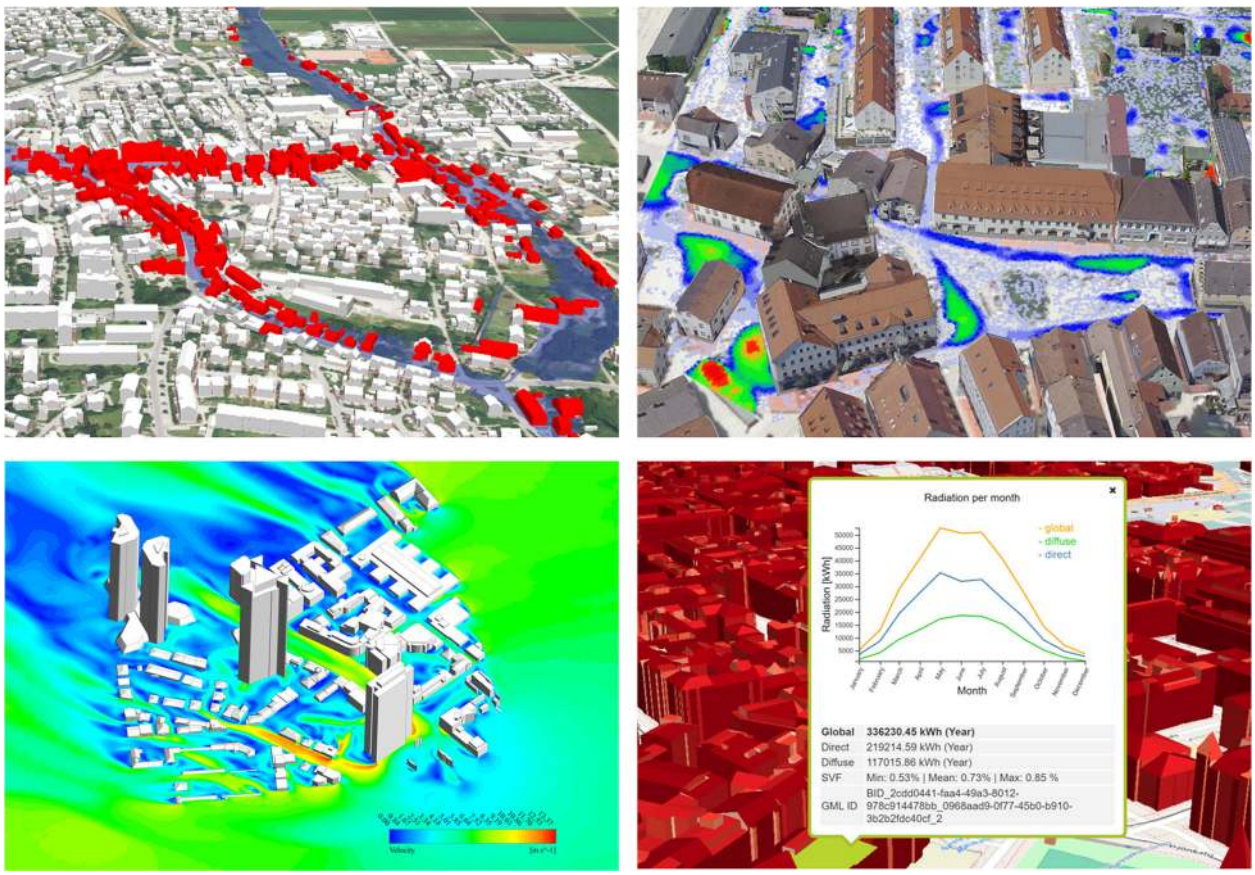

Fig. 20 Application examples using the virtualcitySUITE and 3D city models managed in the 3DCityDB

a highly automated way. Furthermore, M.O.S.S. also provides various commercial services for the fields like real estate industry and renewable energy topics like building heat demand analysis and solar potential assessment.

Conclusions and outlook

This paper presented the 3DCityDB software suite, which is an Open Source 3D geodatabase solution to manage, analyse, and visualize large 3D city models according to the CityGML standard. 3DCityDB is implemented as a relational database schema plus stored procedures for different SRDBMS. It comes with a set of Java-based software tools to provide web-based data access (WFS), import and export of CityGML datasets of arbitrary sizes, and to generate 3D visualization models including thematic data. Furthermore, an HTML5/WebGL-based 3D web client for interactive exploration of the generated 3D visualization models is included. By the combined usage of these software components complete work chains can be established ranging from the reading, processing, and writing of the 3D city model contents into the database, via the conversion to different model representations, to interactive Web-based data visualization and exploration in a 3D map viewer. The emphases of the paper lie on 1) the explanation of the challenges to handle CityGML datasets within a database and the discussion of related systems, 2) the optimized mapping of the complex, object oriented data model of CityGML onto a compact relational schema, 3) the presentation of the provided tools, their functionalities as well as technical details on their implementation, and 4) the presentation and discussion of application examples from research, education, public administration, and industry.

Currently, our focus mainly lies on extending the 3DCityDB database and software tools to support the handling of CityGML Application Domain Extensions (ADEs). ADEs are community defined extensions to the CityGML data model for specific application domains. There exists already a number of ADEs, for example, for building energy assessment (Energy ADE, cf. [34]), environmental noise propagation (Noise ADE, included in the CityGML 2.0 specification), and the representation of technical infrastructures (Utility Network ADE, cf. [4, 27]). Furthermore, the national 3D model of The Netherlands is expressed as a CityGML ADE called

Fig. 21 Example of a 3D building heat demand map for the city of Ludwigsburg created in the SIMSTADT project and using the novaFACTORY software (cf. [26])

IMGeo3D (cf. van den Brink et al. [41]). Recently, a first extension of the 3DCityDB database schema for the Energy ADE has been presented by Agugiaro & Holcik [3]. It was derived manually by following the rules presented in this paper.

The problem concerning the generic support of ADEs is that the current database schema has been created manually from the CityGML data model resulting in a static schema. While this schema offers a number of advantages, it cannot store data elements from CityGML ADEs, because they were not included in the original CityGML data model. While there exists a generic solution, which generally employs a fixed set of tables with a rigid relational structure allowing to map arbitrary XML-based graph like structures onto a relational database model (cf. [12, 29]), this approach poses two significant drawbacks. First, the mapped relational structure usually requires a large number of recursive joins to represent the aggregation and inheritance hierarchies of the object-oriented data model resulting in a low query performance. Second, it lacks the possibility of explicitly describing the mapping relationship between classes and tables, since all classes and their attributes are mapped onto a mixed table structure. This makes it difficult for external applications to retrieve the data contents from the database. Therefore, instead of a generic database structure, the 3DCityDB database models must be dynamically extended in such a way that each CityGML ADE is handled like a 'plugin' for the 3DCityDB extending the fixed 3DCityDB core database schema by respective new tables. Since CityGML ADEs are basically GML application schemas represented in the XML Schema language, they can contain arbitrary feature, attribute, and relation definitions. This renders the automatic derivation of '3DCityDB-style' database schemas a challenging task. Nevertheless, this work made good progress and first results have been presented in [44].

Future work refers to the adaption of 3DCityDB to the next major release 3.0.0 of CityGML, which is currently being compiled and will include many improvements of the current data model, new thematic modules, and an updated levels-of-detail schema (cf. [30]). One of the proposed new concepts is called *Dynamizer*. It extends CityGML by a new representation for dynamic property values and explicit linking with real-time sensor data (see [8]). For CityGML 2.0, a Dynamizer ADE has been created which was employed in the OGC Future Cities Pilot Phase 1 [9]. We are currently working on a corresponding extension of the 3DCityDB, where the time series data will be queried by an OGC Sensor Observation Service running directly on the 3DCityDB.

Availability and requirements

Project name: 3D City Database (3DCityDB).

Project home page: https://www.3dcitydb.org

Operating system(s): Platform independent.

Programming languages: SQL, PL/SQL, Java, JavaScript.

Other requirements: Java 8 or higher, Tomcat 4.0 or higher, ORACLE DBMS >= 10 g R2 with Spatial or

Locator option, PostgreSQL DBMS > = 9.1 with PostGIS extension > = 2.0, Apache Tomcat 7 or higher, WebGL enabled hardware and web browsers.

License: Apache 2.0.

Endnotes

[1]https://www.deegree.org/
[2]http://www.gdal.org/drv_gmlas.html
[3]http://www.cpa-software.de
[4]https://snowflakesoftware.com/geospatial-products/
[5]http://basex.org/
[6]https://www.mongodb.com/ https://www.mongodb.com/
[7]https://georocket.io/
[8]https://neo4j.com/product/
[9]https://github.com/tum-gis/citygml-change-detection
[10]https://github.com/citygml4j
[11]https://github.com/javaee/jaxb-v2
[12]http://www.citydoctor.eu
[13]https://github.com/tudelft3d/val3dity
[14]http://www.gis.bgu.tum.de/en/projects/new-york-city-3d/
[15]http://www.virtualcitysystems.de/
[16]https://hub.safe.com/
[17]https://www.businesslocationcenter.de/wab/maps/main/
[18]http://www.businesslocationcenter.de/berlin3d-downloadportal/
[19]http://www.moss.de/

Abbreviations

3DCityDB: 3D City Database; ADE: Application domain extension; B-Rep: Boundary representation; CityGML: City Geography markup language; COLLADA: Collaborative design activity; CPU: Central processing unit; CZML: Cesium markup language; ESRI: Environmental systems research institute; FME: Feature manipulation engine; GIS: Geographic information system; glTF: GL Transmission format; GML: Geography markup language; HTML: Hypertext markup language; HTTP: Hypertext transfer protocol; ISO: International organisation for standardisation; KML: Keyhole markup language; LOD: Level of detail; OGC: Open Geospatial consortium; SQL: Structured query language; SRDBMS: Spatially-enabled relational database management system; SRDBMS: Spatially-enhanced relational database management system; TUM: Technische Universität München; WebGL: Web graphics library; WFS: OGC Web feature service; XML: Extensible markup language; XSD: XML schema definition

Acknowledgements

We thank all contributors to the 3DCityDB development starting from its beginning in 2003. The names of all contributors and their affiliations are provided in the 3DCityDB documentation. We also thank all sponsors and funding authorities for their financial support. We thank the company ORACLE for their technical support. We also thank the reviewers for their constructive comments.

Funding

The development of the 3DCityDB software has been financially supported by a large number of different research and commercial projects. In the beginning it was funded by the cities of Berlin, Bonn, and Potsdam in Germany as well as by the European Regional Development Fund (ERDF). Over the last eight years most developments were funded by the *Knowledge and Innovation Community on Climate Change and Mitigation* (Climate-KIC) of the *European Institute of Innovation and Technology* (EIT), the *German Research Foundation* (DFG), as well as by the Technical University of Munich and the two companies involved in the 3DCityDB development *virtualcitySYSTEMS* and *MOSS*.

Availability of data and materials

All resources mentioned in this article (database schemas, software tools, and documentation etc.) are freely available, mostly on GitHub (https://github.com/3dcitydb) or on the 3DCityDB homepage (https://github.com/3dcitydb). The hands-on tutorial explaining step by step the download, installation, and manipulation of the software is freely available via: https://www.gis.bgu.tum.de/en/projects/3dcitydb/#c1426

Authors' contributions

Some parts and illustrations of this paper have been adopted from the 3DCityDB documentation (cf. [24]) which was created by the authors of this paper. ZY and THK designed the structure of the paper. ZY mostly wrote the "City Modelling and Mapping onto 3D Geodatabase", "Implementation of the 3DCityDB and its Tools" and "Conclusion" sections. All authors wrote the "Application Examples" section. THK, ZY, CN, AD, and FK revised the paper according to the reviewers' comments. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Author details

[1]Chair of Geoinformatics, Technische Universität München, Arcisstraße 21, 80333 Munich, Germany. [2]virtualcitySYSTEMS GmbH, Tauentzienstraße 7B, 10789 Berlin, Germany. [3]Beuth Hochschule, Luxemburger Straße 10, 13353 Berlin, Germany. [4]M.O.S.S. Computer Grafik Systeme GmbH, Hohenbrunner Weg 13, 82024 Taufkirchen, Germany.

Received: 30 November 2017 Accepted: 11 March 2018
Published online: 14 May 2018

References

1. Agoub A, Kunde F, Kada M. Potential of graph databases in representing and enriching standardized Geodata. In: Kersten TP, editor. Tagungsband der 36. Wissenschaftlich-Technischen Jahrestagung der DGPF in Bern. Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V. (DGPF), vol 25; 2016. p. 208–16.
2. Agugiaro G (2016) First steps towards an integrated CityGML-based 3D model of Vienna. In: 2016 XXIII ISPRS congress, 12–19 July 2016, Prague, Czech Republic. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol III-4, pp 139–146.
3. Agugiaro G, Holcik P (2017) 3D City Database extension for the CityGML Energy ADE 0.8 PostgreSQL Version. Technical Documentation. https://github.com/gioagu/3dcitydb_ade/tree/master/02_energy_ade. Last accessed 29 Jan 2018.
4. Becker T, Nagel C, Kolbe TH. Semantic 3D modeling of multi-utility networks in cities for analysis and 3D visualization. In: Pouliot J, Daniel S, Hubert F, Zamyadi A, editors. Progress and new trends in 3D Geoinformation sciences. Lecture notes in Geoinformation and cartography. Heidelberg: Springer; 2013. p. 41–62.
5. Beil C, Kolbe TH (2017) CityGML and the streets of New York - a proposal for detailed street space modelling. In: Proceedings of the 12th 3D GeoInfo conference 2017, Melbourne, Australia, 26–27 October 2017. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. IV-4/W5, pp 9–16.
6. Biljecki F, Stoter J, Ledoux H, Zlatanova S, Çöltekin A. Applications of 3D city models: state of the art review. ISPRS International Journal of Geo-Information. 2015;4(4):2842–89.
7. Blut C, Blut T, Blankenbach J. CityGML goes mobile: application of large 3D CityGML models on smartphones. International Journal of Digital Earth. 2017:1–18. https://doi.org/10.1080/17538947.2017.1404150.
8. Chaturvedi K, Kolbe TH (2016) Integrating dynamic data and sensors with semantic 3D city models in the context of smart cities. In: Proceedings of the 11th international 3D Geoinfo conference, Athens, Greece, 20–21

October 2016. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol IV-2/W1, pp 31–38.
9. Chaturvedi K, Kolbe TH (eds.) (2017) Future City Pilot 1 Engineering Report. Public Engineering Report, OGC Doc. No. 16–098, Open Geospatial Consortium. http://docs.opengeospatial.org/per/16-098.html. Last accessed 28 Jan 2018.
10. Chaturvedi K, Willenborg B, Sindram M, Kolbe TH (2017) Solar potential analysis and integration of the time-dependent simulation results for semantic 3D city models using Dynamizers. In: Proceedings of the 12th 3D GeoInfo conference 2017, Melbourne, Australia, 26–27 October 2017. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol IV-4/W5, pp 25–32.
11. de Souza Baptista C, Pires CES, Leite DFB, de Oliveiraa MG. NoSQL geographic databases: an overview. In: Pourabbas E, editor. Geographical information systems: trends and technologies, 73, CRC press; 2014. p. 73–103.
12. Florescu D, Kossmann D. Storing and querying XML data using an RDMBS. IEEE Data Engineering Bulletin. 1999;22(3):27–34.
13. Foley J, van Dam A, Feiner SK, Hughes JK. Computer graphics: principles and practice. 2nd ed. Boston: Addison-Wesley; 1995.
14. Gamma E, Helm R, Johnson R, Vlissides J. Design patterns: elements of reusable object-oriented software. Boston: Addison-Wesley; 1995.
15. Ghassoun Y, Löwner MO. Comparison of 2D & 3D parameter-based models in urban fine dust distribution modelling. In: Abdul-Rahman A, editor. Advances in 3D Geoinformation. 10th international conference 3DGeoinfo 2015, Kuala Lumpur, Malaysia, 28–30 October 2015. Lecture notes in Geoinformation and cartography. Springer, Cham; 2017. p. 231–46.
16. Golobisky MF, Vecchietti A. Fundamentals for the automation of object-relational database design. IJCSI International Journal of Computer Science. 2011;8(3):9–22.
17. Gröger G, Kolbe TH, Nagel C, Häfele KH (2012) OGC City Geography Markup Language (CityGML) Encoding Standard, Version 2.0, OGC Doc No. 12–019. Open Geospatial Consortium. https://portal.opengeospatial.org/files/?artifact_id=47842. Accessed 13 Nov 2017.
18. Herreruela J, Nagel C, Kolbe TH. Value-added services for 3D city models using cloud computing. In: Mobilität und Umwelt, Konferenzband zur Tagung Geoinformatik 2012, Braunschweig, 28–30 March 2012; 2012.
19. Keller W (1997) Mapping objects to tables: a pattern language. In: Buschmann F, Riehle D (eds) Proceedings of European conference on pattern languages of programming and computing, Kloster Irsee, Germany, 10 July 1997. Siemens Technical Report, 120/SW1/FB, pp 59–84.
20. Koch S, Löwner MO. Representation of CityGML instance models in BaseX. In: Abdul-Rahman A, editor. Advances in 3D Geoinformation. 10th international conference 3DGeoinfo 2015, Kuala Lumpur, Malaysia, 28–30 October 2015. Lecture notes in Geoinformation and cartography, springer, Cham; 2017. p. 63–78.
21. Kolbe TH (2009) Representing and exchanging 3D city models with CityGML in 3D Geo-Information Sciences. In: Lee J, Zlatanova S (eds) 3D Geo-Information Sciences. Lecture notes in Geoinformation and cartography, springer, Heidelberg, pp 15–31.
22. Kolbe TH, Burger B, Cantzler B. CityGML goes to Broadway. In: Fritsch D, editor. Photogrammetric week '15. 55th photogrammetric week in Stuttgart, September 2015. Wichmann, berlin; 2015. p. 343–56.
23. Kolbe TH, Steinrücken J, Plümer L. Cooperative public web maps. In: Proceedings of the international cartographic congress (ICC), Durban, South Africa, 10–16 august 2003; 2003.
24. Kolbe TH, Yao Z, Nagel C, Redweik R, Willkomm P, Hudra G, Müftüoglu A, Kunde F (2016) 3D Geodatabase for CityGML Documentation Version 3.3.0. https://www.3dcitydb.org/3dcitydb/fileadmin/downloaddata/3DCityDB_Documentation_v3.3.pdf. Accessed 13 Nov 2017.
25. Konde A, Saran S. Web enabled spatio temporal semantic analysis of traffic noise using CityGML. J Geom. 2017;11(2):248–59.
26. Koukofikis A, Coors V. An integration of urban spatial data with energy simulation to produce X3D city models: the case of Landkreis Ludwigsburg. In: Proceedings of the 20th international conference on 3D web technology, Heraklion, Crete, Greece, 18–21 June 2015; 2015.
27. Kutzner T, Kolbe TH. Extending semantic 3D city models by supply and disposal networks for Analysing the urban supply situation. In: Kersten TP, editor. Tagungsband der 36. Wissenschaftlich-Technischen Jahrestagung der DGPF in Bern. Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V. (DGPF), vol 25, DGPF; 2016. p. 382–94.
28. Ledoux H. On the validation of solids represented with the international standards for geographic information. Computer-Aided Civil and Infrastructure Engineering. 2013;28(9):693–706.
29. Li Y, Li J, Zhou S. GML storage: a spatial database approach. In: Wang K, Tanaka S, Zhou TW, Ling J, Guan D, Yang F, Grandi E, Mangina IY, Song MHC, editors. Conceptual modeling for advanced application domains. ER 2004. Lecture notes in computer science. Berlin: Springer; 2004. p. 55–66.
30. Löwner MO, Gröger G, Benner J, Biljecki F, Nagel C. Proposal for a new LOD and multi-representation concept for CityGML. In: Proceedings of the 11th international 3D Geoinfo conference, Athens, Greece, 20–21 October 2016. ISPRS annals of the photogrammetry, remote sensing and spatial information sciences, vol IV-2/W1; 2016. p. 3–12.
31. Mao B, Harrie L, Cao J, Wu Z, Shen J. NoSQL based 3D city model management system. In: Jiang J, Zhang H, editors. ISPRS technical commission IV symposium, Suzhou, China, 14–16 may 2014. The international archives of photogrammetry, remote sensing and spatial information sciences, vol XL-4; 2014. p. 169–73.
32. Moshrefzadeh M, Chaturvedi K, Hijazi I, Donaubauer A, Kolbe TH. Integrating and managing the information for smart sustainable districts - the Smart District data infrastructure (SDDI). In: Kolbe TH, Bill R, Donaubauer A, editors. Geoinformationssysteme 2017 – Beiträge zur 4. Münchner GI-Runde, Wichmann Verlag, Heidelberg; 2017.
33. Nguyen S, Yao Z, Kolbe TH (2017) Spatio-semantic comparison of large 3D city models in CityGML using a graph database. In: Proceedings of the 12th 3D GeoInfo conference 2017, Melbourne, Australia, 26–27 October 2017. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol IV-4/W5, pp 99–106.
34. Nouvel R, Kaden R, Bahu JM, Kaempf J, Cipriano P, Lauster M, Benner J, Munoz E, Tournaire O, Casper E. Genesis of the CityGML energy ADE. In: Proceedings of the international conference CISBAT 2015 on future buildings and districts sustainability from Nano to urban scale, Lausanne, Switzerland, EPFL-Conf-no. 213436, EPFL; 2015. p. 931–6.
35. Ordonez C, Song I-Y, Garcia-Alvarado C. Relational versus non-relational database systems for data warehousing. In: Proceedings of the ACM 13th international workshop on data warehousing and OLAP, Toronto, Canada, 26–30 October 2010; 2010.
36. Santana JM, Wendel J, Trujillo A, Suárez JP, Simons A, Koch A. Multimodal location based services—semantic 3D city data as virtual and augmented reality. In: Gartner G, Huang H, editors. Progress in location-based services 2016, Vienna, Austria, 14–16 November 2016. Lecture notes in Geoinformation and cartography, springer, Cham; 2017. p. 329–53.
37. Schilling A, Bolling J, Nagel C (2016) Using glTF for streaming CityGML 3D city models. In: Proceedings of the 21st international conference on Web3D technology (ACM Web3D), Anaheim, CA, USA, 22–24 July 2016. ACM Press.
38. Stadler A, Kolbe TH. Spatio-semantic coherence in the integration of 3D city models. In: Stein A, editor. Proceedings of the 5th international ISPRS symposium on spatial data quality ISSDQ, Enschede, Netherlands, 24–25 June 2007, ISPRS archives of the photogrammetry, remote sensing and spatial information sciences, vol XXXVI-2/C43; 2007.
39. Stadler A, Nagel C, König G, Kolbe TH. Making interoperability persistent: a 3D geo database based on CityGML. In: Lee J, Zlatanova S, editors. 3D geo-information sciences. Lecture notes in Geoinformation and cartography, springer, Heidelberg; 2009. p. 175–92.
40. Tymkow P, Karpina M, Borkowski A. 3D GIS for flood modelling in river valleys. In: 2016 XXIII. ISPRS congress, Prague, Czech Republic, 12–19 July 2016. The international archives of the photogrammetry, remote sensing and spatial information sciences, vol XLI-B8; 2016. p. 175–8.
41. van den Brink L, Stoter J, Zlatanova S. Establishing a national standard for 3D topographic data compliant to CityGML. Int J Geogr Inf Sci. 2013;27(1):92–113.
42. Wagner D, Wewetzer M, Bogdahn J, Alam N, Pries M, Coors V. Geometric-semantical consistency validation of CityGML models. In: Pouliot J, Daniel S, Hubert F, Zamyadi A, editors. Progress and new trends in 3D Geoinformation sciences. Lecture notes in Geoinformation and cartography. Heidelberg: Springer; 2013. p. 171–92.
43. Willenborg B, Sindram M, Kolbe TH. Applications of 3D city models for a better understanding of the built environment. In: Behnisch M, Meinel G, editors. Trends in spatial analysis and modelling. Geotechnologies and the environment, vol 19. Springer, Cham; 2018.
44. Yao Z, Kolbe TH. Dynamically extending spatial databases to support CityGML application domain extensions using graph transformations. In: Kersten TP, editor.

Tagungsband der 37. Wissenschaftlich-Technischen Jahrestagung der DGPF in Würzburg, Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V. (DGPF), vol 26; 2017. p. 316–31.

45. Yao Z, Sindram M, Kaden R, Kolbe TH. Cloud-basierter 3D-Webclient zur kollaborativen Planung energetischer Maßnahmen am Beispiel von Berlin und London. In: Kolbe TH, Bill R, Donaubauer A, editors. Geoinformationssysteme 2014 – Beiträge zur 1. Münchner GI-Runde, Wichmann, Berlin; 2014. p. 40–52.