




Article

# An Approach of Automatic SPARQL Generation for BIM Data Extraction

Dongming Guo \*, Erling Onstein  and Angela Daniela La Rosa 

Department of Manufacturing and Civil Engineering, Norwegian University of Science and Technology, 2802 Gjøvik, Norway; erling.onstein@ntnu.no (E.O.); angela.d.l.rosa@ntnu.no (A.D.L.R.)

\* Correspondence: dongming.guo@ntnu.no

Received: 13 November 2020; Accepted: 7 December 2020; Published: 9 December 2020



**Abstract:** Generally, building information modelling (BIM) models contain multiple dimensions of building information, including building design data, construction information, and maintenance-related contents, which are related with different engineering stakeholders. Efficient extraction of BIM data is a necessary and vital step for various data analyses and applications, especially in large-scale BIM projects. In order to extract BIM data, multiple query languages have been developed. However, the use of these query languages for data extraction usually requires that engineers have good programming skills, flexibly master query language(s), and fully understand the Industry Foundation Classes (IFC) express schema or the ontology expression of the IFC schema (ifcOWL). These limitations have virtually increased the difficulties of using query language(s) and raised the requirements on engineers' essential knowledge reserves in data extraction. In this paper, we develop a simple method for automatic SPARQL (SPARQL Protocol and RDF Query Language) query generation to implement effective data extraction. Based on the users' data requirements, we match users' requirements with ifcOWL ontology concepts or instances, search the connected relationships among query keywords based on semantic BIM data, and generate the user-desired SPARQL query. We demonstrate through several case studies that our approach is effective and the generated SPARQL queries are accurate.

**Keywords:** building information modelling (BIM); ifcOWL; data extraction; semantic; SPARQL generation

## 1. Introduction

Building information modelling (BIM) is a general digital building representation and information processing platform, and it has been widely used in the architecture, engineering, and construction (AEC) industry [1–4]. Currently, BIM interoperability research and applications have covered the environmental [1,2,5,6], economic [7,8], and social domains [9] and their combinations [1,10–12]. For the sharing and exchange of BIM data by various participants among different software applications, the Industry Foundation Classes (IFC), an open international standard for BIM data, is being continuously developed and updated [13,14]. Furthermore, the IFC standard aims to cover the entire AEC industry, and the IFC schema has already become a common data expression schema for presenting BIM instances and data [15–17].

BIM data generally stand as a whole project's digitized description and data repository. BIM data include multiple dimensions of building information, such as building element relationships and properties, building structure, building space and geometry information, building plumbing, and so on. Building models can be applied in different practical engineering tasks according to specific case requirements. Different stakeholders desire requisite and sufficient building information from BIM data for vendor-specific business processes. Correctly querying and extracting the required BIM data from the IFC file(s) is a necessary step for various data analyses and engineering applications. However, the IFC data model was initially designed for the sharing and exchange of product data,

rather than for data query and analysis tasks. The main purpose of IFC is to maximize its ability to represent basic concepts [18] and descriptive building information with relatively complicated structures. All of these have made it difficult to query and manage IFC instance data. Additionally, when integrating and processing multi-source data, the IFC meta model is not flexible enough [19]. Meanwhile, Zhong et al. [20] pointed out that semantic clarity cannot be achieved in current IFC files when mapping entities and relationships, and this could result in invalid data exchange between different applications. Thus, specific information extraction from BIM data based on the IFC schema is still a challenge for different stakeholders and involves a mass of manual and repetitive operations.

Semantic web/ontology technologies can facilitate information sharing, integration, and linkage and improve the collaborative ability of different application systems [21]. They seem to provide an alternative method for solving issues related to IFC limitations. To promote the use of semantic web technology in AEC industry, the BuildingSMART organization standardized an ifcOWL ontology as a domain foundation ontology for the AEC industry domain [22]. Since then, the use of the SPARQL (SPARQL Protocol and RDF Query Language) query language or SWRL (Semantic Web Rule Language) for building data extraction has become a popular method [19,23–25]. In this kind of method, SPARQL and SWRL queries are manually written by stakeholders, which requires stakeholders to have good programming skills and fully understand the grammar/elements of the SPARQL (or SWRL) language and ifcOWL schema. This raises the bar for SPARQL usage in AEC industry and limits automated data processing to a certain extent. To solve this issue, we propose a simple approach to automatically generate the SPARQL queries desired by engineers based on some simple search keywords for BIM data extraction. Utilizing the path query function provided in the Stardog RDF database management system, we search the shortest path that connects all query keywords in a BIM instance and extract the structure of the shortest path to generate the SPARQL query. This method can reduce the requirements of programming skills and knowledge reserves of AEC engineers when using SPARQL to extract BIM data. Additionally, our method is based on BIM cases, so it can also be applied in different IFC versions or semantically enriched BIM models.

Section 2 briefly introduces the related work about BIM data extraction. We also discuss the potential contributions of our method in different application domains. The main research approach and implementation procedures of our method are introduced in Section 3. Section 4 verifies our approach in two case studies. Finally, we draw conclusions and discuss the benefits and limitations of this method in Section 5.

## 2. Related Work

This section includes two research fields. The first one focuses on data extraction approaches based on different data models. The second one concerns the relevant research on semantic BIM data extraction.

### 2.1. Data Query Approaches Based on Different Data Models

To retrieve the subsets of BIM data according to different specific application scenarios, some researchers have proposed concise, well-defined formal query languages to specify the required information. These specific query languages can provide some effective direct access to the required data from the BIM model. For example, the Building Information Model Query Language (BIMQL) is a query language tailored to IFC building models [26] that is mainly used to retrieve instances and attribute values; however, it lacks the expressiveness to extract and match arbitrary subgraphs [27]. To adequately treat geometric information, Daum et al. [28] introduced a topological and spatial querying system using the Query Language for Building Information Models (QL4BIM), in which the Boundary Representation (BRep) of the operands was used, and elements contained or intersecting within other elements can be retrieved.

The other kind of query methods convert IFC instance models into different formats and use some certain query languages based on specific formats. Ismail et al. [29] developed an approach to convert

and store BIM models in a Neo4j graph database and then use the Cypher query language, specified in the Neo4j database, to query data. Krijnen et al. [27] proposed a novel, binary equivalent format with traditional IFC that also provides more compact storage and less overhead than graph serialization in the Resource Description Framework (RDF), and a SPARQL query engine can be implemented in this format for data extraction.

Additionally, Nepal et al. [30] explored a process and methods for extracting and querying construction-specific information from a BIM, in which custom 2D topological XQuery predicates were created to implement a variety of spatial queries.

## 2.2. Semantic BIM Data Extraction

Some BIM models come from BIM design software; however, this kind of software cannot store data in the RDF data format. Thus, converting the EXPRESS schema (used in IFC) into the ifcOWL schema became a required step for the semantic processing of BIM data. BuildingSMART provides some tools to implement this: the buildingSMART IfcDoc tool (<https://github.com/buildingSMART/IfcDoc>) the UGent—Aalto IFC-to-RDF converter (<https://github.com/pipauwel/IFCtoRDF>), and the Walter Terkaj EXPRESS-to-OWL converter (<http://www.terkaj.com/tools/ExpressToOwl/ExpressToOwl.zip>). Certainly, there have been some other efforts for this kind of conversion [22,31–33]. These tools and research have accelerated the application and development of semantic web technology in AEC industry [20,24,34].

Nepal et al. [35] introduced a new approach that included ontology-based feature modeling, automatic feature extraction (using XQuery query predicates), and query formulation and processing (using a feature-based model and formal query specifications) to quickly extract the construction-specific information from a given BIM model. Zhang et al. [36] designed an approach of ontology-based partial BIM model extraction, in which a partial model extraction algorithm was achieved through Java.

More commonly, semantic BIM data are generally extracted through SPARQL or SWRL query codes. For example, a shared ontology approach was developed to improve the finding and integration of building information distributed in different knowledge bases, while SPARQL was used to search for information from knowledge bases [37]. Beetz et al. [38] applied RDF(S) sub-graph retrieval for a partial building view extraction and generated a graph pattern matching the query with SPARQL. de Farias et al. [25] utilized SWRL to extract building data from semantic BIM data to create building model views.

However, due to the complexity of the IFC schema, many queries and analysis tasks are still laborious when using SPARQL in AEC industry, and many required properties and relationships, such as product geometry quantities, spatial and topological relations, etc., are difficult to retrieve [19]. To solve these issues, Zhang et al. [19] developed SPARQL extensions for querying ifcOWL building data, called BimSPARQL. In BimSPARQL, some new RDF vocabularies that can be used in SPARQL queries were designed through a set of extension functions, and a module for geometry-related functions was implemented to derive implicit BIM information [19]. Additionally, Liu et al. [39] proposed an ontology-based semantic approach to extract construction-oriented quantity take-off (QTO) information for the purpose of construction operation planning and to allow the use of the building product ontology formalized from construction perspectives to semantically query (SPARQL query) the needed QTO data in the BIM model.

As a method of data extraction, the SPARQL query language is also usually applied to the quality and regulatory checking of BIM models [40]. For example, a system based on the Semantics of Business Vocabulary and Business Rules (SBVR) and SPARQL was proposed to formalize the regulations [41]. This kind of regulation-checking application adopts SPARQL query programs as the representation of different regulations, and then executes these SPARQL codes on the semantic BIM model to implement regulations checking. This process is similar to BIM data extraction. The SPARQL filter conditions of data extraction can be viewed as detailed regulations in rule compliance checking. The query results can be either rule-compliant or rule-incompliant BIM data, depending on the SPARQL programming.

That is, rule compliance checking and data extraction of BIM have the same processing flow, and the differences are the origin and contents of the query filter conditions. For example, an ontology-based framework was proposed to carry out environmental monitoring and compliance checking cross BIM and different information systems, in which the regulation clauses were transformed into SPARQL codes [42]. Additionally, a semantic BIM Reasoner (SBIM-Reasoner) was developed for IFC model semantic validation, in which semantic technologies are used to build the semantic repository from the input IFC model, and in the same way, validation regulations were designed as SPARQL queries [43].

In general, although SPARQL has widely been used for data extraction and related research in AEC industry, SPARQL queries are typically generated/programmed manually [19,37–43]. To bridge this gap, we develop herein an approach of automated SPARQL query generation for BIM data extraction. This approach can be utilized in various applications, such as data extraction, rule compliance checking, data retrieval in semantic enrichment of BIM models, and so on. In this paper, we only focus on SPARQL generation according to query keywords provided by users/engineers, so natural language processing (NLP) technologies (e.g., semantic parsing, syntactic structure, and dependency parsing) and the engineering applications of data extraction are beyond the scope of this article.

### 3. Main Research Approach

#### 3.1. Research Approach and Definitions

Some basic principles should be introduced firstly. RDF is a standard model that can facilitate data merging on the Web, and the RDF model is a graph-based data model with a directed and labeled graph data format [44]. SPARQL is a graph-based query language for RDF and can be used to query data across different data sources [45]. A SPARQL query generally consists of two clauses: **SELECT** and **WHERE**. The **SELECT** clause specifies the searched target(s) and the **WHERE** clause is a set of triples, in which each triple can be a query filter condition. In a knowledge base, the assertion box (ABox) and the terminological box (TBox) are generally used to describe two different types of statements: the TBox describes the statements about conceptual entities and the ABox describes the statements involving concrete entities [46]. In order to use semantic building data with semantic web technologies, we need to convert the IFC data into the RDF data format (based on the ifcOWL schema), as described in Section 2.2.

Additionally, we introduce the path concept, which is the connected relationships between two specified nodes, including the linked nodes and properties. The common nodes in different paths are frequently referred to. Figure 1 shows our definitions and assumptions about the length of a path and the shortest path. The length of a path is the number of connected edges in a path. The shortest path connecting the nodes “3” and “4” is the path of minimum length among all paths that include the nodes “3” and “4”, shown in Figure 1c.

A SPARQL query is a sub-graph of the whole data model, so in our approach, we should firstly acquire the structure of the SPARQL query desired in the BIM model, and then utilize the structure of the SPARQL query to generate the required SPARQL query. Considering that not all civil engineers have programming experience, we tried to avoid intricate programming, so we chose the Stardog 7 RDF graph data platform because Stardog 7 provides path queries that can traverse an RDF graph, return all intermediate nodes on each path, and allow arbitrary SPARQL graph patterns to be used in the query [47]. One can easily get a query path in the Stardog database without complex programming. The path query syntax provided in the Stardog RDF database is shown in Figure 2, in which **START** and **END** specify a start node and an end node, and **VIA** assigns a graph pattern to match each edge in the path [47]. According to the query keywords provided by engineers/users, all paths that contain different query keywords can be found through the Stardog 7 database. Based on the found paths, simple programming is developed to find the common nodes in path files and to connect all query keywords through the shortest path. The structure of the shortest path is the structure of the SPARQL

query. After that, the nodes in the shortest path are replaced with SPARQL variables and the final SPARQL query is generated.

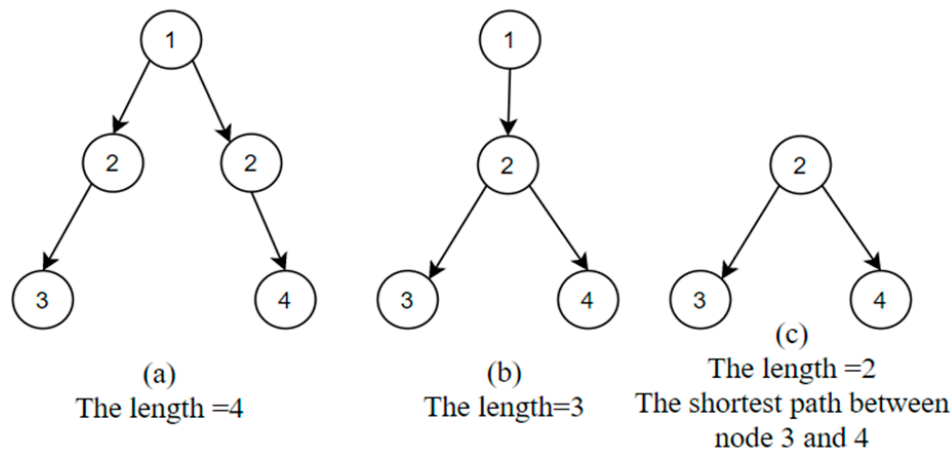


Figure 1. Definitions of the length of paths and the shortest path.

```

PATHS [SHORTEST|ALL] [CYCLIC] [<DATASET>]
START ?s [= <IRI> | <GRAPH PATTERN> ] END ?e [= <IRI> | <GRAPH PATTERN> ] VIA <GRAPH PATTERN> <VAR> | <PATH>
[<MAX LENGTH <int>]
[<OFFSET <int>]
[<LIMIT <int>]
    
```

Figure 2. The path query syntax provided in the Stardog RDF database.

To formalize the query keywords, a query tabulation was designed to record the query requirements and was then used as a query basis to produce the SPARQL query, listed in Table 1. The first line of this tabulation is the query target(s), and the column below a target can be filled with query filter conditions about the target(s). A query tabulation can be filled with several targets. When multiple targets are extracted in a SPARQL query, a semicolon is used to separate targets, and a dot indicates the dependency relation of targets. For example, wall.GlobalUniqueID means the GlobalUniqueID of a wall, shown in Table 1b. Table 1b expresses that the targets are the GlobalUniqueIDs of walls and walls with some query conditions. The number of filter conditions is not limited, and one can increase or reduce the number of query conditions based on different requirements. It should be noted that specific query targets and conditions are necessary and the basis for correct execution results in our approach. It is best to fill in some query keywords according to the naming conventions, such as names of instances/classes/properties in a certain BIM case, because different BIM cases may adopt different instance naming conventions. For example, the “second floor” is marked as “Level 2” in the first case and “Plan E2” in the second case used to verify our approach in this paper. Moreover, only string-matching technology was adopted in this paper, rather than semantic NLP technologies, so, when querying a window, one can fill in “window”, “windows”, “IFCWINDOW”, or “IfcWindow” in the query tabulation as a query target. Based on string matching, the term closest to the query target will be identified as a query keyword for following processing in a BIM case. Due to the lack of semantic NLP technologies, when querying door information, one can fill in “doors”, “ifcdoor”, or “door”, rather than “exit” or “gate” in the query tabulation, although “exit” and “gate” have similar semantics to “door” to some extent in building models.



**Table 1.** The format of query requirements.

<b>(a)</b>	
<b>Target:</b>	<b>Window</b>
Condition:	Level 2
Condition:	
<b>(b)</b>	
<b>Target:</b>	<b>Wall; Wall.GlobalUniqueID</b>
Condition:	wall.Plan E2
Condition:	wall. External
Condition:	wall.TRUE

### 3.2. Implementation of the Proposed Approach

To implement our approach, we follow these operations:

1. Establish the BIM knowledge base. The IFC to RDF tool [48] is used to convert the IFC schema into a semantic BIM model and store the RDF BIM data in the Stardog RDF database.
2. Create the Tbox and Abox based on the BIM knowledge base. Query keywords provided in a query tabulation are matched with concepts or instances in the BIM knowledge base, and the terms that are most similar to query keywords are used for the following processing.

An RDF model is a directed and labeled graph data model, and a SPARQL query is a sub-graph of the data model. In order to generate a SPARQL query, we should firstly capture a data path that connects all query keywords in the semantic BIM data model.

3. Acquire all paths related to the query keywords. We adopted the Stardog path query function that can explore all unidirectional relationships of a query keyword and express the results in data path(s). According to the query keywords, all paths containing the query keywords in the BIM instance model are found and stored in csv files, including all paths that begin with keywords or end with keywords. A path file includes all paths related with a query keyword. In the Stardog RDF database, the "Run to File" function can store path information in a csv file.
4. Extract the structure of the SPARQL query. To generate effective query results and the corresponding SPARQL query, the provided query keywords should have certain connected relationships in a given BIM model. That means that all keywords should be covered in a sub-graph of the BIM model. In other words, some paths that are gained in Step 3 should have some common node(s). So, the searching the common node(s) in path files is a key step in extracting the structure of the SPARQL query.
  - When there are only two path csv files (meaning that there are only two query keywords: one target and one query condition), the search task is simple; each node is iteratively indexed from one path file and checked as to whether it exists in the other path file. Once a common node is checked, this common node and the two sub-paths from the common node to the corresponding keywords are recorded. The common node is viewed as a top node to connect the two sub-paths, and then a path connecting the two query keywords is gained. When all common nodes and connected paths are sought out, the shortest path(s) connected two query keywords can be ascertained and stored in a new csv file, named the shortest-path file. After that, the structure of a SPARQL query in the BIM model can be obtained based on the structure and connected relationships of nodes in the shortest path.
  - When multiple query keywords are provided and multiple path files are generated, the search task should keep to the following search rules and steps:

- I. The target keyword(s) will be the important keyword(s) and its/(their) path file(s) will be viewed as the important search target(s). A target keyword is viewed as a core keyword when it has relationships with several other query keywords in the query tabulation. The path file(s) of the core keyword(s) will be viewed as main target file(s). The path files of related query keywords will be viewed as relevant-path files.
  - II. Utilizing the same search processing method as in the case of only two path files, the nodes in every relevant-path file should be matched with nodes in the main target file, and then the search results are stored in a shortest-path file. After processing all relevant-path files, multiple shortest-path files can be obtained. This is considered the first iteration.
  - III. For dealing with these shortest-path files, we similarly search the common node(s) between two shortest-path files and merge the two paths through the common node(s) into a new path in which the common node(s) should be only recorded once. After that, new paths are stored in a new shortest-path file for the next iteration. In a new iteration, the new shortest-path file and a shortest-path file (generated in the first iteration) are processed and merged into a newer shortest-path file for the next iteration. When all shortest-path files are merged into one file, the iterations are finished. The shortest path in the final file that connects all query keywords is the final result. The structure and connected relationships of nodes in the shortest path is the structure of the desired SPARQL query. It is noted that there can be more than one such shortest path in the final file; however, the structure of the shortest path is generally the same.
5. Generate the SPARQL query. Once the structure of the SPARQL query is acquired, the SPARQL query is created. Every two connected nodes and their relationship property in the shortest path are converted into a query filter condition in the *WHERE* clause of SPARQL. If the nodes of the path may be some given keywords or classes, these values can be kept in the *WHERE* clause of SPARQL. The other endpoint nodes and all intermediate nodes are replaced with SPARQL variables, such as ?a, ?b, or ?c. In the shortest path, the same node uses a uniform variable name and different nodes use different variable names. The variable that replaces the target keyword is used in the *SELECT* clause. If a target keyword is mapped with a class in the ifcOWL schema, the target variable (in the *SELECT* clause) chooses the instance variable or a value variable that is directly linked with this class in the shortest path. Then, the SPARQL query is generated.

The whole processing flow of our approach is illustrated in Figure 3.

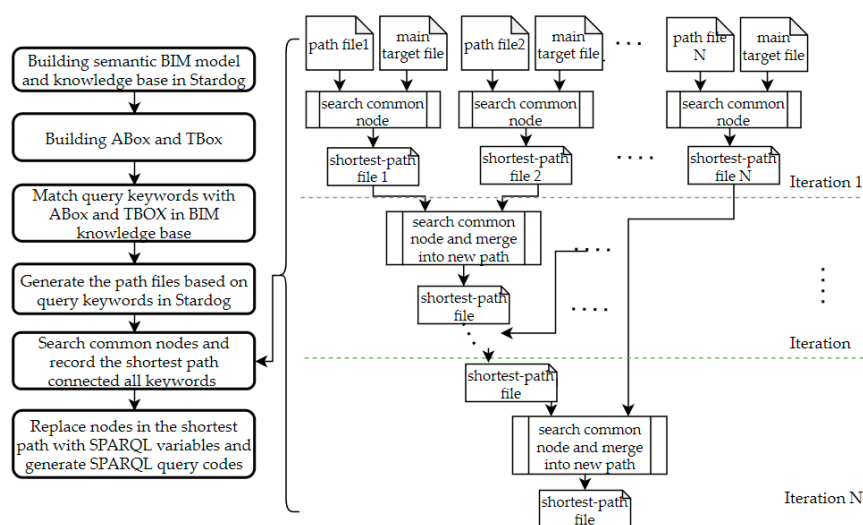


Figure 3. The processing flow of the proposed approach.

#### 4. Case Studies

To validate the feasibility and effectiveness of the proposed approach, we tested it in the following environment:

- Intel processor Xeon(R) E-2176M CPU 2.7 GHz, SSD 1024 GB, and 32 GB RAM memory;
- Microsoft 64-bit Windows 10 Operating System;
- Stardog triple store and API version 7;
- A wrapper library called pystardog (a Python virtualenv).

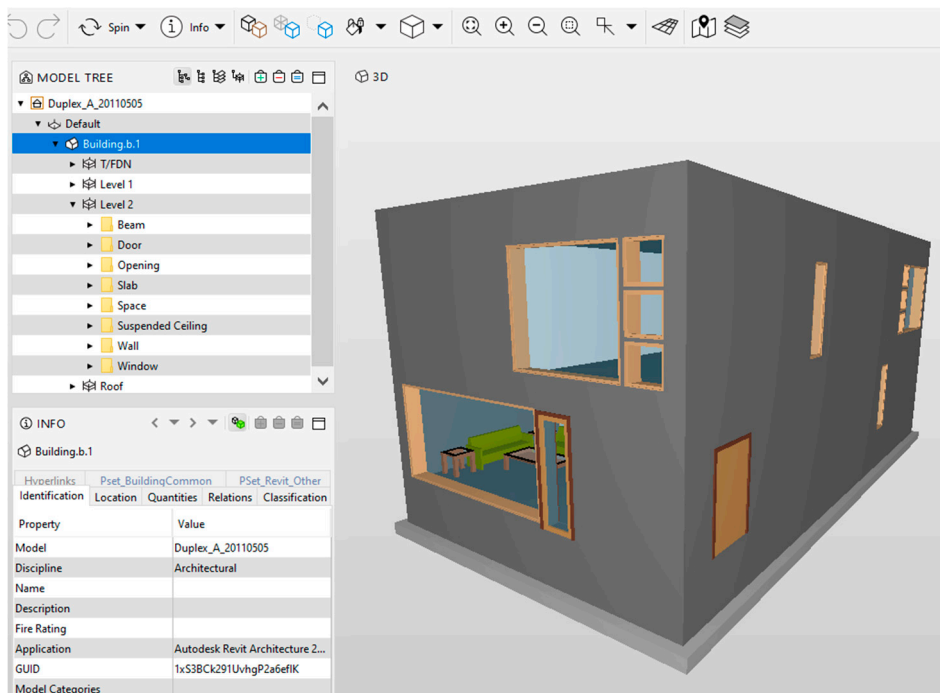
For case studies, we used two different cases to verify that our approach is simple, effective, and accurate.

##### 4.1. Case Study One: A Duplex Apartment Case

This is a public case with IFC file and equivalent RDF file (.ttl) available online [49], shown in Figure 4 in the Solibri Model Viewer tool [50]. In this case, we used the case prefixes in the .ttl file as our SPARQL prefixes. Table 2 lists the namespace Internationalized Resource Identifier (IRI) and the related prefixes used in this case.

**Table 2.** The prefixes in this case.

Prefix	Namespace IRI
Ifcowl	<a href="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#">http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#</a>
inst	<a href="http://linkedbuildingdata.net/ifc/resources20170627_104702/">http://linkedbuildingdata.net/ifc/resources20170627_104702/</a>
list	<a href="https://w3id.org/list#">https://w3id.org/list#</a>
express	<a href="https://w3id.org/express#">https://w3id.org/express#</a>
rdf	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a>
xsd	<a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a>
owl	<a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a>

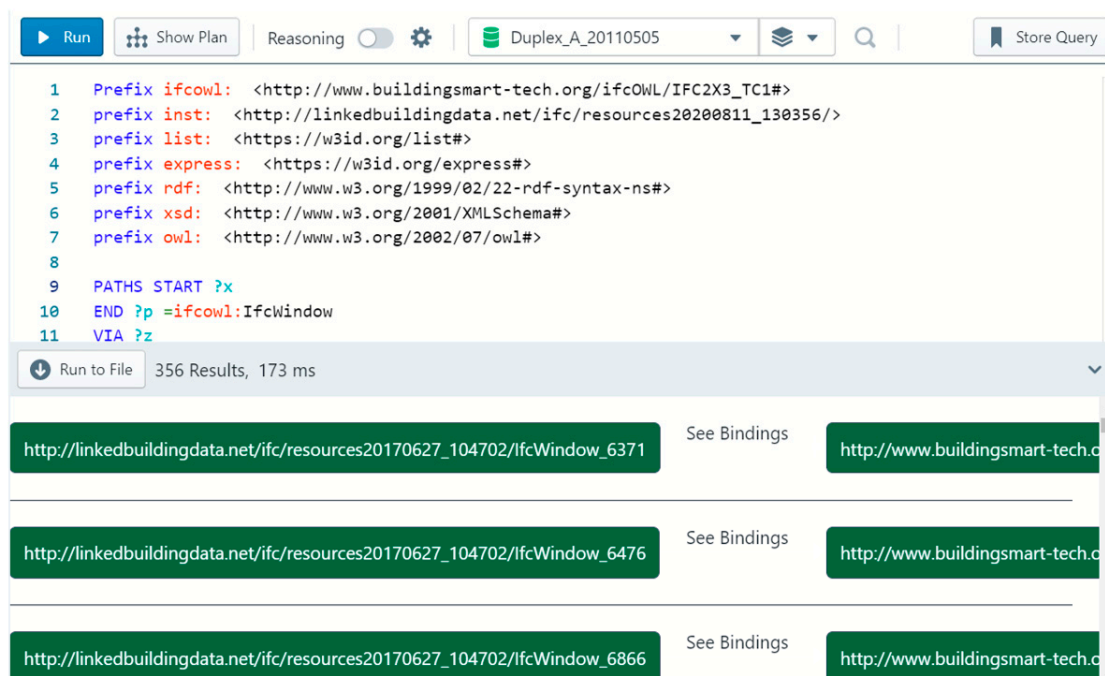


**Figure 4.** A duplex apartment BIM project.



When one tries to query window information for the second floor, the target in the query is filled with “window” and the query condition is filled with “Level 2”, listed in Table 1a. In terms of string matching, “Window” is the closest to “IfcWindow” because window instances in BIM models generally have longer names than “IfcWindow”. We utilized “IfcWindow” and “Level 2” as the path beginning and ending to search data paths in the semantic BIM case stored in the Stardog RDF database. The results of Stardog path query and the corresponding path query codes are shown in Figures 5 and 6. There were 356 paths ending in “IfcWindow”, and no path began with “IfcWindow”. The results of Stardog path query were exported to a csv format file using the “Run to File” function in Stardog. Similarly, we obtained 236 paths ending in “Level 2” and exported the results to a csv file. The following step was to find the same node(s) in these two csv files and the shortest path connecting these two keywords. Because there were only two path files in this case, searching common nodes in the two path files was easy through different implementation methods. For example, csv files can be opened in Excel software, and we can utilize the search function in Excel to find common nodes between two path files. We could also achieve the searching operation in Python. When all common nodes were found, the shortest path(s) connecting the two query keywords could also be identified, shown in Figure 7. After that, we used SPARQL variables to replace the entity nodes.

When the endpoint nodes were specified keywords or instance classes, they were not replaced with variables, illustrated in Figure 8. Every arc and the two corresponding connected nodes in Figure 8 generated a triple in the *WHERE* clause, while the target in the *SELECT* clause was chosen as the instance variable linked with the class of IfcWindow. The final SPARQL query is shown in Figure 9. We then ran the SPARQL query in Stardog and gained 18 query results, also shown in Figure 9. The query results were verified as correct by comparing the 18 window results with the window information illustrated in Solibri Model Viewer software. This proved that our generated SPARQL query was correct.



```

1 Prefix ifcowl: <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
2 prefix inst: <http://linkedbuildingdata.net/ifc/resources20200811_130356/>
3 prefix list: <https://w3id.org/list#>
4 prefix express: <https://w3id.org/express#>
5 prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
6 prefix xsd: <http://www.w3.org/2001/XMLSchema#>
7 prefix owl: <http://www.w3.org/2002/07/owl#>
8
9 PATHS START ?x
10 END ?p =ifcowl:IfcWindow
11 VIA ?z

```

Run to File 356 Results, 173 ms

<a href="http://linkedbuildingdata.net/ifc/resources20170627_104702/ifcWindow_6371">http://linkedbuildingdata.net/ifc/resources20170627_104702/ifcWindow_6371</a>	See Bindings	<a href="http://www.buildingsmart-tech.org/ifc/IFC2X3_TC1/IfcWindow">http://www.buildingsmart-tech.org/ifc/IFC2X3_TC1/IfcWindow</a>
<a href="http://linkedbuildingdata.net/ifc/resources20170627_104702/ifcWindow_6476">http://linkedbuildingdata.net/ifc/resources20170627_104702/ifcWindow_6476</a>	See Bindings	<a href="http://www.buildingsmart-tech.org/ifc/IFC2X3_TC1/IfcWindow">http://www.buildingsmart-tech.org/ifc/IFC2X3_TC1/IfcWindow</a>
<a href="http://linkedbuildingdata.net/ifc/resources20170627_104702/ifcWindow_6866">http://linkedbuildingdata.net/ifc/resources20170627_104702/ifcWindow_6866</a>	See Bindings	<a href="http://www.buildingsmart-tech.org/ifc/IFC2X3_TC1/IfcWindow">http://www.buildingsmart-tech.org/ifc/IFC2X3_TC1/IfcWindow</a>

Figure 5. The results of Stardog path query of the keyword “IfcWindow”.

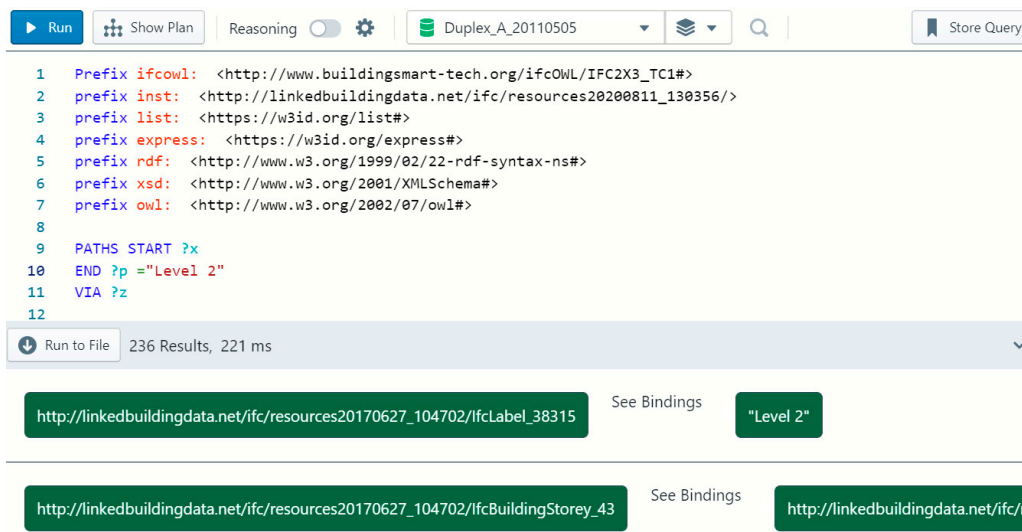


Figure 6. The result of Stardog path query of the keyword “Level 2”.

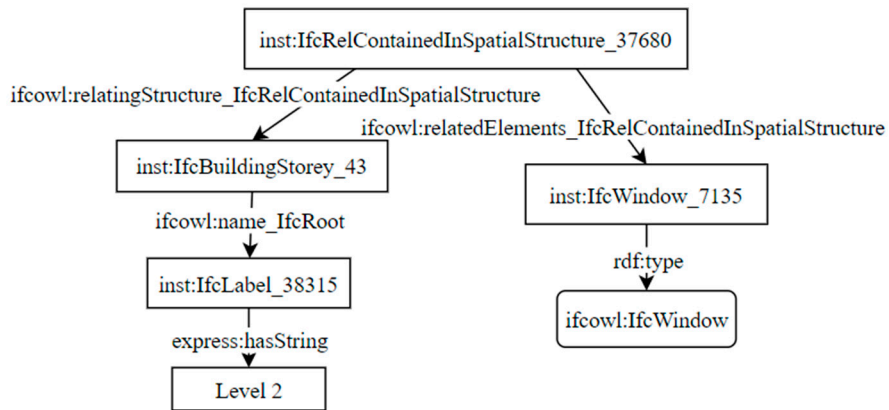


Figure 7. The shortest path connecting the two keywords.

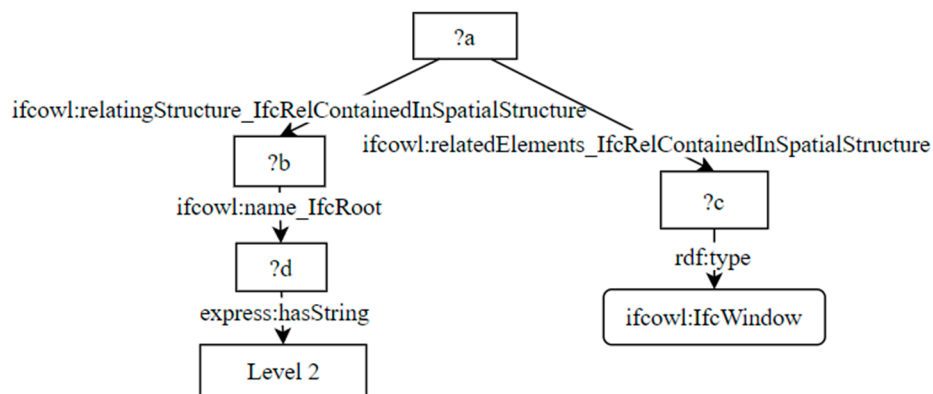


Figure 8. The structure of the shortest path with query variables.

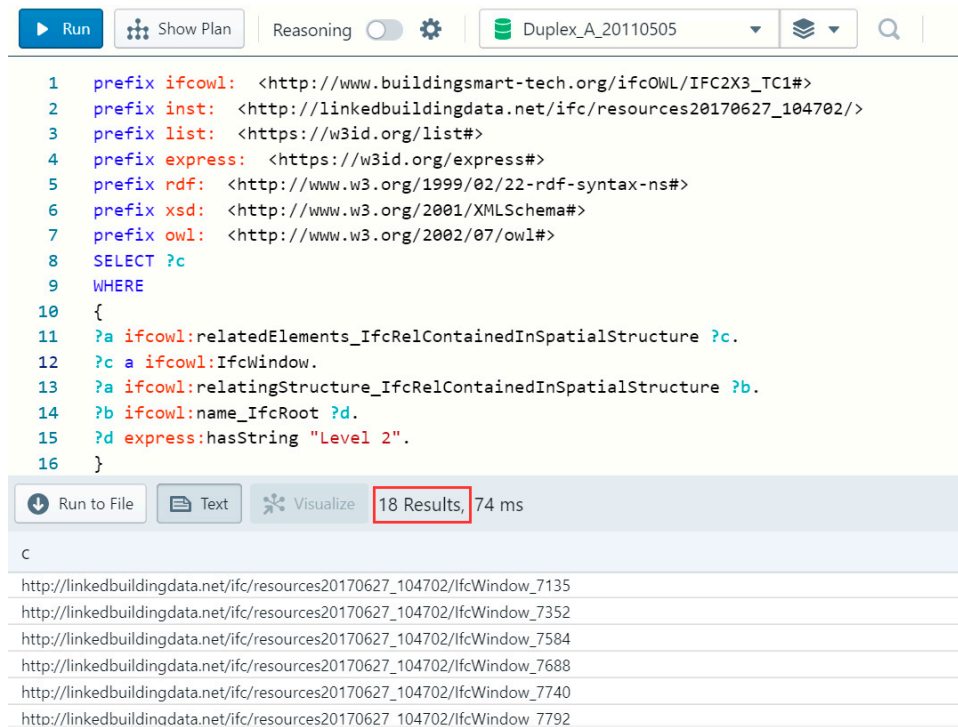


Figure 9. The generated SPARQL query and query results.

#### 4.2. Case Study Two: A Single House in Norway

We tested our approach using another BIM case, shown in Figure 10. Table 1b lists the query requirements to extract all external walls on the second floor and the Globally Unique Identifiers (GUIDs) of these walls in the BIM model. Because there are two properties to jointly express external walls in the IFC schema (IsExternal property and a Boolean value), the total three query conditions are “Plan E2”, External, and TURE, listed in Table 1b. Table 3 lists the prefixes and IRI in this case.

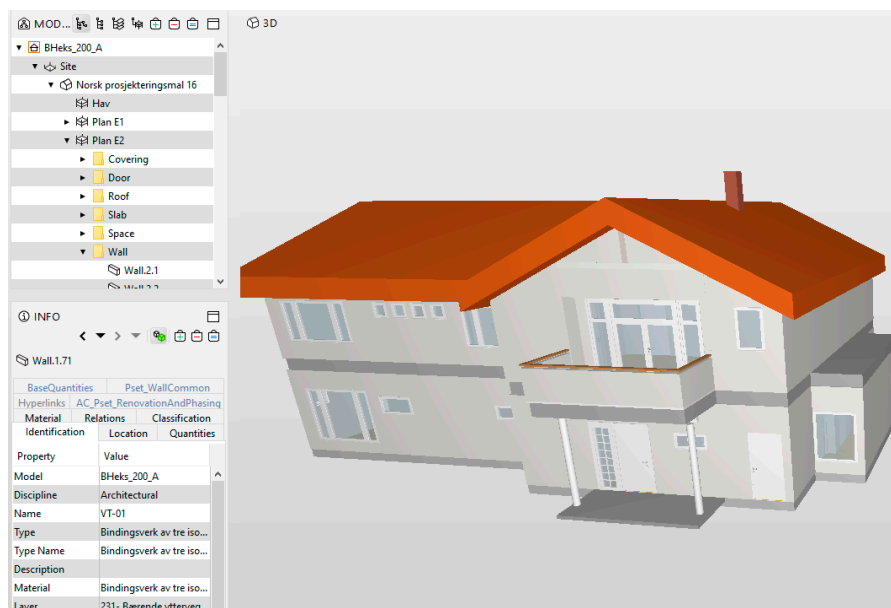


Figure 10. A single house project in Norway.

**Table 3.** The prefixes in this Norwegian case.

Prefix	Namespace IRI
Ifc	<a href="http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#">http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#</a>
inst	<a href="http://linkedbuildingdata.net/ifc/resources20200811_130356/">http://linkedbuildingdata.net/ifc/resources20200811_130356/</a>
list	<a href="https://w3id.org/list#">https://w3id.org/list#</a>
express	<a href="https://w3id.org/express#">https://w3id.org/express#</a>
rdf	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a>
xsd	<a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a>
owl	<a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a>

Similarly, we implemented the path queries in Stardog using five different query keywords, shown in Figure 11. When the results of a Stardog path query number more than 1000 paths, only 1000 results are shown in the Stardog database. However, when the path queries are exported through “Run to File”, all query results can be stored in a csv file. In this case, the core keyword was “wall”, related to all other query keywords, and the closest term to “wall” was “IfcWall” in the BIM knowledge base, so the path file of the keyword “IfcWall” was viewed as the main target.

```

1 prefix ifc: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#>
2 prefix inst: <http://linkedbuildingdata.net/ifc/resources20200811_130356/>
3 prefix list: <https://w3id.org/list#>
4 prefix express: <https://w3id.org/express#>
5 prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
6 prefix xsd: <http://www.w3.org/2001/XMLSchema#>
7 prefix owl: <http://www.w3.org/2002/07/owl#>
8 PATHS START ?x
9 END ?p= "Plan E2"
10 VIA ?z
    
```

Run to File 6 Results 96 ms

(a) The codes and results of path queries based on the keyword of “Plan E2”

```

1 prefix ifc: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#>
2 prefix inst: <http://linkedbuildingdata.net/ifc/resources20200811_130356/>
3 prefix list: <https://w3id.org/list#>
4 prefix express: <https://w3id.org/express#>
5 prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
6 prefix xsd: <http://www.w3.org/2001/XMLSchema#>
7 prefix owl: <http://www.w3.org/2002/07/owl#>
8 PATHS START ?x
9 END ?p= ifc:IfcWall
10 VIA ?z
    
```

Run to File 1,000 Results 219 ms

(b) The results of path queries based on the keyword of “ifc:IfcWall”

```

8 PATHS START ?x
9 END ?p= "IsExternal"
10 VIA ?z
    
```

Run to File 381 Results 236 ms

(c) The results of path queries based on the keyword of “IsExternal”

**Figure 11.** Cont.

```

8 PATHS START ?x
9 END ?p= TRUE
10 VIA ?z

```

Run to File 1,000 Results, 527 ms

(d) The results of path queries based on the keyword of "TRUE"

```

8 PATHS START ?x
9 END ?p= ifc:IfcGloballyUniqueId
10 VIA ?z

```

Run to File 1,000 Results, 105 ms

(e) The results of path queries based on the keyword of "ifc:IfcGloballyUniqueId"

Figure 11. The codes and results of path queries based on different query keywords.

As the first iteration, we began to search the common node(s) between the path file for "IfcWall" and the other path file, generating paths that connected the keyword "IfcWall" and the other query keyword, and then stored the shortest path(s) into a shortest-path file. In the first iteration, we obtained four shortest-path files. The next iteration was to search for common node(s) between pairs of shortest-path files. Two paths were merged into a new path through the common node(s) and stored in a new shortest-path file for the next iteration. For example, Zone I in Figure 12 shows the shortest path connecting "IfcWall" and "Plan E2" obtained in the first iteration. The other shortest path in the other path file was merged into this path, and then a new shortest path was generated, shown in the combination of Zone I and Zone II in Figure 12. Nodes in the red dashed circle are common nodes between these two shortest paths, which connect the two paths. However, ifc:IfcGloballyUniqueId is a class, not a value. Actually, the GUID value of an instance has a same upper node as the node "ifc:IfcGloballyUniqueId" of this instance. When finding the node of "ifc:IfcGloballyUniqueId" of an instance, one also finds the position of the GUID value of an instance because they have a fixed structure in the ifcOWL schema. So, we can add the fixed structure in the shortest path to find the GUID value, shown in the red font in Zone II of Figure 12.

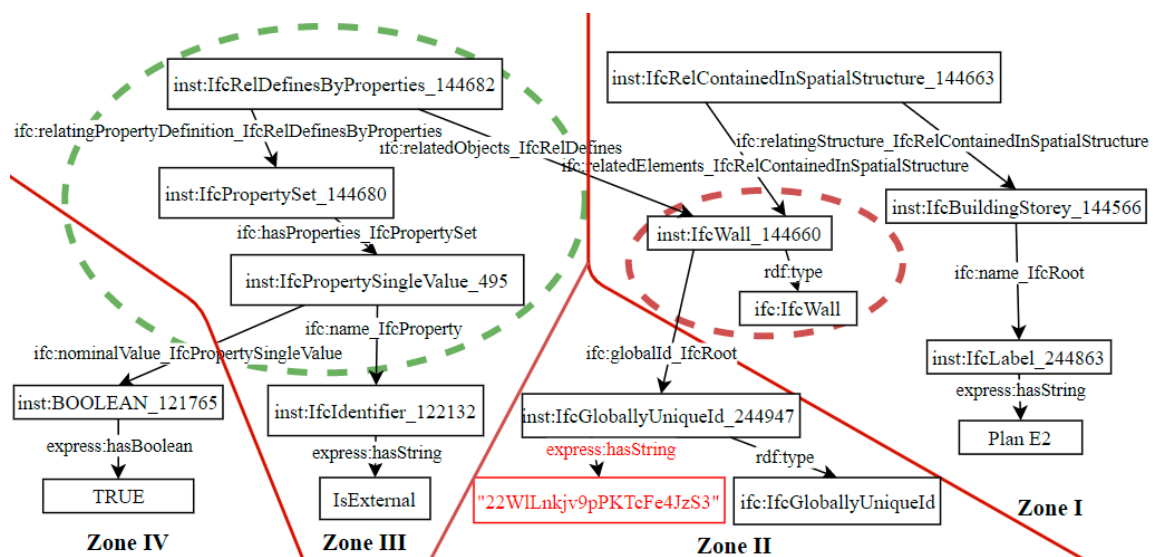


Figure 12. The structure of the shortest path of query requirements.



In the third iteration, a path in the path file of the keyword “IsExternal” was joined into the new shortest path through the common node(s) in the red dashed circle in Figure 12. The result of the third iteration is shown in the combination of Zones I, II, and III in Figure 12. In the last iteration, the path in the last shortest-path file was combined into the shortest path that was gained in the third iteration. The greed and red dashed circles cover all common nodes in this iteration; however, the common nodes were only added into the shortest path once. So, the whole of Figure 12 illustrates the final result of the shortest path connecting the five query keywords in this semantic BIM case.

We used SPARQL variables to replace the node instances, shown in Figure 13. Figure 14 exhibits the generated SPARQL query and the query results of SPARQL codes. When compared with the external walls of the second floor in the Solibri Model Viewer tool, the query results from the generated SPARQL query were found to be correct, which also proves that our approach is effective and correct.

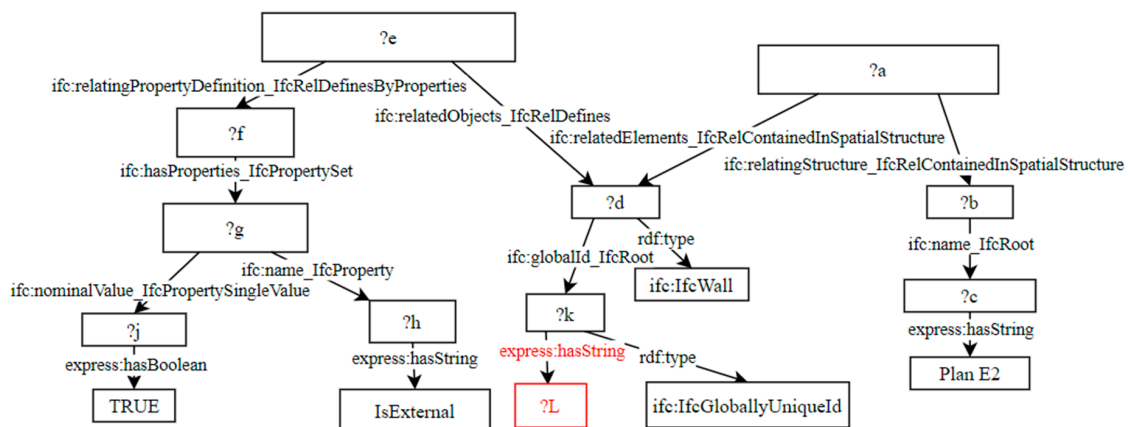


Figure 13. The structure of the shortest path with query variables in this case.

```

8 SELECT ?d ?L
9 WHERE
10 {?a ifc:relatingStructure_IfcRelContainedInSpatialStructure ?b.
11 ?b ifc:name_IfcRoot ?c.
12 ?c express:hasString "Plan E2".
13 ?a ifc:relatedElements_IfcRelContainedInSpatialStructure ?d.
14 ?d a ifc:IfcWall.
15 ?e ifc:relatedObjects_IfcRelDefines ?d.
16 ?e ifc:relatingPropertyDefinition_IfcRelDefinesByProperties ?f.
17 ?f ifc:hasProperties_IfcPropertySet ?g.
18 ?g ifc:name_IfcProperty ?h.
19 ?h express:hasString "IsExternal".
20 ?g ifc:nominalValue_IfcPropertySingleValue ?j.
21 ?j express:hasBoolean true.
22 ?d ifc:globalId_IfcRoot ?k.
23 ?k a ifc:IfcGloballyUniqueId.
24 ?k express:hasString ?L. }
    
```

Run to File		Text	Visualize	17 Results, 45 ms
d				L
http://linkedbuildingdata.net/ifc/resources20200811_130356/IfcWall_146468				
http://linkedbuildingdata.net/ifc/resources20200811_130356/IfcWall_148977				
http://linkedbuildingdata.net/ifc/resources20200811_130356/IfcWall_149111				
http://linkedbuildingdata.net/ifc/resources20200811_130356/IfcWall_149230				

Figure 14. The generated SPARQL query and results of query codes run in this BIM case.

We implemented our approach using the Python programming language to automatically search common nodes and generate the SPARQL query. We also manually executed our approach to generate the SPARQL query, and only spent a couple of minutes on creating the final SPARQL query in the second case. So, it is feasible for engineers of different stakeholders without SPARQL programming skills to generate the required SPARQL query for BIM data extraction by using our approach.

## 5. Conclusions

The SPARQL query language is widely used in different data extraction or rule compliance checking applications in AEC industry; however, SPARQL queries are generally programmed manually by experts/engineers. To automate this process, in this paper, we proposed an approach for the automatic generation of SPARQL codes without programming proficiency or full understanding of the structure of the semantic BIM model. To allow engineers without SPARQL programming experience to obtain the SPARQL queries they desire, we used the path query function provided by the Stardog RDF database to achieve path finding in the semantic BIM model. By searching the common nodes in different path files, the shortest path(s) containing all query keywords can be obtained. Because of the numerous congeneric products/elements in a BIM model, the shortest path may not be unique, but this does not affect the structure of the shortest path. By its structure information, the *WHERE* and *SELECT* clauses in the SPARQL query can be produced. After two case studies, we verified that the generated SPARQL queries are correct and our approach is effective. Additionally, our approach can be flexibly used in different IFC versions or semantic/information enriched BIM models.

However, natural language processing technologies were not applied in this paper, so the query keywords written by users are restricted to some extent. For example, it is best for the query keywords to be similar to the terms in the BIM knowledge base because only string matching was adopted in our approach to find the searched terms. Additionally, data extraction of the complex relationships among multiple targets will produce a great many Stardog path query files in our approach, and it may be difficult to find the shortest path to connect all keywords manually. At this time, a common node search program may be required. The usage of the Stardog RDF database simplifies the exploration of the linked relationships (path) among different query keywords. Certainly, the linked paths among query keywords may be identified through programming or other tools. Meanwhile, our approach mainly depends on path exploration in the explicit BIM model. Implicit data extraction needs further research in the future.

**Author Contributions:** Resources and data collection, D.G. and E.O.; methodology, D.G.; formal analysis and investigation, D.G.; writing—original draft preparation, D.G.; writing—review and editing, D.G., E.O. and A.D.L.R. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Acknowledgments:** NTNU Open Access publishing funds covered the article processing charges.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Santos, R.; Costa, A.A.; Silvestre, J.D.; Pyl, L. Informetric analysis and review of literature on the role of BIM in sustainable construction. *Autom. Constr.* **2019**, *103*, 221–234. [[CrossRef](#)]
2. Soust-Verdaguer, B.; Llatas, C.; García-Martínez, A. Critical review of bim-based LCA method to buildings. *Energy Build.* **2017**, *136*, 110–120. [[CrossRef](#)]
3. Chong, H.Y.; Lee, C.-Y.; Wang, X. A mixed review of the adoption of Building Information Modelling (BIM) for sustainability. *J. Clean. Prod.* **2017**, *142*, 4114–4126. [[CrossRef](#)]
4. Lu, Q.; Xie, X.; Heaton, J.; Parlikad, A.K.; Schooling, J. From BIM towards digital twin: Strategy and future development for smart asset management. In *Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future*; Springer: Cham, Switzerland, 2020; pp. 392–404.
5. Janjua, S.Y.; Sarker, P.K.; Biswas, W.K. A Review of Residential Buildings' Sustainability Performance Using a Life Cycle Assessment Approach. *J. Sustain. Res.* **2019**, *1*, 190006.

6. Seghier, T.E.; Ahmad, M.H.; Wah, L.Y.; Samuel, W.O. Integration Models of Building Information Modelling and Green Building Rating Systems: A Review. *Adv. Sci. Lett.* **2018**, *24*, 4121–4125. [[CrossRef](#)]
7. Kehily, D.; Underwood, J. Embedding life cycle costing in 5D BIM. *J. Inf. Technol. Constr.* **2017**, *22*, 145–167.
8. Marzouk, M.; Azab, S.; Metawie, M. BIM-based approach for optimizing life cycle costs of sustainable buildings. *J. Clean. Prod.* **2018**, *188*, 217–226. [[CrossRef](#)]
9. Mihić, M.; Vukomanović, M.; Završki, I. Review of previous applications of innovative information technologies in construction health and safety. *Organ. Technol. Manag. Constr. Int. J.* **2019**, *11*, 1952–1967. [[CrossRef](#)]
10. Plebankiewicz, E.; Juszczak, M.; Kozik, R. Trends, Costs, and Benefits of Green Certification of Office Buildings: A Polish Perspective. *Sustainability* **2019**, *11*, 2359. [[CrossRef](#)]
11. Sandberg, M.; Mukkavaara, J.; Shadram, F.; Sandberg, M. Multidisciplinary Optimization of Life-Cycle Energy and Cost Using a BIM-Based Master Model. *Sustainability* **2019**, *11*, 286. [[CrossRef](#)]
12. Nwodo, M.N.; Anumba, C.J.; Asadi, S. BIM-Based Life Cycle Assessment and Costing of Buildings: Current Trends and Opportunities. In Proceedings of the ASCE International Workshop on Computing in Civil Engineering (IWCC), Seattle, WA, USA, 25–27 June 2017; Lin, K.Y., El Gohary, N., Tang, P., Eds.; University of Washington: Seattle, WA, USA, 2017; pp. 51–59.
13. ISO. ISO 16739:2013 Industry Foundation Class (IFC) for Data Sharing in the Construction and Facility Management Industries. Available online: <https://www.iso.org/standard/51622.html> (accessed on 8 September 2020).
14. ISO. ISO 16739-1:2018 Industry Foundation Classes (IFC) for Data Sharing in the Construction and Facility Management Industries—Part 1: Data schema. Available online: <https://www.iso.org/standard/70303.html> (accessed on 8 September 2020).
15. Zhang, L.; El-Gohary, N.M. Automated IFC-based building information modelling and extraction for supporting value analysis of buildings. *Int. J. Constr. Manag.* **2020**, *20*, 269–288. [[CrossRef](#)]
16. Isailović, D.; Stojanovic, V.; Trapp, M.; Richter, R.; Hajdin, R.; Döllner, J. Bridge damage: Detection, IFC-based semantic enrichment and visualization. *Autom. Constr.* **2020**, *112*, 103088. [[CrossRef](#)]
17. Gui, N.; Wang, C.; Qiu, Z.; Gui, W.; Deconinck, G. IFC-Based Partial Data Model Retrieval for Distributed Collaborative Design. *J. Comput. Civ. Eng.* **2019**, *33*, 04019016. [[CrossRef](#)]
18. Pauwels, P.; Van Deursen, D.; Verstraeten, R.; De Roo, J.; De Meyer, R.; Van De Walle, R.; Van Campenhout, J. A semantic rule checking environment for building performance checking. *Autom. Constr.* **2011**, *20*, 506–518. [[CrossRef](#)]
19. Zhang, C.; Beetz, J.; De Vries, B. BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data. *Semantic Web* **2018**, *9*, 829–855. [[CrossRef](#)]
20. Zhong, B.; Wu, H.; Li, H.; Sepasgozar, S.; Luo, H.; He, L. A scientometric analysis and critical review of construction related ontology research. *Autom. Constr.* **2019**, *101*, 17–31. [[CrossRef](#)]
21. Nacer, H.; Aissani, D. Semantic web services: Standards, applications, challenges and solutions. *J. Netw. Comput. Appl.* **2014**, *44*, 134–151. [[CrossRef](#)]
22. Pauwels, P.; Terkaj, W. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Autom. Constr.* **2016**, *63*, 100–133. [[CrossRef](#)]
23. Xu, Z.; Wang, X.; Zhou, W.; Yuan, J. Study on the Evaluation Method of Green Construction Based on Ontology and BIM. *Adv. Civ. Eng.* **2019**, *2019*, 5650463. [[CrossRef](#)]
24. Pauwels, P.; Zhang, S.; Lee, Y.-C. Semantic web technologies in AEC industry: A literature overview. *Autom. Constr.* **2017**, *73*, 145–165. [[CrossRef](#)]
25. De Farias, T.M.; Roxin, A.; Nicolle, C. A rule-based methodology to extract building model views. *Autom. Constr.* **2018**, *92*, 214–229. [[CrossRef](#)]
26. Mazairac, L.W.; Beetz, J.J. BIMQL—An open query language for building information models. *Adv. Eng. Inform.* **2013**, *27*, 444–456. [[CrossRef](#)]
27. Krijnen, T.; Beetz, J. A SPARQL query engine for binary-formatted IFC building models. *Autom. Constr.* **2018**, *95*, 46–63. [[CrossRef](#)]
28. Daum, S.; Borrmann, A. Processing of Topological BIM Queries using Boundary Representation Based Methods. *Adv. Eng. Inform.* **2014**, *28*, 272–286. [[CrossRef](#)]
29. Ismail, A.; Nahar, A.; Scherer, R. Application of graph databases and graph theory concepts for advanced analysing of BIM models based on IFC standard. In Proceedings of the 24th International Workshop on Intelligent Computing in Engineering, Nottingham, UK, 10–12 July 2017.

30. Nepal, M.; Staub-French, S.; Pottinger, R.; Webster, A. Querying a building information model for construction-specific spatial information. *Adv. Eng. Inform.* **2012**, *26*, 904–923. [CrossRef]
31. Schevers, H.; Drogemuller, R. Converting the Industry Foundation Classes to the Web Ontology Language. In Proceedings of the 2005 1st International Conference on Semantics, Knowledge and Grid, Beijing, China, 27–29 November 2005.
32. Beetz, J.J.; Van Leeuwen, J.J.; De Vries, B.B. IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Artif. Intell. Eng. Des. Anal. Manuf.* **2008**, *23*, 89–101. [CrossRef]
33. Terkaj, W.; Sojic, A. Ontology-based representation of IFC EXPRESS rules: An enhancement of the ifcOWL ontology. *Autom. Constr.* **2015**, *57*, 188–201. [CrossRef]
34. Boje, C.; Guerriero, A.; Kubicki, S.; Rezgui, Y. Towards a semantic Construction Digital Twin: Directions for future research. *Autom. Constr.* **2020**, *114*, 103179. [CrossRef]
35. Nepal, M.; Staub-French, S.; Pottinger, R.; Zhang, J. Ontology-Based Feature Modeling for Construction Information Extraction from a Building Information Model. *J. Comput. Civ. Eng.* **2013**, *27*, 555–569. [CrossRef]
36. Zhang, L.; Issa, R.R.A. Ontology-Based Partial Building Information Model Extraction. *J. Comput. Civ. Eng.* **2013**, *27*, 576–584. [CrossRef]
37. Niknam, M.; Karshenas, S. A shared ontology approach to semantic representation of BIM data. *Autom. Constr.* **2017**, *80*, 22–36. [CrossRef]
38. Beetz, J.; De Vries, B.; Van Leeuwen, J. RDF-based distributed functional part specifications for the facilitation of service-based architectures. In Proceedings of the 24th W78 Conference, Maribor, Slovenia, 27–29 June 2007.
39. Liu, H.; Lu, M.; Al-Hussein, M. Ontology-based semantic approach for construction-oriented quantity take-off from BIM models in the light-frame building industry. *Adv. Eng. Inform.* **2016**, *30*, 190–207. [CrossRef]
40. Krijnen, T.; Van Berlo, L. Methodologies for requirement checking on building models. In Proceedings of the DDSS2016 13th International Conference on Design & Decision Support Systems in Architecture and Urban Planning, Eindhoven, The Netherlands, 27–28 June 2016.
41. Bouzidi, K.R.; Fies, B.; Faron-Zucker, C.; Zarli, A.; Le Thanh, N. Semantic Web Approach to Ease Regulation Compliance Checking in Construction Industry. *Future Internet* **2012**, *4*, 830–8511. [CrossRef]
42. Zhong, B.; Gan, C.; Luo, H.; Xing, X. Ontology-based framework for building environmental monitoring and compliance checking under BIM environment. *Build. Environ.* **2018**, *141*, 127–142. [CrossRef]
43. Fahad, M.; Bus, N.; Fies, B. Semantic BIM reasoner for the verification of IFC Models. In *Ework and Ebusiness in Architecture, Engineering and Construction*; Karlshoj, J., Scherer, R., Eds.; CRC Press: Boca Raton, FL, USA, 2018; pp. 361–368.
44. The RDF Working Group. Resource Description Framework (RDF). 2014. Available online: <https://www.w3.org/RDF/> (accessed on 8 September 2020).
45. Group, T.W.C.S.W. SPARQL 1.1 Overview W3C Recommendation 21 March 2013. 2013. Available online: <https://www.w3.org/TR/2013/REC-sparql11-query-20130321/> (accessed on 8 September 2020).
46. Haarslev, V.; Pai, H.-I.; Shiri, N. Uncertainty Reasoning for Ontologies with General TBoxes in Description Logic. In *Uncertainty Reasoning for the Semantic Web I: URSW 2006, URSW 2007, URSW 2005. Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5327.
47. Stardog Union. Stardog 7: The Manual—Documentation Page. 2020. Available online: <https://www.stardog.com/docs/> (accessed on 7 June 2020).
48. Pauwels, P. IFC to RDF Tool. 2020. Available online: <https://github.com/pipauwel/IFCtoRDF> (accessed on 8 September 2020).
49. DuplexModel-IFC-2011-05-05. Available online: <http://smartlab1.elis.ugent.be:8889/IFC-repo/http.openifcmodel.cs.auckland.ac.nz/030811DuplexModel-IFC-2011-05-05/> (accessed on 8 December 2020).
50. Solibri, Solibri Model Viewer. Available online: <http://www.solibri.com/products/solibri-model-viewer/> (accessed on 10 May 2020).

**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).