

Integrating 3D city data through knowledge graphs

Linfang Ding, Guohui Xiao, Albulen Pano, Mattia Fumagalli, Dongsheng Chen, Yu Feng, Diego Calvanese, Hongchao Fan & Liqiu Meng

To cite this article: Linfang Ding, Guohui Xiao, Albulen Pano, Mattia Fumagalli, Dongsheng Chen, Yu Feng, Diego Calvanese, Hongchao Fan & Liqiu Meng (2025) Integrating 3D city data through knowledge graphs, *Geo-spatial Information Science*, 28:2, 780-799, DOI: [10.1080/10095020.2024.2337360](https://doi.org/10.1080/10095020.2024.2337360)

To link to this article: <https://doi.org/10.1080/10095020.2024.2337360>



© 2024 Wuhan University. Published by Informa UK Limited, trading as Taylor & Francis Group.



Published online: 24 Apr 2024.



[Submit your article to this journal](#)



Article views: 2518



[View related articles](#)



[View Crossmark data](#)



Citing articles: 8 [View citing articles](#)

Integrating 3D city data through knowledge graphs

Linfang Ding ^a, Guohui Xiao ^{b,c}, Albulen Pano ^d, Mattia Fumagalli ^d, Dongsheng Chen ^e, Yu Feng ^e,
Diego Calvanese ^{c,d,f}, Hongchao Fan ^a and Liqui Meng ^e

^aDepartment of Civil and Environmental Engineering, Norwegian University of Science and Technology, Trondheim, Norway; ^bDepartment of Information Science and Media Studies, University of Bergen, Bergen, Norway; ^cOntopic S.r.l, Bolzano, Italy; ^dFaculty of Engineering, Free University of Bozen-Bolzano, Bolzano, Italy; ^eChair of Cartography and Visual Analytics, Technical University of Munich, Munich, Germany; ^fDepartment of Computing Science, Umeå University, Umeå, Sweden

ABSTRACT

CityGML is a widely adopted standard for representing and exchanging 3D city models. The representation of semantic and topological properties in CityGML makes it possible to query such 3D city data for analysis in various applications. Nevertheless, the potential of querying CityGML data has not been fully exploited. The official GML encoding of CityGML is mainly an information model used for data storage and exchange, but not suitable for performing complex queries. The most common way of dealing with CityGML data is to store them as tables in the 3DCityDB system. However, it remains a challenging task for end users to formulate SQL queries over 3DCityDB directly for their ad-hoc analytical tasks because of the gap between the semantics of CityGML and the relational schema adopted in 3DCityDB. The technology of Knowledge Graphs (KGs), where an ontology is at the core, is a good solution to bridge such a gap. Moreover, embracing KGs makes it easier to integrate with other spatial data sources, e.g. OpenStreetMap, and to perform queries combining information from multiple data sources. In this work, we describe a CityGML-KG framework to expose the CityGML data in 3DCityDB as a KG. To evaluate our approach, we use CityGML data from the city of Munich as a test area and integrate OpenStreetMap data.

ARTICLE HISTORY

Received 6 December 2023
Accepted 27 March 2024

KEYWORDS

CityGML; OpenStreetMap;
data integration; query
answering; geospatial
knowledge graph; ontology

1. Introduction

CityGML is a widely adopted standard by the Open Geospatial Consortium (OGC) for representing and exchanging 3D city models (Gröger et al. 2012; Kolbe et al. 2021). It defines the three-dimensional geometry, topology, semantics, and appearance of the most relevant topographic objects in urban or regional contexts. The representation of semantic and topological properties in CityGML makes it possible query such 3D city data and often integrated with other data sources. For example, CityGML data sources have been used to integrate with Building Information Modeling (BIM) for visual detection of concealed facilities (Wang and Xie 2022), and with Oracle Spatial for implementing 3D network analysis (Atila, Karas, and Abdul-Rahman 2013).

At the implementation level, CityGML is defined as a GML application schema for the Geography Markup Language (GML) (Gröger et al. 2012). In its most common implementation, CityGML datasets consist of a set of XML files and possibly some accompanying image files that are used as textures. Each XML file represents a part of the dataset, such as a specific region, a specific type of object (such as a set of roads), or a predefined Level of Detail (LoD). The

structure of a CityGML file is a hierarchy that ultimately reaches down to individual objects and their attributes. These objects have a geometry that is described using GML. Another important implementation of CityGML is 3DCityDB (Yao et al. 2018), an open source and platform-independent software suite for 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.

However, the potential of querying CityGML data has not been fully exploited. The official GML/XML encoding of CityGML is mainly used as an exchange format but not suitable for complex queries, in particular those with spatial analysis (Koch and Löwner 2017). The most common way of dealing with CityGML data is to store them in the 3DCityDB system as relational tables and then query them with the standard SQL query language. Nevertheless, for end users, it remains a challenging task to formulate queries over 3DCityDB directly for their ad-hoc

analytical tasks, because there is the gap between the conceptual semantics of CityGML and the relational schema adopted in 3DCityDB.

One possibility to bridge this gap is to use *semantic technology*, which addresses the challenges of data with a complex structure and associated knowledge. At the core of solutions based on semantic technology, we typically have an *ontology* to provide semantics to the data. In computer science, the term “ontology” denotes a concrete artifact that conceptualizes a domain of interest and allows one to view the information and data relevant for that domain in a sharable and coherent way. In the CityGML standard, the semantics is defined as a collection of UML diagrams, which can be used as a basis to construct an ontology. The instances of an ontology are *knowledge graphs* (KGs) (Hogan et al. 2021), where data is structured in the form of a graph. Domain objects and data values are represented as nodes of such a graph, and properties of objects as edges. For CityGML, the nodes in a KG could represent instances of buildings, streets, and surfaces, among others. Moreover, embracing KGs makes it possible to integrate with existing KGs, e.g. Wikidata (Vrandečić and Krötzsch 2014), DBpedia (Lehmann et al. 2015), GeoNames,¹ and LinkedGeoData (Ding, Xiao, Pano, et al. 2021; Stadler et al. 2012). This allows us to express interesting queries that require a combination of information from multiple sources.

The aim of this work is to investigate how to integrate CityGML and other geospatial data sources into a KG, and exploit the resulting KG to support query answering for specific information needs. In the following, we describe a CityGML-KG framework to expose CityGML data as a Knowledge Graph and to integrate it with other data (e.g. OSM data). The CityGML-KG or the integrated KG can be queried using the standard GeoSPARQL query language. To demonstrate the feasibility of this framework, we use the 3D CityGML building data at LoD2 of the municipality of Munich, Germany as test data. We adopt and extend the CityGML ontology created by the University of Geneva and develop a suitable R2RML mapping (Das, Sundara, and Cyganiak 2012) to 3DCityDB. Moreover, as a demonstration of the capability of this methodology for integrating CityGML data with other datasets, we collect OSM data in the same test area. The OSM data is one of the most popular crowdsourcing data worldwide and it contains complementary spatial and semantic information with available CityGML data that can be combined for interesting queries and applications. We stress that the flexibility of the information model in CityGML can support storing the same information in the OSM model as well, however, in reality such datasets are rare, so in practice an integration with OSM or other datasets is of interest. To show

the usefulness of the generated KG, we collect real-world geospatial analytical tasks and formulate them as intuitive GeoSPARQL queries, which show a high degree of expressiveness. Finally, we test three popular KG systems, i.e. Ontop, Apache Jena, and GraphDB, and confirm that the queries can be evaluated efficiently.

2. Related work

2.1. CityGML

CityGML is a data model and exchange format for 3D digital modeling of cities and landscapes (Kolbe 2009). The main advantage of CityGML in comparison to other data formats is that it offers the possibility to integrate semantic information within 3D city models. In 2008, CityGML 1.0 became the international standard in the Open Geospatial Consortium (OGC) (Gröger et al. 2012). The next major version CityGML 2.0 has been approved in 2011. Since then, CityGML has attracted more and more attention from mapping authorities, industries and academic societies. Nowadays, it is widely used for different applications in many countries and regions. The latest version CityGML 3.0 is defined as several standards, most of which have been finalized recently, e.g. CityGML 3.0 CM (conceptual model) and GML encoding.² The most common way of dealing with CityGML data is to store them as relational tables in the 3DCityDB system (Yao et al. 2018) and then query them using the standard SQL query language. As the time of writing, the latest stable release of 3DCityDB v4.1 supports only CityGML 2.0. A major reconstruction is undergoing to support CityGML 3.0 but no information on the expected release date is available. In the following, we still focus on CityGML 2.0 since its toolchain is more mature.

Aiming at modeling 3D cities in the digital world, CityGML covers almost all types of features that could appear in urban area, namely, Building, Water body, Terrain, Transportation, Bridge, City Furniture, Land Use, Tunnel, etc. These features are organized into modules in CityGML. Although CityGML defines levels of detail (LoDs) for all types of features, the LoD of building objects is the mostly agreed and recognized concept in the 3D city modeling community.

In total, there are 5 LoDs defined for building objects in CityGML, ranging from coarse model (LoD0) to very detailed models (LoD4) with geometries and semantic information. As denoted in Figure 1, an LoD0 building model in CityGML is actually a 2D footprint in a closed polygon which is semantically indicated as building object and can be added with various attributes. An LoD1 building in CityGML is a block model with height information,

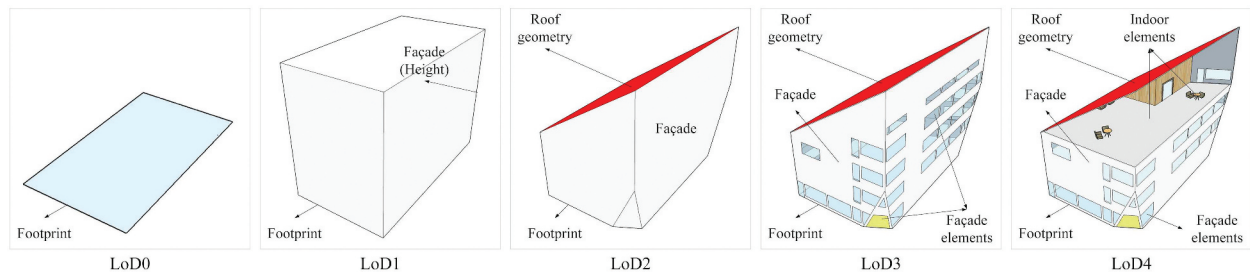


Figure 1. Five LoDs in CityGML.

while an LoD2 building model needs have detailed roof structure but with walls as extruded 3D objects from 2D footprint. In LoD3 building models, architectural details on facades, such as windows, doors and other elements can be modeled in addition to LoD2 models. For a further step, interior objects can be modeled in LoD4. To fulfill task-specific application requirements, Baig and Abdul-Rahman (2013) studied how to reduce data volume by generalizing CityGML data from higher LoDs to lower LoDs.

CityGML is very powerful for modeling 3D cities with rich semantic information. However, its complex and hierarchical structure as well as interoperability issues lead to difficulties and complexity in transformation and decoding for visualization and application scenarios (Noardo et al. 2019). To overcome this problem, CityJSON was developed (Ledoux et al. 2019) by combining the advantages of JSON and CityGML. In other words, CityJSON can be regarded as a JSON implementation of a subset of CityGML version 2.0. In 2023, CityJSON v2.0 was accepted as OGC Community standard. Work is currently underway to adapt CityJSON to the CityGML 3.0 conceptual model.

2.2. Knowledge graphs and geospatial knowledge graphs

The Semantic Web research area deals with the challenges that come with data with a complex structure and associated knowledge. A prominent technology within the Semantic Web is that of knowledge graphs (KGs) (Hogan et al. 2021), where data is structured in the form of a graph. At the core of solutions based on KGs we typically have an ontology to provide semantics to the data. In computer science, the term “ontology” denotes a concrete artifact that conceptualizes a domain of interest and allows one to view the information and data relevant for that domain in a sharable and coherent way. To simplify the sharing and reuse of ontologies, the World Wide Web Consortium (W3C)³ has defined standard languages. We refer here to the Resource Description Framework (RDF) (Manola and Miller 2004) that provides a simple

mechanism to represent the data in a certain domain, and Web Ontology Language (OWL) (Hitzler et al. 2009) that provides a very rich language to encode complex knowledge in the domain of interest.

Geospatial KGs are KGs with geospatial objects, geometries, and their relations. GeoSPARQL is a standard by OGC for the representation and querying of Geospatial KGs (Open Geospatial Consortium 2023). The GeoSPARQL ontology introduces classes like features, geometries, and their representation using Geography Markup Language (GML) and Well-Known Text (WKT) literals, and includes topological relationship vocabularies. GeoSPARQL also provides an extension of the standard SPARQL query interface, supporting a set of topological functions for quantitative reasoning.

Geospatial KGs are often converted from geospatial data sources which are stored in spatial databases or other popular formats like Shapefiles. A systematic approach to such conversion is the *ontology-based data access* (OBDA) paradigm, which enables end users to access data sources through a domain ontology. Typically, the domain ontology imports the GeoSPARQL ontology, and is semantically linked to the data sources by means of a mapping, which is expressed in the R2RML language (Das, Sundara, and Cyganiak 2012) standardized by the W3C. OBDA can be realized in a materialized or virtual fashion:

- In the *Materialized Knowledge Graph (MKG)* approach, the original data sources are first materialized as RDF Graphs using systems, e.g. GeoTriples (Kyzirakos et al. 2018) and Ontop (Xiao et al. 2020), and are then loaded into RDF stores that support geospatial KGs, e.g. Apache Jena,⁴ GraphDB,⁵ and Stardog.⁶
- In the *Virtual Knowledge Graph (VKG)* approach, the content of KG is not generated but can be kept virtual. The ontology and mapping together, called a *VKG Specification*, exposes the underlying data source as a virtual RDF graph, and makes it accessible at query time. For example, Ontop (Ding, Xiao, Pano, et al. 2021) is a popular VKG system that supports GeoSPARQL.

One of the most famous Geospatial KG projects is LinkedGeoData (Ding, Xiao, Pano, et al. 2021; Stadler et al. 2012), which mostly relies on the VKG approach to expose the OSM data as Geospatial KGs. To evaluate the systems for Geospatial KGs, Jovanovik, Homburg, and Spasic (2021) proposed a GeoSPARQL Compliance Benchmark, and Li et al. (2022) carried out an extensive evaluation of several Geospatial RDF triple stores, showing that both MKG and VKG systems have their own advantages. Geospatial KGs have been applied for spatial data quality assessment (Ding, Xiao, Calvanese, et al. 2021; Janowicz et al. 2016) and geodata integration (Ding et al. 2020; Schade and Smits 2012) with an emphasis on using 2D geospatial data collected from e.g. OSM, open data portals, and various sensors. However, research work employing knowledge graphs for 3D geodata integration is still limited.

2.3. Semantic technologies for 3D city models

The OGC CityGML standard does not include an ontology, but the conceptual model in the standard can be used as a basis for creating an ontology. The most well-known CityGML ontology, developed by the Knowledge Engineering @ CUI group at the University of Geneva, is a direct translation of the CityGML XML Schema to OWL with some manual tuning.⁷ This ontology has some minor issues, as discussed by Chadzynski et al. (2021), with respect to the OWL 2 standard but could be easily fixed.

There have been early attempts to convert CityGML to knowledge graphs. In the pioneer work of Chadzynski et al. (2021), they first refined an existing CityGML ontology from the University of Geneva, and then extended a corresponding data transformation tool that was originally designed to work alongside CityGML, which allowed for the transformation of the original data into a form of semantic triples. Various scalable technologies for this semantic data storage were compared and Blazegraph was chosen due to the required geospatial search functionality. Regarding the usage of the virtual approach, an earlier version of this work (Ding et al. 2023) described the CityGML VKG framework to expose the 3DCityDB as a VKG and tested with CityGML building data at LoD2 in the municipality of Tartu, Estonia. Many applications in the context of urban informatics require detailed information about the physical urban environment, which necessitates the integration of 3D city data with other data sources. Ahmadian and Pahlavani (2022) reported the integration of OpenStreetMap and CityGML using the formal concept analysis approach. Most work focuses on the integration of CityGML and BIM. Huang et al. (2020) proposed two approaches to integrate and reconcile city models and BIM in the context of solar

energy simulations, where BIM data is stored in IFC and the city model in CityGML (LoD2). The first approach is to perform a schema matching in an ETL tool, so as to convert and import window information from the IFC file into the CityGML model to create a LoD2–3 building model. In the second approach, they adopted a semantic web approach, in which both the BIM and city models are transformed into knowledge graphs (linked data). City models and BIM utilize their respective but interlinked domain ontologies. Particularly, two ontologies are investigated for BIM data, i.e. the ifcOWL ontology⁸ and the building topology ontology (BOT).⁹

3. Methodology

In what follows we describe the approach we adopted to generate the target KG from CityGML data and to integrate other data sources. An overall view, denoted as a pipeline, is unveiled through the utilization of the *Business Process Model and Notation (BPMN)*¹⁰ diagram represented in Figure 2.

The scope of the whole process is twofold. Firstly, it allows for the generation of a KG to support query answering over CityGML data. Secondly, it allows for the evolution of the created KG by importing new data and knowledge, thus enabling the extension of question-answering services.

As shown in Figure 2, this methodology consists of four main phases: (i) *Initialization with CityGML data* (hereafter “Initialization”), (ii) *KG construction*, (iii) *Integration of further resources* (hereafter “Integration”), and (iv) *Application*. These phases are composed of sub-tasks or steps, which, in turn, may receive and/or produce different kinds of data. Differently, the KG group represents the final output to support the query-answering activities, which are represented in the Application phase group. The output KG can be either a VKG or an MKG. The VKG contains two sub-components, namely an ontology, and a mapping function which are used to generate the RDF triples from the physical storage on demand. Representing the KG as an MKG instead eliminates the need to maintain a virtualized pipeline for RDF data, with the trade-off of larger space requirements and the need to rematerialize the RDF triples every time the source data changes. All these components can then evolve through the steps of the Integration phase.

In this setting, the *Initialization* phase has the primary goal of generating a reference data storage out of CityGML data and, also, searching for and providing a suitable CityGML ontology as a baseline version of the knowledge graph. For the creation of the reference data storage *the input CityGML dataset is embedded into a relational database* (See Generate SQL in Figure 2), mapping each row in the data into entities

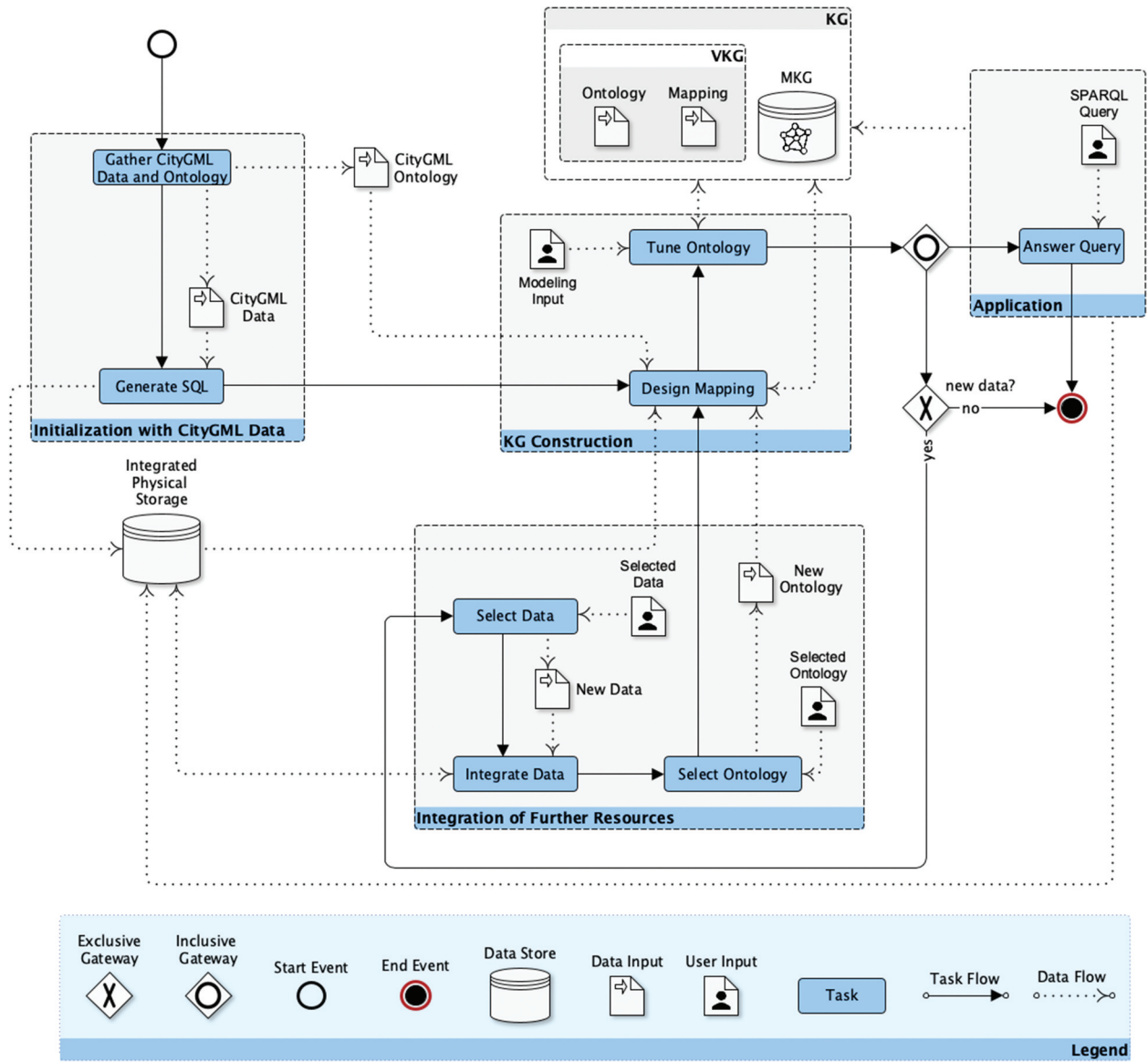


Figure 2. Overview of the CityGML-KG framework.

and columns in specific information fields so that the data can be then queried, retrieved, stored, and possibly updated. Note that, to address this step, albeit the multiple automatic and comprehensive solutions are available, some *ad hoc* customization sub-steps may be involved. This is due to the fact that the output physical storage of this phase must be compliant with the technology used to generate the knowledge graph in the following phase. For instance, if the solution for generating SQL databases from the input data uses XML attributes containing (semi-)structured information, a customization step would be needed to generate multiple fields out of each XML attribute.¹¹

The second phase, *KG construction*, crafts a KG that can be used as a reference point for the query-answering activities. Note that this step can be iterated multiple times. The first time is after the initialization phase, and the inputs for KG construction will naturally be the *CityGML* ontology, the *Physical Storage* hosting the *CityGML* data, and the *mapping* that is

necessary to connect the former to the latter. In the following times, the inputs of this phase are the result of the integration phase. Either way, the KG Construction phase is mainly concerned with the definition of the ontology and the related mappings, with the main scope of (i) defining the set of concepts, relationships, and properties within the reference domain of knowledge; (ii) capturing the semantics of the stored information, by enabling extended reasoning and inference capabilities; and (iii) fostering interoperability among the data sources to be integrated. A key aspect here is also to find an already existing ontology that covers as much of the semantics of the selected data as possible. Once the ontology is selected a mapping step is performed. The database generated through the initialization phase is then aligned with the ontology concepts. If the information cannot be straightforwardly mapped, manual intervention is required. This mainly involves a modification of the selected ontology to properly account for the

information in the physical storage. For example, if a property in the input dataset is not present in the ontology, the ontology can be manually extended with the required property. Once the ontology is tuned according to the dataset requirements, the KG is created and ready to support the query-answering application.

After completing the phases of the initialization and KG construction, a reference KG is ready for query answering. However, at this point, our approach allows for the integration of more data sources and then for the creation of an extended KG, on top of the previous version. This evolution is addressed through the *Integration* phase. Here the main tasks are (i) the selection of new data by the user, (ii) the integration of the new data with the existing physical storage, and (iii) the selection of a new ontology or new ontological information by the user, in order to account for the newly integrated data. Task (ii) takes place with the support of an automatic component that involves two sub-steps, namely (ii.a) post-processing where heterogeneous geo-object types, data formats, and coordinate reference systems (CRS) are harmonized and unified and (ii.b) spatial matching where the similarity between geo-entities from different datasets is calculated.

The final phase *Application* is dedicated to using the output KG. Here the user, via an *ad hoc* interface, is enabled to request information via SPARQL queries,

which can potentially be returned in either textual or visual format.

4. System architecture

In this section, we describe how the conceptual methodology framework in Section 3 can be realized in a concrete system. As shown in Figure 3, in the context of our particular application scenario, we examine two typical 3D and 2D geospatial data sources, namely *CityGML* and *OpenStreetMap (OSM)*. *CityGML* data is used throughout the whole phases, while *OSM* data, one of the most comprehensive and widely used geospatial data sources, is adopted to illustrate the *Integration* phase. More specifically, we use LoD2 *CityGML* building data retrieved from the Bavarian Open Data portal¹² in the central area of Munich, Germany and *OSM* data in the same area as a demonstration. Please refer to Section 5.1 for further details on the test area and datasets.

4.1. Initialization with CityGML data

The initialization phase generates a reference data storage out of *CityGML* data. We selected the *3DCityDB schema* as the preferred solution to import *CityGML* data into an SQL database. *3DCityDB* as a software solution provides both a predefined SQL schema¹³ and importer-exporter¹⁴ tool which can

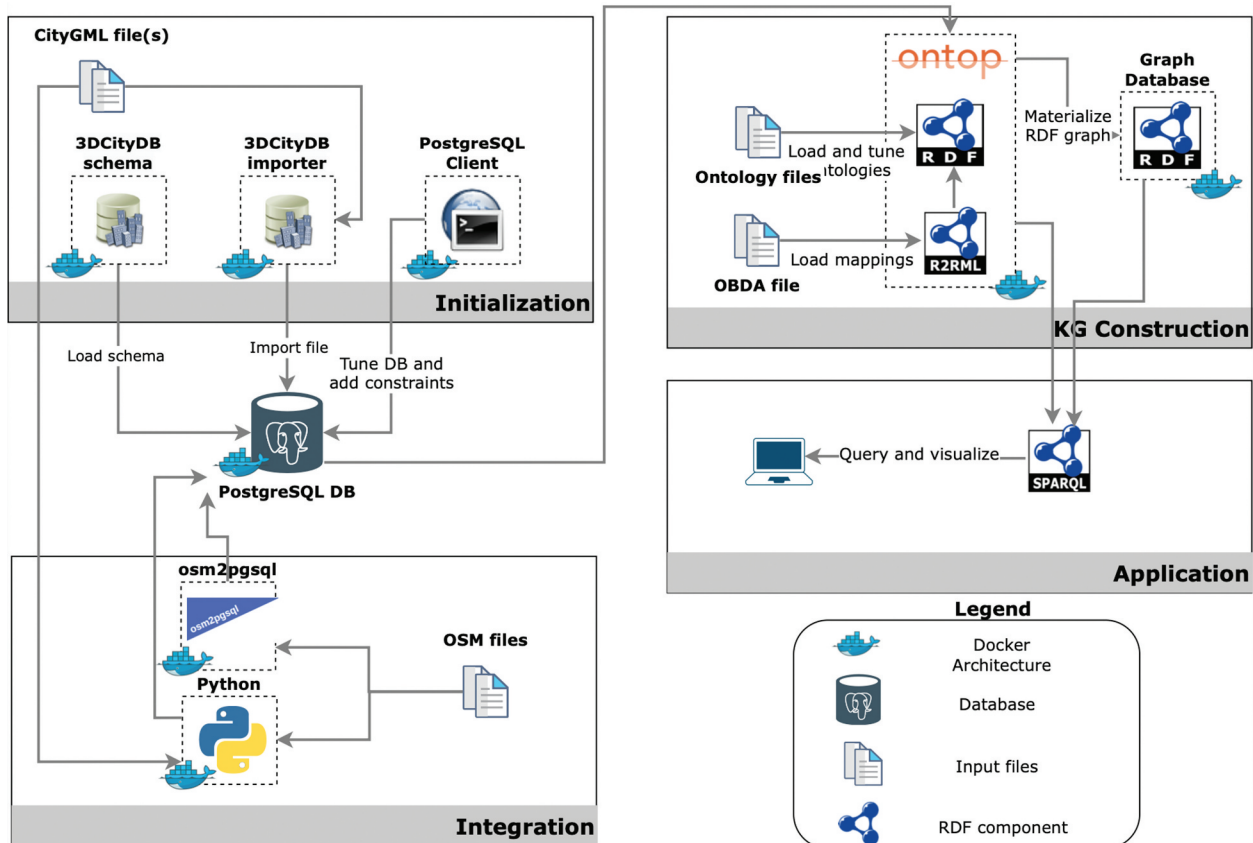
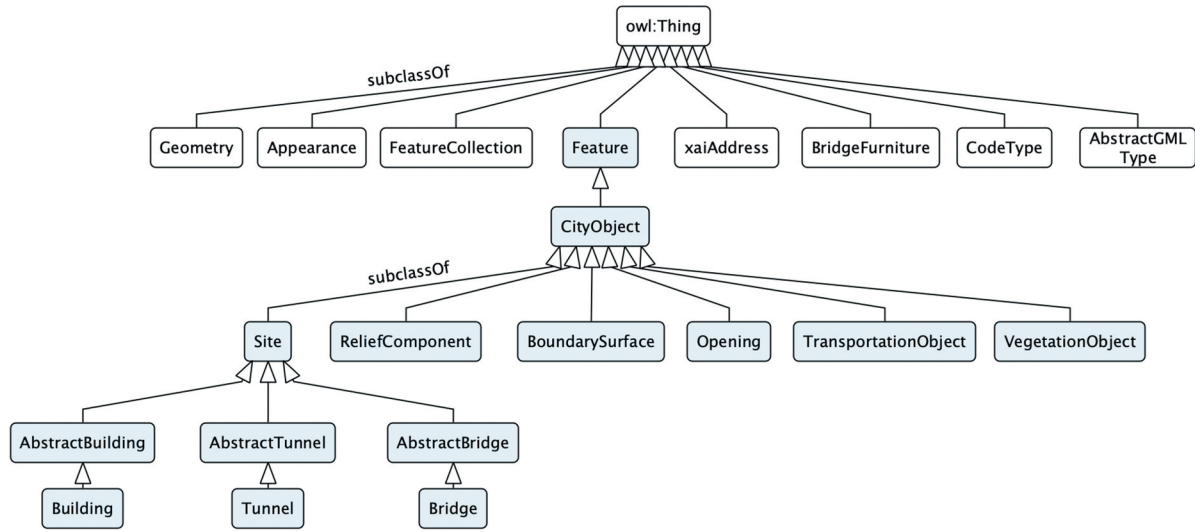


Figure 3. VKG over CityGML: Architecture.

Table 1. Excerpt from the CityGML building table.

Id	Objectclass_id	Building_root_id	Roof_type	Measured_height	Lod2_solid_id
10	26	10	1000	13.363	117
54	26	54	3100	13.99	315
248	26	248	1000	17.362	1258

**Figure 4.** A subset of the concepts in the CityGML ontology with the subtree of the class feature highlighted.

process the cumulative addition of an arbitrary number of CityGML files into the database. The software provides the option to use PostgreSQL or Oracle as the backend for the relational data storage. In this work, we chose PostgreSQL in particular because it is open source and its geospatial extension PostGIS is renowned for its high adoption and maturity in the geospatial domain.

An excerpt of the table building with three records is presented in Table 1 (omitting any attributes with missing data). Every building is uniquely identified by the attribute id and has a corresponding LoD2 solid identifier lod2_solid_id. The latter is mapped onto its respective polyhedral surface serialization in the surface_geometry table. Further sample data is provided in Figure 5.

Finally, we note that despite the comprehensive default SQL schema provided by 3DCityDB, a further step is needed to *tune DB and add constraints*. For example, the attribute address in the default database schema is encoded as XML strings and has to be decomposed into more specific attributes, e.g. administrative area, thoroughfare, etc. Relevant constraints like primary and foreign keys are added to enhance the efficiency.

4.2. Knowledge graph construction

Our architecture supports the construction of KGs in both VKG and MKG variants. We first construct the VKG utilizing the *Ontop* system. The main activity is

to develop/refine the *ontology* and create *mappings* to link the terms (classes and properties) in the ontology to the data sources. We will also use *Ontop* to materialize the RDF triples into an MKG, and load it into a triple store system like Jena or GraphDB.

4.2.1. Ontology

We adopt the most prominent and well-known *CityGML* ontology version 2.0¹⁵ developed by the University of Geneva for the ontology component of the KG construction phase. We first *load and tune ontologies* from the ontology provided. Following validation of the ontology, there were 92 declarations of object and data properties using the same IRIs, which makes the ontology invalid. The same inconsistencies have been diagnosed in previous research (Chadzynski et al. 2021). We tackled the issue in a similar fashion resolving any inconsistencies manually depending on the most intuitive definition of each property. In order to review and resolve these inconsistencies and get an overview of the collection of ontologies we utilized the open-source tool Protégé version 5.6.1. Figure 4 shows part of the top-level classes of the selected ontology and the sub-tree starting from Feature and Building, which are of primary interest in this study.

We also note that in addition to the primary CityGML ontology, auxiliary ontologies were also employed with the following prefixes: geosparql, gml, sf, sosa, core, Dublin_core_elements. The GeoSPARQL ontology (Open Geospatial Consortium 2023), for instance, allows the differentiation of

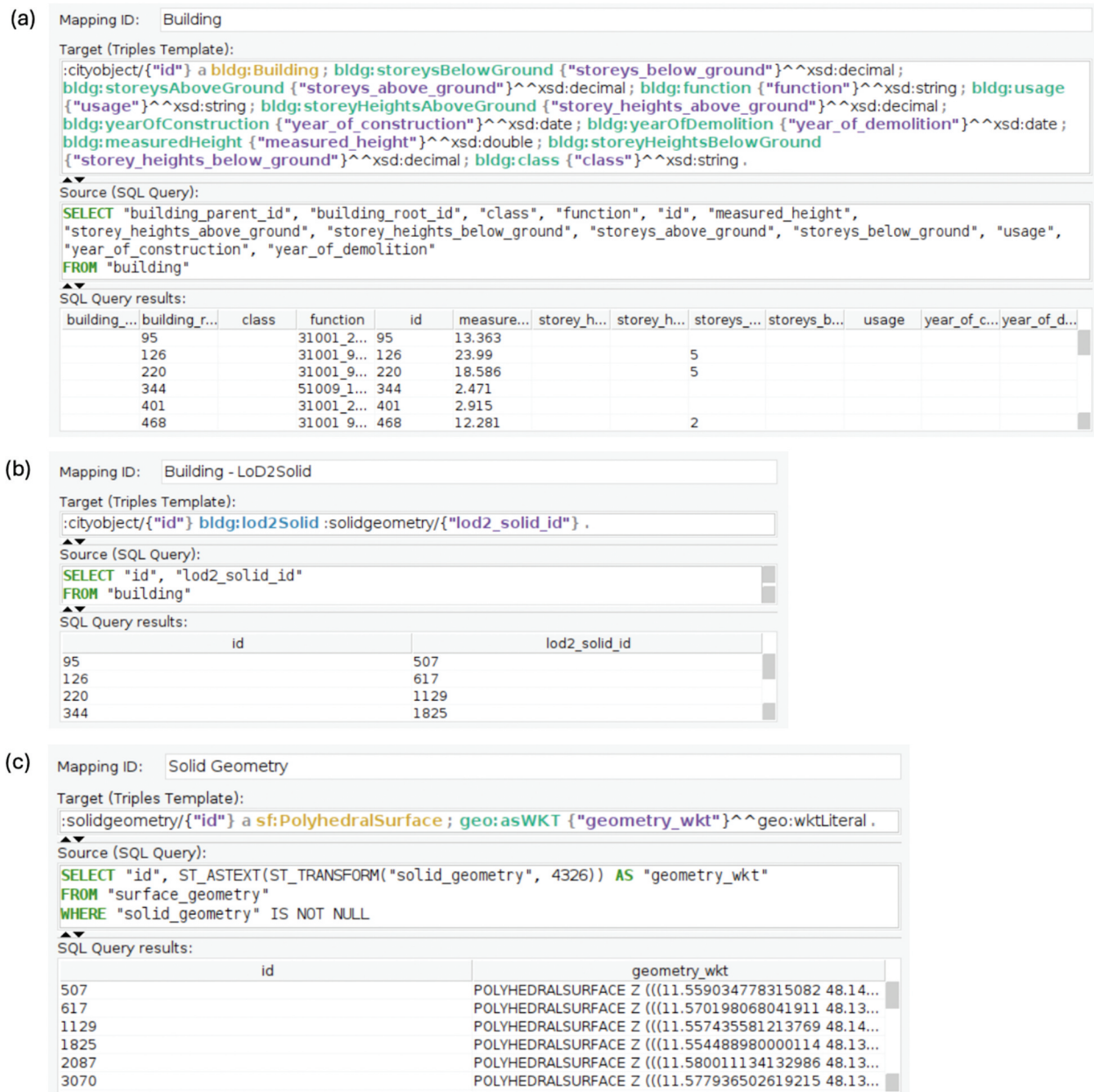


Figure 5. Three example mappings shown in the Ontop Protégé Plugin.

geometric classes such as polyhedral surfaces from standard surfaces while complying with the standard OGC recommendations. For other ontologies, gml¹⁶ encodes the transport and storage of geographic information, sf defines simple feature geometries,¹⁷ sosa¹⁸ broadly defines sensors and their observations, core (or rather skos core)¹⁹ provides a standard way to represent knowledge organization systems and finally Dublin_core_elements²⁰ describes physical and virtual resources (e.g. books, video, artworks etc.).

4.2.2. Mappings

Mapping design is the most crucial user-centric step in generating a VKG. Individual RDB2RDF mappings exploit attributes from the PostGIS database to populate the RDF graph of CityGML. Consequently, SQL queries have to be formulated to map individual

3DCityDB attributes to their respective ontological concepts. Due to the limitation of LoD2 Bavarian data (and any open CityGML data we can find), which contains exclusively buildings, and the lack of any complementary real-world LoD3 files, many 3DCityDB tables are empty. Therefore, while for completeness purposes any column from the 3DCityDB schema that could be mapped has been mapped to an ontological concept, in practice no triples can be generated from many of these mappings.

A mapping consists of three components: a mapping ID, a source, and a target. The mapping ID is an arbitrary but unique mapping identifier. The source refers to an SQL query expressed over a relational database to retrieve data. The target is RDF triple pattern(s) that uses the answer variables from the preceding SQL query as placeholders.

Figure 5 illustrates three example mappings written in the Ontop Protégé plugin editor – define a building, link a building with the corresponding solid geometry, and define the serialization of that geometry. More specifically, in the first mapping in Figure 5(a), the class Building is mapped with its respective data properties such as building height, storeys above and below ground, function, year of construction, etc. The second mapping in Figure 5(b) shows that object property bldg:lod2solid links any building with its respective solid geometry identifier. We distinguish between a solid geometry class and its respective serializations which are properties of the class. In the third mapping in Figure 5(c), class sf:PolyhedralSurface defines objects of type polyhedral surface and their respective Well-Known Text (WKT) geometry serialization.

4.2.3. KG materialization

With the completion of all the ontology and mappings, the CityGML VKG has been successfully created. This VKG can be queried already by Ontop, or be materialized to use native MKG systems. The MKG is constructed by utilizing the functionality of Ontop to generate RDF triples or data assertions based on the ontology, mappings, and physical storage, which were discussed in detail in the previous section. The resulting triples can be in turn loaded into RDF triple stores like GraphDB, Apache Jena, RDF4J, and similar tools to facilitate query answering via SPARQL. Further details on how these triple stores can exploit the materialized MKG are provided in [Section 5](#). [Figure 6](#) shows a sub-graph from the example mappings representing a building and its LoD2 geometry and address.

4.3. Integration of OSM data

Below we describe the steps of integrating further geospatial data sources in our architecture by using OSM data as an example. For any other generic geospatial data, the integration task depends on the type of data we wish to integrate and on the existing popular ontologies. With regard to OSM data, as mentioned in [Section 2](#), the LinkedGeoData project has already leveraged the loading of OSM data into PostgreSQL and developed an ontology and mapping that can be reused in this work. What’s missing is the linking between CityGML and OSM data at both the data level and the ontology level. This requires computing the correspondence between the data items, i.e. buildings, between these two data sets, and creating additional suitable mappings and ontological axioms to capture these correspondences. To handle this heterogeneity issue we leverage an entity resolution step that produces a reference linkage table for the generation of the output physical storage as PostgreSQL DB. Below we present a geometry-based method for linking entities in CityGML and OSM data sources and incorporating the results in the KG.

4.3.1. OSM and CityGML data linking

Due to the heterogeneity between the CityGML and OSM datasets, we cannot expect that the resulting data linking is always 1:1. The building information of OSM data mainly consists of the building footprint layer (polygons) and the point of interest (POI) layer (points). In contrast, the CityGML data is three-dimensional, hence we primarily rely on the ground surfaces of the CityGML buildings. The CityGML dataset normally has more detailed information about the buildings. In particular, CityGML buildings

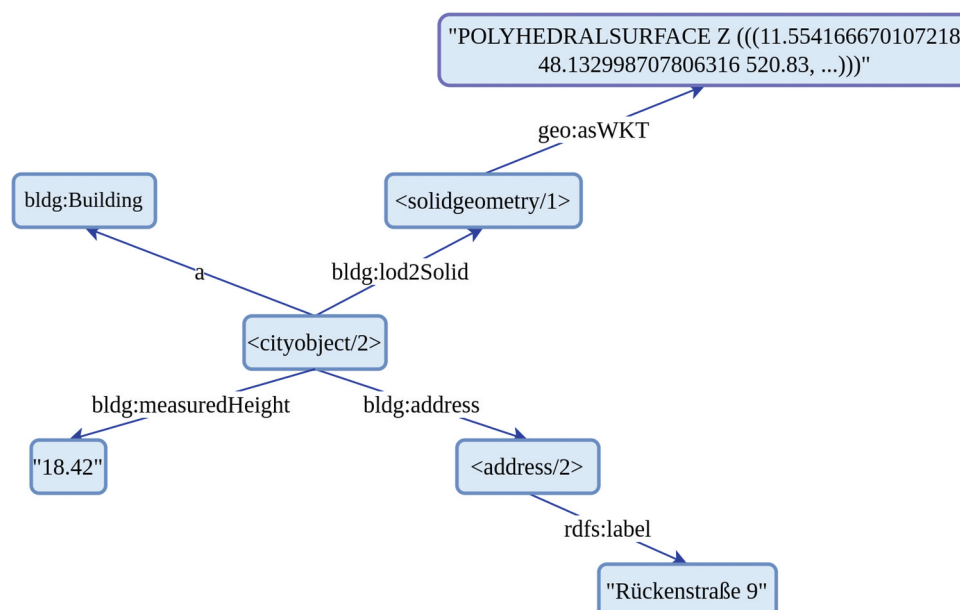


Figure 6. Triples for CityGML.

frequently encompass minor ancillary features like stairs and garages, which are often absent in OSM building footprints, and may lead to a $n:1$ matching result.

We propose a three-step method of data linking:

- (1) Computing direct spatial correspondence using CityGML ground surfaces and OSM polygons,
- (2) Exploiting the adjacent ground surfaces in CityGML to enrich the results of step (1), and
- (3) Matching OSM POI points with CityGML buildings.

Note that more sophisticated approaches, e.g. formal concept analysis (Ahmadian and Pahlavani 2022), can be adopted in this architecture as well, but to simplify the presentation, we only use the geometry-based approach in this work.

In the first step of spatial matching, since the CityGML data in this study only contains building information, we refer to the linking of CityGML and OSM polygons as the linking of building information between them. Given any CityGML building ($bldg$) and OSM polygon (osm), their direct spatial correspondences are identified based on Equation (1) (Fan et al. 2014; Liu et al. 2023), derived from individual

areas of any CityGML and OSM polygons ($Area(bldg_i)$, $Area(osm_j)$).

$$\frac{Area(osm_i \cap bldg_j)}{\min(Area(osm_i), Area(bldg_j))} \geq t \quad (1)$$

where $Area(osm_i \cap bldg_j)$ represents the overlapping area between the i -th OSM polygon and the j -th CityGML polygon; t is an empirical hyperparameter, which can be adjusted based on the spatial consistency between the two datasets. Following Fan et al. (2014), there are four possible matching results based on the ratio: 1:1, 1:n, $m:1$, $m:n$ (examples illustrated in Figure 7). Relation 1:1 indicates that an OSM building and a CityGML building are uniquely matched with each other (Figure 7(a)). Relation $m:1$ represents multiple OSM buildings matching with one CityGML building (Figure 7(b)) and relation 1:n the opposite case (Figure 7(c)). Relation $m:n$ represents at least two OSM buildings matched together with at least two CityGML buildings (Figure 7(d)).

The second step specifically endeavors to include adjacent ground surfaces as secondary matched (adjacent) relations, guaranteeing the inclusion of all amenities. In the examples illustrated in Figure 8, $bldg2$ and $osm1$ are matched as adjacent if the following

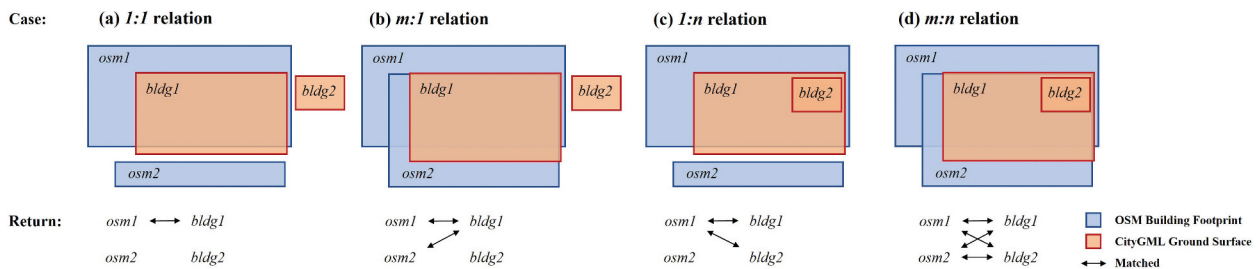


Figure 7. Schematic diagram of the four spatial matching relations.

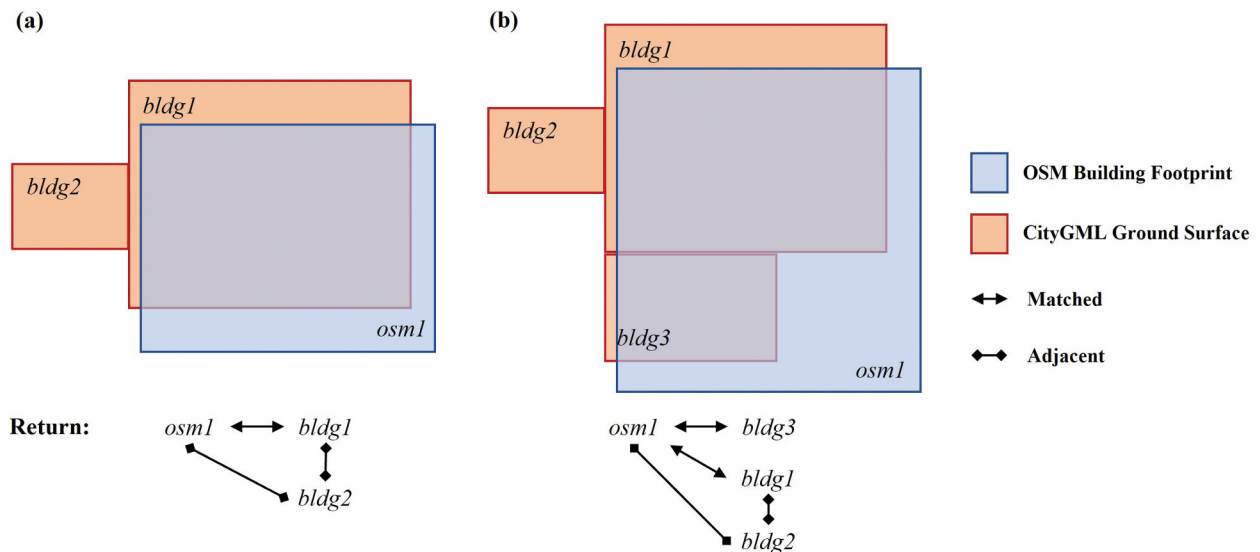


Figure 8. Schematic diagram of the two adjacent identification situations in the OSM building $osm1$ perspective: (a) 1:1 relation with adjacency, and (b) 1:n relation with adjacency.

three conditions are met: (a) *bldg1* directly matches *osm1*, (b) *bldg1* and *bldg2* are adjacent, and (c) there is no other match for *bldg2*.

The third step aims to match the CityGML data with OSM POIs to enhance the semantic information of CityGML. The OSM POIs contain the place information in the buildings, e.g. various shops on different floors. The spatial locations of building-related POIs are based on the building footprints. Thus, we applied OSM's building footprints as a mediator to determine the spatial relationship between OSM POIs and CityGML buildings. As shown in Figure 9, given any POI, if the OSM building footprint where it is located matches a CityGML ground surface, the POI also matches the corresponding CityGML ground surface.

Empirical matching results As for the selected study area in Munich, the input OSM data contain 3839 building footprints while the input CityGML

data contain 5728 ground surfaces. Previous studies have considered a minimum threshold t of 30% is necessary to determine the matching relationship (Fan et al. 2014; Liu et al. 2023). Empirically, we tried several settings and chose to set the tolerance threshold t to 50% in this case study according to the performance on both OSM and CityGML data. In order to evaluate the correctness of spatial linking workflow, 50 randomly selected building polygons were manually examined, where all of them were correctly identified as matched or adjacent.

Figure 10(a) shows the results of Step 1 (spatial matching) between CityGML and OSM polygons in the study area. The majority of CityGML ground surfaces are successfully matched with OSM polygons. The 1:1 relations account for 42.92% (2090 buildings). The 1: n relations (shown in Figure 10(c,d)) make up 16.20% (789 buildings) while the m :1 relations only

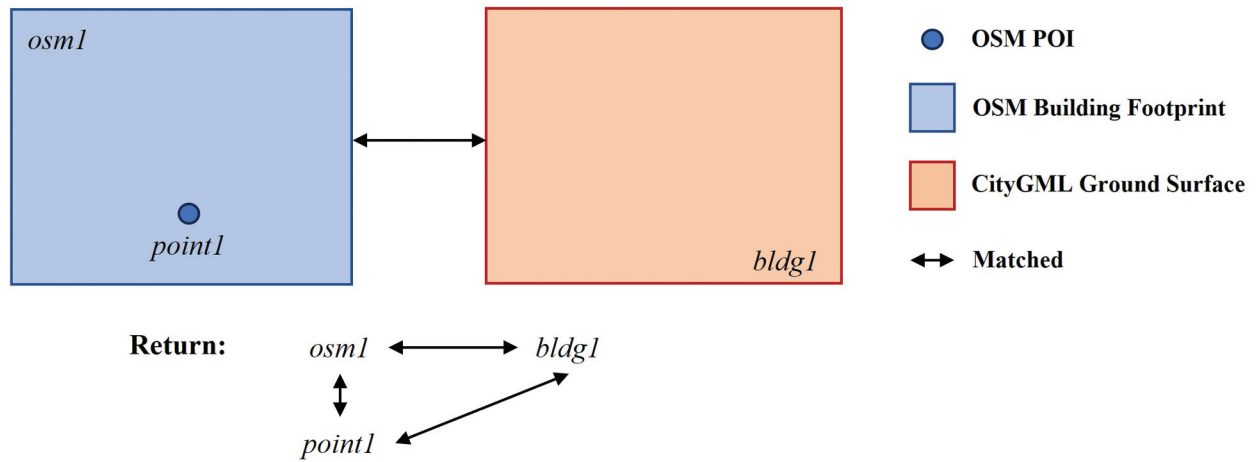


Figure 9. Schematic diagram of the spatial match between OSM POIs and CityGML ground surfaces.

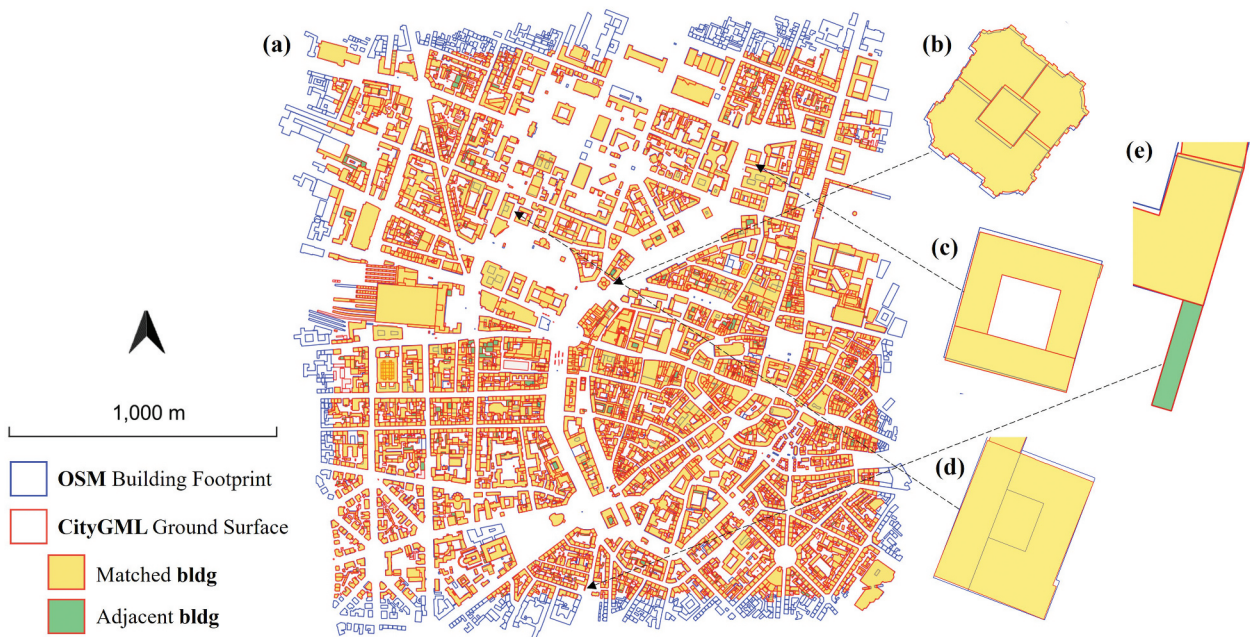


Figure 10. Distribution of (a) the spatial integration result in the case study and the cases of four common spatial relations, i.e., (b) 1:1, (c) 1: n , (d) m :1, and (e) adjacent relations.

account for 5.87% (286 buildings) of the total in Step 1. This indicates that CityGML provides more detailed information, representing individual building accessories (e.g. staircases) as separate ground surfaces. 0:1 relations account for 21.17% (1031 buildings) in *Step 1* for the same reason. For instance, in [Figure 11\(a,b\)](#), the CityGML polygons pointed by the arrows should be included in their main buildings and linked with the corresponding OSM polygons.

After Step 2, these unmatched CityGML polygons are defined as matched by their adjacent OSM polygons. As a result, the 0:1 relations decrease from 21.17% (1031 cases) to 5.54% (270 cases), which demonstrates the necessity of including the adjacent structures as *Step 2*. Additionally, there is a 12.83% (625 cases) occurrence of 1:0 relations, where certain OSM buildings are absent in the CityGML data. This is mainly due to the slightly larger coverage of the downloaded OSM data compared to the CityGML data (as demonstrated in [Figure 10](#)), ensuring that no CityGML buildings on the tile edges are missing in OSM. Ideally, overlapping polygons within a dataset's ground layer should be avoided. For instance, locations marked by arrows in [Figure 11\(c,d\)](#) should not serve as two buildings' foundations. The integration step is intentionally designed to account for such specialized relations. In this study area, only 49 cases of m:n relation (1%) were identified.

As for Step 3, within the specified study area, we have successfully matched 2718 OSM POIs with CityGML buildings.

4.3.2. Modeling the linking results into the KG

Integration at the relational database level needs to be further upstreamed to the ontology and knowledge graph level. First, in order to query OSM concepts, a supplementary ontology, namely the *LinkedGeoData* (LGD) ontology, originally defined by Stadler et al. (2012), was adopted. LGD defines over 1200 classes and 700 properties by leveraging the most ubiquitous tags present in OSM data. LGD enriches the CityGML

KG with classes that represent OSM points of interest like hotels, residential buildings, and primary highways, as well as properties such as business opening hours and websites.

Second, there is a task to model at the KG level the connections between the still disjoint CityGML and OSM sub-KGs. Utilizing existing owl terms like owl:sameAs is insufficient since we do not model identity or matches between individual buildings but other properties like adjacency, and potential associations between buildings could be enriched beyond that.

Modeling the spatial matching results from the database to the KG necessitates the reification of the association between a CityGML building and OSM building. In practice this requires creating an additional class to represent this relationship which we define as *Association_CityGML_OSM*. This relationship allows the addition of further properties for *Association_CityGML_OSM*, e.g. it can now model both matched buildings and adjacent buildings. An example of how an association would be expressed in the KG is shown in [Figure 12](#). The upper part illustrates the schema (i.e. classes, properties, and their relations), and the lower part concrete instances corresponding to the example from [Figure 8\(a\)](#) where e.g. an OSM building lgdo:way/osm1 is linked to both a matching and adjacent CityGML building surface as defined respectively by gmlid/bldg1 and gmlid/bldg2. Both the match and adjacency relations are modeled as subproperties of linkage.

5. Experiments

In this section, we conduct a series of experiments in order to evaluate the following aspects:

- (1) the expressiveness capabilities of the KG. We determine whether a KG constructed over CityGML data and further integrated with additional ad hoc data sources suffices in answering legitimate semantic queries designed by domain experts;

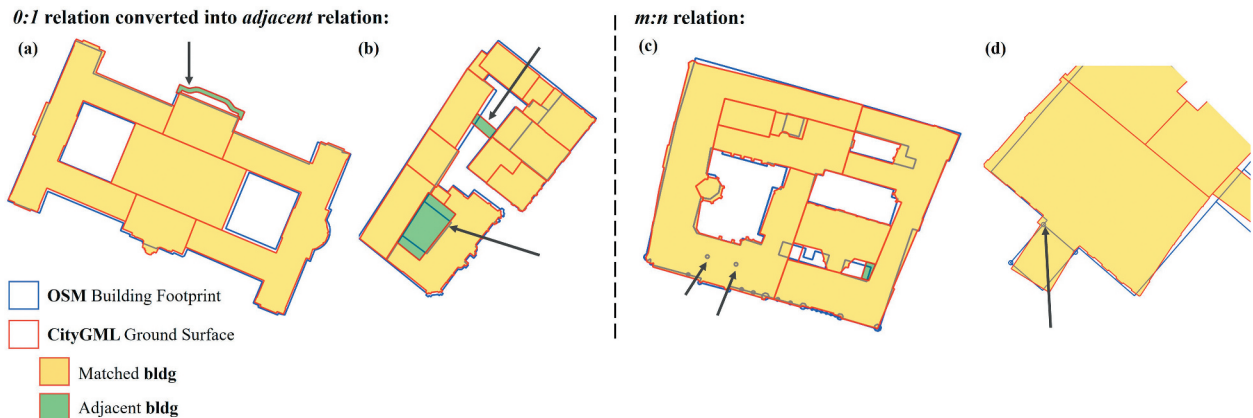


Figure 11. Cases of 0:1 relation converted into adjacent relation (a,b) and cases of m:n relation (c,d).

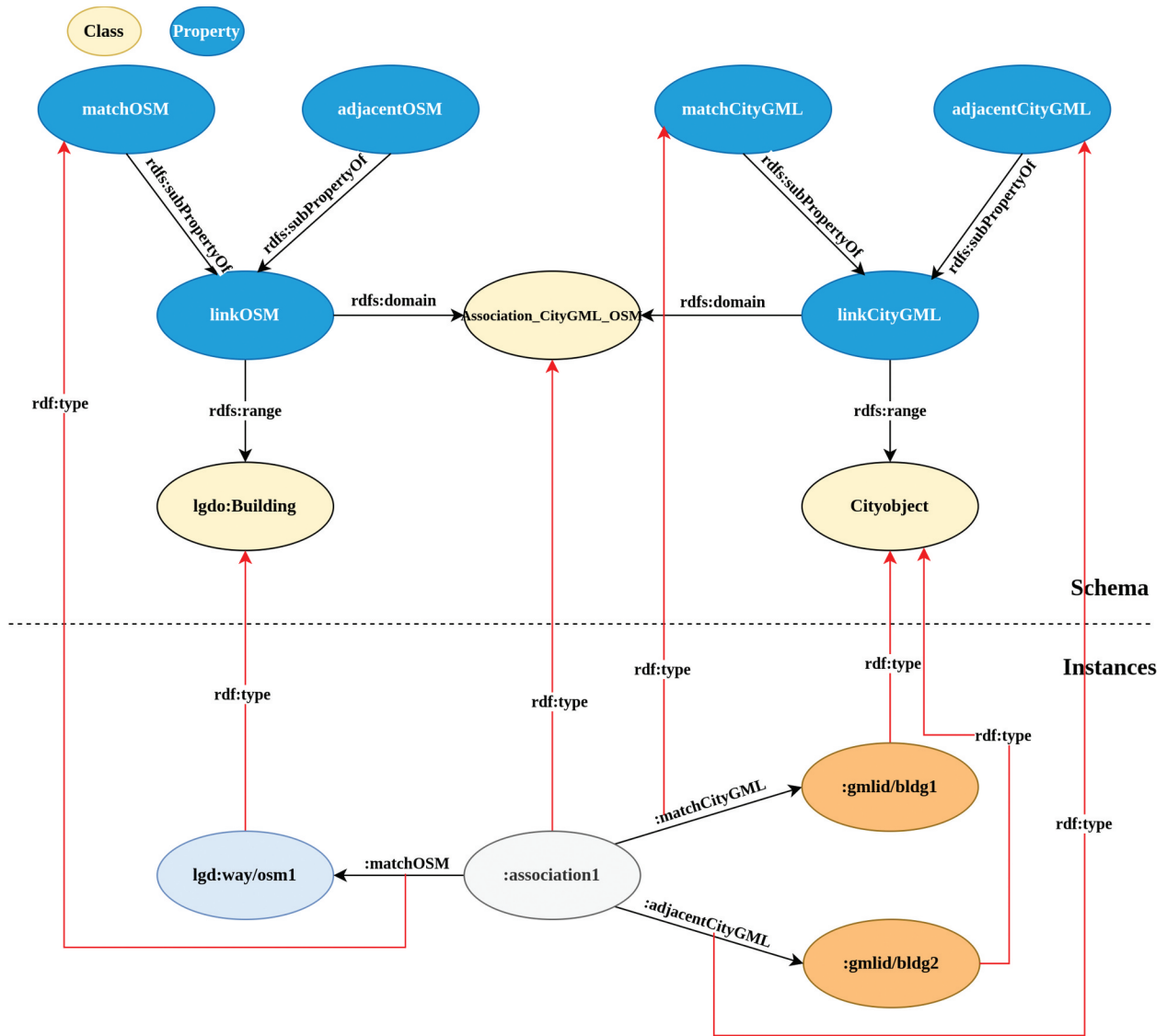


Figure 12. Modelling CityGML and OSM association in the KG.

- (2) the performance of the query evaluation with representative KG systems, including both VKG and MKG systems. This will serve to determine whether results can be retrieved within a reasonable time frame from domain experts.

The experiments are reproducible by running the respective queries and setup described in the online appendix²¹ as well as Appendix A.

5.1. Experimental setup

The experiments are conducted on a normal laptop machine running 4 cores (Intel(R) Xeon(R) Gold 6154 CPU @ 3.00 GHz), 16GB RAM, and 350GB SSD hard disk, running Ubuntu operating system. The whole experiment environment has been setup as Docker containers to encapsulate all necessary software, which means experiments can be conducted under any operating system. For storing and querying

CityGML data, we use the Docker image of 3DCityDB v4.1.0 that comes with PostgreSQL v15 and PostGIS v3.3, corresponding to the latest versions available at the time.

Three KG systems that support GeoSPARQL have been selected for the experiments: one VKG system Ontop and two MKG systems (a.k.a triple stores) Apache Jena and GraphDB. For GraphDB and Jena, a preparatory step of materializing all the triples is necessary. We carry out this using Ontop, and then load the file into Apache Jena and GraphDB as an input. Further descriptions of the evaluated systems are provided below:

- Ontop²² is an open-source software project that focuses on providing a platform for efficient querying of relational databases using Semantic Web technologies, specifically the RDF data model, SPARQL query language, OWL 2 QL ontology, and R2RML mapping language. Ontop also supports the GeoSPARQL query

language over PostgreSQL/PostGIS database. We use the latest Ontop v5.0.2 in this experiment.

- Apache Jena²³ is an open-source framework for Java that allows for reading, writing, and querying RDF graphs. Apache Jena Fuseki²⁴ is a sub-project of Jena which is a SPARQL server, combined with a UI for admin and query. TDB is a component of Jena for RDF storage and query which can be used as a high-performance RDF store. Unlike Ontop, Apache Jena Fuseki does not handle any SPARQL-to-SQL translation or virtualization but rather utilizes RDF triples as input. GeoSPARQL and spatial index support are available via the Jena-fuseki-geosparql extension. Note that an RDF dataset needs to be “wrapped” as a GeoSPARQL dataset since the default Apache Jena Fuseki installation does not support GeoSPARQL query functionalities. We use the latest Apache Jena v4.8.0 in this experiment.
- GraphDB²⁵ is an RDF store developed by Ontotext, which supports SPARQL 1.1. OWL reasoning and is compliant with W3C Standards. It is a materialization-based system in a similar sense to Apache Jena, but although commercial, it does offer a free limited version. For the purpose of our experiments, we use the latest GraphDB 10.2.2, which supports all SPARQL 1.1 and GeoSPARQL functionalities.

Study area and data. The experiments are carried out in the central area of Munich, Germany (Figure 13 (a,b)). Munich serves as both the capital and the largest city of the German state of Bavaria. Ranking as the third largest city in Germany, following Berlin and Hamburg, Munich spans an approximate area of 310.43 km². Our selected study area covers an area around 2 km × 2 km, and is a construction-dense area and can provide an adequate gauge of query performance. Figure 13(c,d) depict the test datasets in the area of interest from CityGML and OSM respectively. The applied CityGML data is obtained from the official 3D building models from the Bavarian Surveying Administration.²⁶ It is a Level of Detail 2 (LoD2) data with ALKIS²⁷-compliant standard roof shapes and descriptive attributes. The specific tile of our study area is with ID 690_5336. The OpenStreetMap (OSM) data is an well-known open source crowdsourcing data (Haklay and Weber 2008). We spatially clipped the building footprint data of our study area based on the spatial extent of the selected CityGML tile.

5.2. Expressiveness test

Geospatial queries are used in many application scenarios, e.g. urban planning or management, disaster management, tourism, and energy (solar panels). In

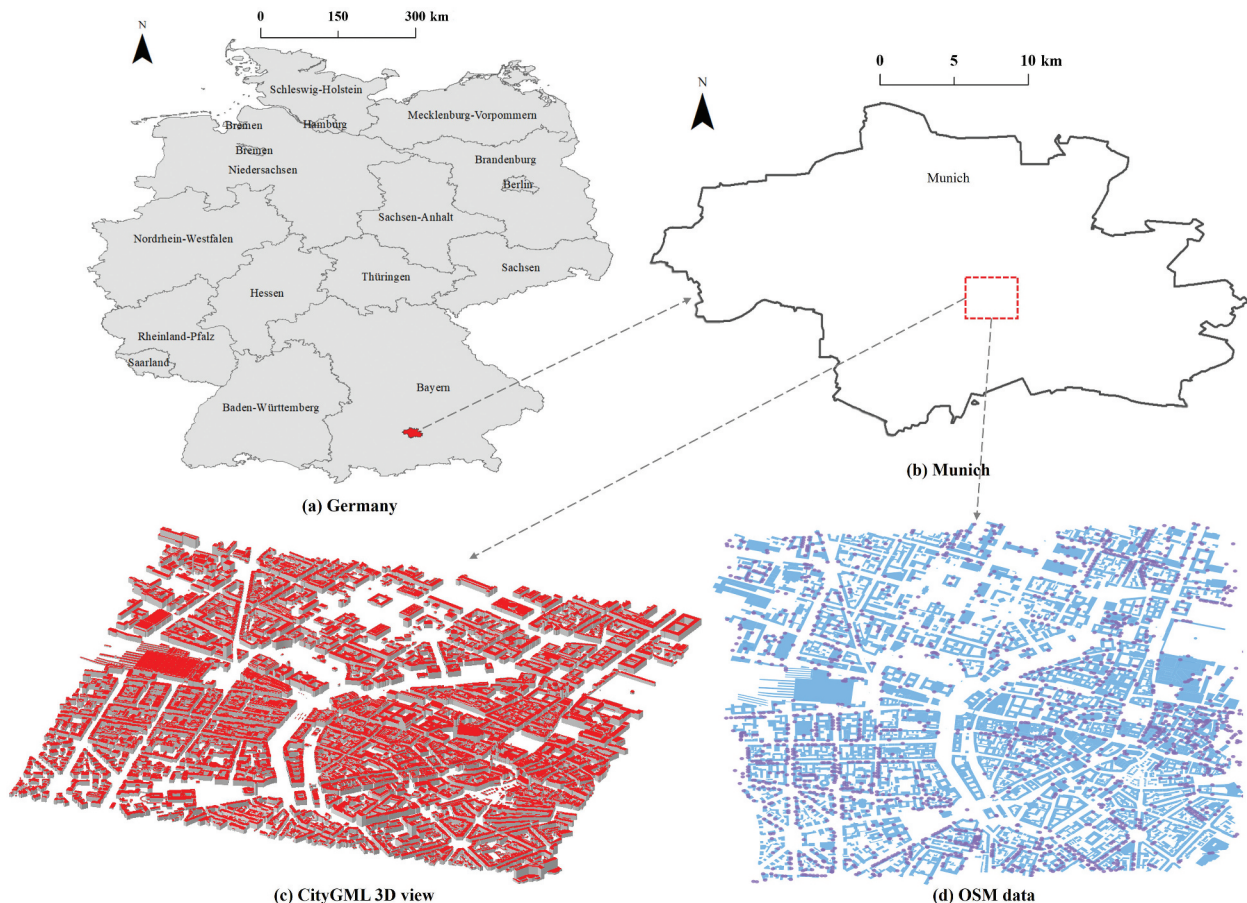


Figure 13. Study area.

order to test the expressiveness of GeoSPARQL queries that can be formulated over the KGs constructed from CityGML and OSM data in this work, we have collected 10 queries from domain experts and formalized them as GeoSPARQL queries. These queries encompass not only conventional question types about 3D buildings but also those designed to address pragmatic real-world demands.

5.2.1. Queries

Queries Q1–Q5 represent basic information needs for 3D buildings:

- Q1: Find the addresses of buildings with height above 30 m
- Q2: Find buildings with the address “Stephansplatz”
- Q3: Find 10 buildings that have the maximum number of roof surfaces
- Q4: Find roof surfaces of buildings over 30 m
- Q5: Find 3D geometries of buildings over 30 m

Queries 6–10 make use of both CityGML data and OSM data. Each case would encompass the possibility of being applied to a practical task involving the retrieval or analysis of real-world geospatial data. A summary of these queries are listed in Table 2.

If a researcher aims to perform a geometric analysis across OSM and CityGML data for different height ranges or building usage types, they might potentially have the following query:

- Q6: Find CityGML ground surfaces and OSM building polygons for all residential buildings.

For tourists who are seeking a hotel with a superior city view, they might inquire:

- Q7: Find hotels over 30 m high

In the context of emergency evacuation during a hurricane disaster, the inquiry might be posed as:

- Q8: Find residential buildings over 30 m high.

Various roof types such as gabled roofs and hip roofs possess the potential for installing

photovoltaic panels. In the studied dataset, the roof type codes in CityGML follow the German cadastre information ALKIS codelists for CityGML 2.0 (see Appendix B for the details). This query is framed as:

- Q9: Find residential buildings with non-flat roofs.

In the context of urban renewal, individuals could inquire about the structures that could be affected and the conceivable expense or workload that might be earmarked for demolition:

- Q10: Find buildings along a certain road within 20 m and calculate the total affected area in m².

5.2.2. Results

All of the ten queries are successfully formulated via the SPARQL query language. Below we provide the respective scripts for queries 9 and 10, while the remaining queries for our experiments can be found in Appendix A. The prefixes utilized are listed in Table 3 and refer to the base namespaces of the CityGML and LGD ontologies, where the remaining prefixes reference authoritative vocabularies such as RDFS and GeoSPARQL.

Query 9 uses CityGML roof type codes in conjunction with their respective labels derived from ALKIS definitions and translated into English (see Table S1).

Query 10 utilizes two GeoSPARQL functions `geof:buffer` and `geof:sfIntersects` to both buffer and intersect geometries. It provides an example of how powerful geospatial functions can also be applied to linked data.

5.3. Performance test

In some application scenarios, e.g. disaster management, expressiveness is not the sole aspect that matters, query execution time is also critical. Given our dual virtual and materialized KG setup, we assess whether our queries can be executed within a reasonable time as well as how the KG setting might impact these results. The quantitative measures we analyze are database storage and query response time.

Table 2. Summary of the features used in Q6–Q10.

	CityGML	OSM	Filter
Q6	Building Geometry	Residential Building	Building Height
Q7	Building, Building Height	Hotel	Building Height
Q8	Building, Building Height	Residential Building	Building Height
Q9	Building, Roof Type	Residential Building	ALKIS RoofType
Q10	Building	Highway	Buffer and Intersection

Table 3. List of prefixes used for SPARQL queries.

Prefix	IRI Namespace
:	https://github.com/yuzzfeng/D2G2/citygml#
bldg:	http://www.opengis.net/citygml/building/2.0/
geo:	http://www.opengis.net/ont/geosparql#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
lgdo:	http://linkedgeodata.org/ontology/

5.3.1. Database size

Although evaluating the storage requirements of each MKG and VKG solution was not a primary goal of our analysis, it can provide a useful indicator of scalability. The CityGML and OSM data took up 501 M of storage in PostgreSQL, which includes not only the data but also geospatial indices. Ontop acting as a lightweight layer does not require additional storage, whereas Apache Jena and GraphDB store materialized triples generated by Ontop. Both Apache Jena and GraphDB use a 1.1 G Turtle file of RDF triples to store the materialized triples. Moreover, for both these MKG solutions, we cannot create a geospatial index because an index cannot be added to a polyhedral surface or geometry collection respectively, rendering this figure a lower bound.

5.3.2. Query response time

The results in terms of query response time in seconds are provided in Table 4 and visualized in Figure 14. Each query is run three times and the average result is recorded. Most of the queries can be evaluated by all the systems. All of the queries can be executed in under 10 seconds, with only one query exceeding 3 seconds. Given the relatively quick response time we

deem overall performance satisfactory for a subjective domain practitioner for all queries.

Due to the limitation of supporting non-simple features such as polyhedral surfaces (by Apache Jena) and geometry collections (by GraphDB), only Ontop is able to handle all the queries, including Q10, thanks to its more mature backend PostGIS. While there are variations across individual queries (Q1–Q9), performance is relatively similar for both the VKG and MKG solutions, no system outperforms across most queries. RDF stores tend to outperform Ontop for integrated queries (Q6–Q9) mostly due to the large number of UNION clauses needed to assemble potential matching OSM data for multiple tag types i.e. node, way, relation and points of interest (e.g. ResidentialBuilding and House). For these large integrated scenarios, GraphDB normally performs better compared to Apache Jena.

5.4. Qualitative comparison with pure relational databases

In this section, we provide a qualitative comparison between SPARQL queries over the KGs, and equivalent SQL queries over relational database storing the original data. We first observe that the query reformulation time by Ontop is typically less than 10 ms, and it is negligible in the total query response time. Therefore, the performance of Ontop vs other materialized KG systems (as in Table 4) can be regarded as the performance of plain relational database systems vs conventional KG systems. In general, since KGs represent a higher level of abstraction with the terminology used in the domain, it is easier to formulate queries in SPARQL, and the resulting SPARQL queries are more understandable. Generating simple queries which rely solely on CityGML would be

Table 4. Query response time.

Query	Ontop	Jena	GraphDB
Q1	0.411	0.235	0.3
Q2	0.109	0.085	0.2
Q3	0.178	1.285	0.5
Q4	0.375	0.259	0.2
Q5	0.209	0.278	0.4
Q6	0.521	0.943	0.3
Q7	0.592	0.441	0.1
Q8	2.125	1.533	0.1
Q9	2.259	2.007	0.6
Q10	9.886	NA	NA

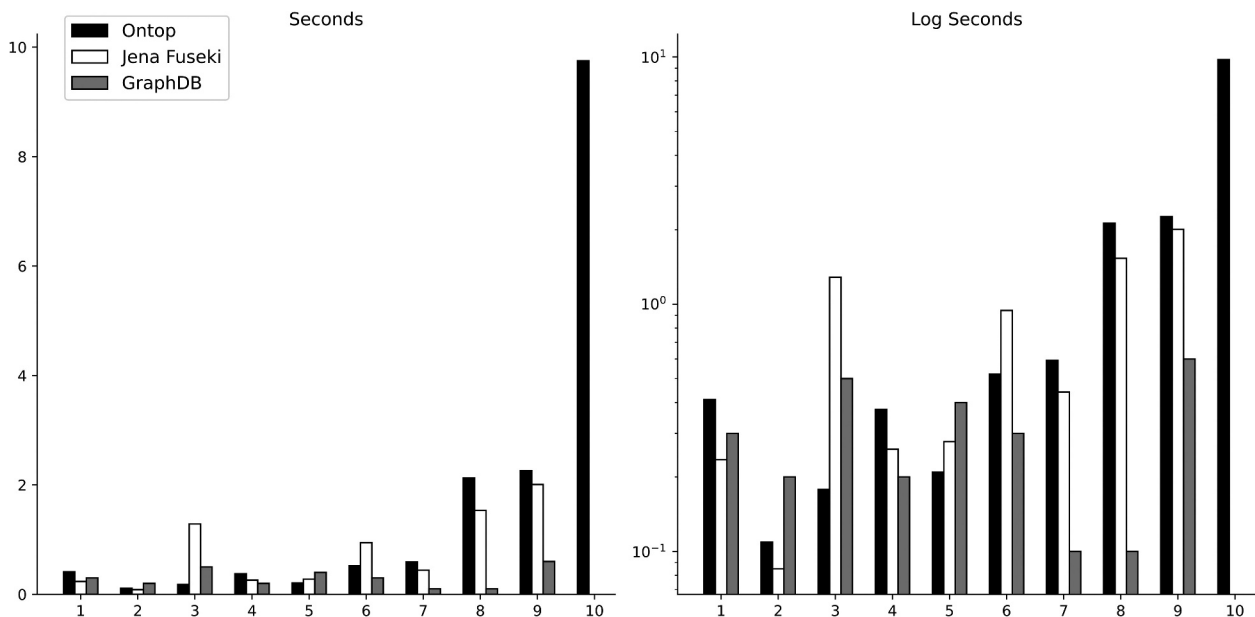


Figure 14. Query response time.

comparable in both SQL and SPARQL. For example, a user can run a query retrieving building addresses by simply joining two tables from the SQL schema. However, the task becomes considerably more difficult when an additional data source such as OSM is integrated. For example, we provide the SQL translation of Q10 formulated previously in Figure 15. We note that this SQL query is automatically generated by Ontop (slightly simplified for readability), but this would be rather close to what a human expert could produce. It would be extremely difficult and laborious for a human user to construct such a complex query.

6. Conclusions and future work

6.1. Conclusions

This paper presents a comprehensive framework to analyze 3D City data via KGs. It provides a methodology to integrate CityGML data with other geospatial data sources and utilize the resulting KG to answer user queries. The experiments confirm that the expressive queries can be formulated over the KGs, and can be efficiently evaluated with state-of-the-art KG systems.

6.2. Future work

Although the obtained results are promising, some limitations encountered while running the experiments, and possible future improvements are discussed below.

- (1) *Support for Complex Geometries.* Many KG systems have limitations with respect to the handling of more complex geometries such as

polyhedral surfaces and geometry collections, which could not be parsed correctly and missed support of respective geospatial index. Specifically, this meant that our MKG approach (GraphDB and Apache Jena) suffered with respect to the execution of Q10 that involves computation with complex geometries. Instead, the VKG system Ontop can leverage mature and well-established relational spatial databases such as PostGIS, and thus avoided such issues.

- (2) *CityGML 3.0.* The CityGML ontology in this work supports only the CityGML specification up to version 2. The latest version of CityGML, version 3, not only introduces new concepts such as time-dependent features but also revises the existing specification, e.g. dropping LoD4 (Kutzner, Chaturvedi, and Kolbe 2020). There is a community effort to build an ontology for CityGML 3.0 by LIRIS, CNRS. However, the version of 3DCityDB system for CityGML 3.0 is still under development at the time of writing. It is expected that the number of relational tables in the new 3DCityDB schema will be substantially reduced. In order to support CityGML 3.0 in our framework, the VKG mapping also need to be revised with respect to the new ontology and 3DCityDB schema.

- (3) *CityGML heterogeneity.* The CityGML data produced by and with the specifications of the government of Bavaria, Germany was used for this analysis. However, during the study, we found that different countries might have different standards for encoding their CityGML data which gives rise to issues such as SRID differences. For example, Estonia

```
SELECT ST_ASTEXT(ST_TRANSFORM(v8."geometrylm27",4326)) AS "v1",
ST_ASTEXT(ST_TRANSFORM(v8."geometrylm60",4326)) AS "v4"
FROM (SELECT DISTINCT v1."cityobject_id" AS "cityobject_idlm12",
v1."geometry" AS "geometrylm27", v2."geometry" AS "geometrylm60",
v1."id" AS "idlm11", v2."id" AS "idlm44", v3."osm_id" AS "osm_idlm451",
ST_ASTEXT(v4."geom") AS "v0", CAST(v5."building_id" AS TEXT) AS "v2",
CAST(v5."cityobject_id" AS TEXT) AS "v3"
FROM "surface_geometry" v1, "surface_geometry" v2, "public"."classes" v3,
"public"."classes" v4,
(SELECT v1."building_id" AS "building_id", v2."cityobject_id" AS "cityobject_id"
FROM "thematic_surface" v1 LEFT JOIN "surface_geometry" v2
ON v1."lod2_multi_surface_id" = v2."root_id") v5,
(SELECT v1."id" AS "id"
FROM "cityobject" v1 LEFT JOIN "objectclass" v2 ON v1."objectclass_id"=v2."id"
WHERE v2."classname"='BuildingGroundSurface') v6
WHERE (ST_INTERSECTS(ST_BUFFER(CAST(ST_ASTEXT(v4."geom") AS GEOGRAPHY),'20'),
CAST(ST_TRANSFORM(v1."geometry",4326) AS GEOGRAPHY))
AND v5."cityobject_id" = v1."cityobject_id"
AND v5."cityobject_id" = v6."id"
AND v1."cityobject_id" = v2."cityobject_id"
AND v3."osm_id" = v4."osm_id"
AND ('W' = v3."osm_type" AND 'SecondaryHighway' = v3."class")
AND 'W' = v4."osm_type")
) v8
```

Figure 15. SQL translation of Q10.

provides a geometry element with its address, and it links each building not to the corresponding 3D solid but to individual surface geometries (failing to provide any solid geometries). Hence, designing VKG mappings to link the ontology with the same 3DCityDB physical storage for analyzes across countries is not necessarily robust. The mapping should be tested for robustness by repeating these experiments with datasets from as many countries as possible to strive to reach a unified design.

- (4) *CityGML data paucity*. Although a significant degree of expressiveness was successfully tested by leveraging the data on CityGML version 2 buildings, there is data paucity for both higher levels of detail such as LoD3, and other non-building items such as vegetation, waterbodies, bridges, etc. The lack of this data makes a significant portion of the 3DCityDB SQL schema redundant for almost all publicly available CityGML datasets. Moreover, CityGML data at LoD3 includes building parts, which could make the integration with OSM more challenging, but we are not able to assess this situation in this work due to the lack of real LoD3 CityGML data.
- (5) *Integrating further data sources*. Our paradigm and evaluation sought to measure CityGML and OSM data integration. While OSM is a popular source of geospatial data, our analysis might not be generalized to other geospatial domains. Expressiveness and the respective overall performance might diminish with the introduction of additional types and combinations of geospatial data. Tasks such as the complexity of matching objects might correspondingly become more complex and have an impact on both ontology integration and query design. These possible risks can only be addressed through further experimental research.

Notes

1. <http://geonames.org/>.
2. <https://www.ogc.org/standard/citygml/>.
3. <https://www.w3.org/>.
4. <https://jena.apache.org/>.
5. <https://graphdb.ontotext.com/>.
6. <https://www.stardog.com/>.
7. <https://cui.unige.ch/isi/ke/ontologies>.
8. <https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/>.
9. <https://w3c-lbd-cg.github.io/bot/>.
10. Note that BPMN is a conceptual modeling language adopted to represent tasks and procedures within a system. To get more information about BPMN, the authors refer the readers to White (2004).

11. For more information about implementation issues please refer to Section 4.
12. <https://geodaten.bayern.de/opengeodata/OpenDataDetail.html?pn=lod2>.
13. <https://github.com/3dcitydb/3dcitydb>.
14. <https://github.com/3dcitydb/importer-exporter>.
15. <https://cui.unige.ch/isi/ke/ontologies>.
16. <https://www.ogc.org/standard/gml/>.
17. https://opengeospatial.github.io/ogc-geosparql/geosparql11/sf_geometries.html.
18. <https://www.w3.org/TR/vocab-ssn/>.
19. <https://www.w3.org/2004/02/skos/intro>.
20. <https://www.dublincore.org/specifications/dublin-core/>.
21. <https://doi.org/10.5281/zenodo.10276433>.
22. <https://ontop-vkg.org/>.
23. <https://jena.apache.org/index.html>.
24. <https://jena.apache.org/documentation/fuseki2/index.html>.
25. <https://graphdb.ontotext.com/>.
26. <https://geodaten.bayern.de/opengeodata/OpenDataDetail.html?pn=lod2>.
27. ALKIS stands for the Authoritative Real Estate Cadastre Information System in Germany. <https://www.adv-online.de/Products/Real-Estate-Cadastre/ALKIS/>.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This research is supported by the German Research Foundation (DFG) and the Autonomous Province of Bolzano-Bozen through its joint project “Dense and Deep Geographic Virtual Knowledge Graphs for Visual Analysis - D2G2” [grant number 500249124].

Notes on contributors

Linfang Ding is an Associate Professor in Geomatics at the Department of Civil and Environmental Engineering, Norwegian University of Science and Technology (NTNU). Her current research interests include geospatial knowledge graphs, geovisual analytics, mobility analysis, and 3D city modeling.

Guohui Xiao is an Associate Professor at the Department of Information Science and Media Studies, University of Bergen, Norway. His research interests include knowledge graphs, Semantic Web, ontologies, data integration, and automated reasoning.

Albulen Pano is a Research Assistant at the Research Centre for Knowledge and Data (KRDB), Free University of Bozen-Bolzano (Italy). His research interests include virtual knowledge graphs, geospatial data integration, and geospatial knowledge graphs.

Mattia Fumagalli is a Fixed-term Assistant Professor at the Research Centre for Knowledge and Data (KRDB), Free University of Bozen-Bolzano (Italy). His research concerns primarily Artificial Intelligence, with a particular focus on

the automated support for Knowledge Representation and Conceptual Modeling.

Dongsheng Chen is a PhD candidate at the Chair of Cartography and Visual Analytics, Technical University of Munich, Germany. His research interests are Urban morphology, Knowledge graph, and GeoAI.










Yu Feng is a Postdoc researcher at the Chair of Cartography and Visual Analytics, Technical University of Munich, Germany. His research interests are volunteered geographic information, cartographic generalization, and geospatial artificial intelligence.

Diego Calvanese is a Professor at the Research Centre for Knowledge and Data (KRDB), Free University of Bozen-Bolzano (Italy), and Wallenberg Guest Professor in AI for Data Management at Umeå University (Sweden). His research interests include knowledge representation and reasoning, virtual knowledge graphs, ontology languages, description logics, conceptual data modeling and data integration.

Hongchao Fan is Professor in Geomatics at the Department of Civil and Environmental Engineering, Norwegian University of Science and Technology (NTNU). His research interests include 3D city modeling, spatial data mining from VGI data and laser scanning.

Liqiu Meng is a Professor of Cartography and Visual Analytics at the Technical University of Munich. Her research interests include geodata integration, mobile map services, multimodal navigation algorithms, geovisual analytics, and ethical concerns in social sensing.

ORCID

Linfang Ding  <http://orcid.org/0000-0002-3707-5845>
 Guohui Xiao  <http://orcid.org/0000-0002-5115-4769>
 Albulen Pano  <http://orcid.org/0000-0002-0905-9004>
 Mattia Fumagalli  <http://orcid.org/0000-0003-3385-4769>
 Dongsheng Chen  <http://orcid.org/0000-0002-4486-0462>
 Yu Feng  <http://orcid.org/0000-0001-5110-5564>
 Diego Calvanese  <http://orcid.org/0000-0001-5174-9693>
 Hongchao Fan  <http://orcid.org/0000-0002-0051-7451>
 Liqiu Meng  <http://orcid.org/0000-0001-8787-3418>

Data availability statement

The data that support the findings of this study are openly available in zenodo at <https://doi.org/10.5281/zenodo.10276433>.

References

- Ahmadian, S., and P. Pahlavani. 2022. "Semantic Integration of OpenStreetmap and CityGML with Formal Concept Analysis." *Transactions in GIS* 26 (8): 3349–3373. <https://doi.org/10.1111/tgis.13006>.
- Atila, U., I. R. Karas, and A. Abdul-Rahman. 2013. "Integration of CityGML and Oracle Spatial for Implementing 3D Network Analysis Solutions and Routing Simulation within 3D-GIS Environment." *Geo-Spatial Information Science* 16 (4): 221–237. <https://doi.org/10.1080/10095020.2013.867102>.
- Baig, S. U., and A. Abdul-Rahman. 2013. "Generalization of Buildings within the Framework of CityGML." *Geo-Spatial Information Science* 16 (4): 247–255. <https://doi.org/10.1080/10095020.2013.866617>.
- Chadzynski, A., N. Krdzavac, F. Farazi, M. Q. Lim, S. Li, A. Grisiute, P. Herthogs, A. von Richthofen, S. Cairns, and M. Kraft. 2021. "Semantic 3D City Database – An Enabler for a Dynamic Geospatial Knowledge Graph." *Energy and AI* 6:100106. <https://doi.org/10.1016/j.egyai.2021.100106>.
- Das, S., S. Sundara, and R. Cyganiak. 2012. "R2RML: RDB to RDF Mapping Language." W3C Recommendation. W3C.
- Ding, L., G. Xiao, D. Calvanese, and L. Meng. 2020. "A Framework Uniting Ontology-Based Geodata Integration and Geovisual Analytics." *ISPRS International Journal of Geo-Information* 9 (8): 474. <https://doi.org/10.3390/ijgi9080474>.
- Ding, L., G. Xiao, D. Calvanese, and L. Meng. 2021. "Consistency Assessment for Open Geodata Integration: An Ontology-Based Approach." *Geoinformatica* 25 (4): 733–758. <https://doi.org/10.1007/s10707-019-00384-9>.
- Ding, L., G. Xiao, A. Pano, H. Fan, D. Calvanese, and L. Meng. 2023. "Querying CityGML Data Through Virtual Knowledge Graphs." *Abstracts of the International Cartographic Association* 6:53. <https://doi.org/10.5194/ica-abs-6-53-2023>.
- Ding, L., G. Xiao, A. Pano, C. Stadler, and D. Calvanese. 2021. "Towards the Next Generation of the LinkedGeodata Project Using Virtual Knowledge Graphs." *Journal of Web Semantics* 71:100662. <https://doi.org/10.1016/J.WEBSEM.2021.100662>.
- Fan, H., A. Zipf, Q. Fu, and P. Neis. 2014. "Quality Assessment for Building Footprints Data on OpenStreetmap." *International Journal of Geographical Information Science* 28 (4): 700–719. <https://doi.org/10.1080/13658816.2013.867495>.
- Gröger, G., T. H. Kolbe, C. Nagel, and K.-H. Häfele. 2012. "OGC City Geography Markup Language (CityGML) Encoding Standard." Open Geospatial Consortium.
- Haklay, M., and P. Weber. 2008. "Openstreetmap: User-Generated Street Maps." *IEEE Pervasive Computing* 7 (4): 12–18. <https://doi.org/10.1109/MPRV.2008.80>.
- Hitzler, P., M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph. 2009. "OWL 2 Web Ontology Language Primer." Technical Report. W3C. <http://www.w3.org/TR/owl2-primer/>.
- Hogan, A., E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutiérrez, S. Kirrane, et al. 2021. "Knowledge Graphs." *ACM Computing Surveys* 54 (4): 1–37. <https://doi.org/10.1145/3447772>.
- Huang, W., P.-O. Olsson, J. Kanters, and L. Harrie. 2020. "Reconciling City Models with BIM in Knowledge Graphs: A Feasibility Study of Data Integration for Solar Energy Simulation." *ISPRS Annals of the Photogrammetry, Remote Sensing & Spatial Information Sciences* VI-4/W1-2020:93–99. <https://doi.org/10.5194/isprs-annals-VI-4-W1-2020-93-2020>.
- Janowicz, K., Y. Hu, G. McKenzie, S. Gao, B. Regalia, G. Mai, R. Zhu, B. Adams, and K. Taylor. 2016. "Moon Landing or Safari? A Study of Systematic Errors and Their Causes in Geographic Linked Data." In *Geographic Information Science*, edited by J. A. Miller, D. O'Sullivan, and N. Wiegand, 275–290. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-45738-3_18.

- Jovanovik, M., T. Homburg, and M. Spasic. 2021. "A GeoSPARQL Compliance Benchmark." *ISPRS International Journal of Geo-Information* 10 (7): 487. <https://doi.org/10.3390/ijgi10070487>.
- Koch, S., and M.-O. Löwner. 2017. *Representation of CityGML Instance Models in BaseX*, 63–78. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-25691-7_4.
- Kolbe, T. H. 2009. "Representing and Exchanging 3D City Models with CityGML." In *3D Geo-Information Sciences*, 15–31. Springer. https://doi.org/10.1007/978-3-540-87395-2_2.
- Kolbe, T. H., T. Kutzner, C. S. Smyth, C. Nagel, C. Roensdorf, and C. Heazel. 2021. *OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard*. OGC standard. OGC.
- Kutzner, T., K. Chaturvedi, and T. H. Kolbe. 2020. "CityGML 3.0: New Functions Open Up New Applications." *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 88 (1): 43–61. <https://doi.org/10.1007/s41064-020-00095-z>.
- Kyzirakos, K., D. Savva, I. Vlachopoulos, A. Vasileiou, N. Karalis, M. Koubarakis, and S. Manegold. 2018. "GeoTriples: Transforming Geospatial Data into RDF Graphs Using R2RML and RML Mappings." *Journal of Web Semantics* 52–53:16–32. <https://doi.org/10.1016/j.websem.2018.08.003>.
- Ledoux, H., K. A. Ohori, K. Kumar, B. Dukai, A. Labetski, and S. Vitalis. 2019. "CityJSON: A Compact and Easy-To-Use Encoding of the CityGML Data Model." *Open Geospatial Data, Software and Standards* 4 (1): 1–12. <https://doi.org/10.1186/s40965-019-0064-0>.
- Lehmann, J., R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, et al. 2015. "DBpedia - a Large-Scale, Multilingual Knowledge Base Extracted from Wikipedia." *Semantic Web Journal* 6 (2): 167–195. <https://doi.org/10.3233/SW-140134>.
- Liu, L., Z. Fu, Y. Xia, H. Lin, X. Ding, and K. Liao. 2023. "A Building Polygonal Object Matching Method Based on Minimum Bounding Rectangle Combinatorial Optimization and Relaxation Labeling." *Transactions in GIS* 27 (2): 541–563. <https://doi.org/10.1111/tgis.13039>.
- Li, W., S. Wang, S. Wu, Z. Gu, and Y. Tian. 2022. "Performance Benchmark on Semantic Web Repositories for Spatially Explicit Knowledge Graph Applications." *Computers, Environment and Urban Systems* 98:101884. <https://doi.org/10.1016/j.compenvurbsys.2022.101884>.
- Manola, F., and E. Miller. 2004. "RDF Primer." W3C Recommendation. W3C. <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>.
- Noardo, F., C. Ellul, L. Harrie, I. Overland, M. Shariat, K. A. Ohori, and J. Stoter. 2019. "Opportunities and Challenges for GeoBIM in Europe: Developing a Building Permits Use-Case to Raise Awareness and Examine Technical Interoperability Challenges." *Journal of Spatial Science*. <https://doi.org/10.1449/8596.2019.162725314498596.2019.1627253>.
- Open Geospatial Consortium. 2023. "OGC GeoSPARQL Specification." <https://opengeospatial.github.io/ogc-geosparql/geosparql11/index.html>. Online.
- Schade, S., and P. Smits. 2012. "Why Linked Data Should Not Lead to Next Generation SDI." In *2012 IEEE International Geoscience and Remote Sensing Symposium*, 2894–2897. New York City, US: IEEE. <https://doi.org/10.1109/IGARSS.2012.6350721>.
- Stadler, C., J. Lehmann, K. Höffner, and S. Auer. 2012. "LinkedGeodata: A Core for a Web of Spatial Open Data." *Semantic Web Journal* 3 (4): 333–354. <https://doi.org/10.3233/SW-2011-0052>.
- Vrandečić, D., and M. Krötzsch. 2014. "Wikidata: A Free Collaborative Knowledgebase." *Communications of the ACM* 57 (10): 78–85. <https://doi.org/10.1145/2629489>.
- Wang, X., and M. Xie. 2022. "Integration of 3D GIS and BIM and Its Application in Visual Detection of Concealed Facilities." *Geo-Spatial Information Science* 27 (1): 132–141. <https://doi.org/10.1080/10095020.2022.2054732>.
- White, S. A. 2004. "Introduction to BPMN." *IBM Cooperation*, 2.
- Xiao, G., D. Lanti, R. Kontchakov, S. Komla-Ebri, E. Güzel-Kalayci, L. Ding, J. Corman, B. Cogrel, D. Calvanese, and E. Botoeva. 2020. "The Virtual Knowledge Graph System Ontop." In *International Semantic Web Conference*, 259–277. Cham, Switzerland: Springer. https://doi.org/10.1007/978-3-030-62466-8_17.
- Yao, Z., C. Nagel, F. Kunde, G. Hudra, P. Willkomm, A. Donaubaue, T. Adolphi, and T. H. Kolbe. 2018. "3DCityDB—A 3D Geodatabase Solution for the Management, Analysis, and Visualization of Semantic 3D City Models Based on CityGML." *Open Geospatial Data, Software and Standards* 3 (1): 1–26. <https://doi.org/10.1186/s40965-018-0046-7>.