Application of graph databases and graph theory concepts for advanced analysing of BIM models based on IFC standard

Ali Ismail, Ahmed Nahar, Raimar Scherer
TU Dresden, Germany
ali.ismail@tu-dresden.de

Abstract. In this paper we present a workflow for automatic transformation of IFC schema and IFC models into an IFC Meta and object graph databases. The aim of this research is to study and demonstrate the potential of using graph theory concepts and graph databases in order to manage, visualize and analyse the huge information and complex relationships of BIM models based on IFC standard. For the validation a set data retrieval queries and advanced model analysis for model topology analysis and comparison of different IFC models are carried out in order to demonstrate the flexibility and advantages of the suggested approach.

1. Introduction

The very fast development in the sector of information technology has been successfully exploited in construction and engineering field to adopt new digital methods such as Building Information Modelling (BIM) for construction project managing. However, BIM models may contain a huge amount on information with complex relationships between the model entities. This information could remain inaccessible in several cases due to the use of closed property formats or the absent of suitable data management tools in case of using open standard formats like Industry Foundation Classes (IFC).

A lot of data retrieval queries are hard to be accomplished using currently available software's and most of them operate on single IFC models. The rigid and complex hierarchical structure of the IFC schema prevents simple manual extraction of building information and requires deep understanding of the IFC object model itself. The BIM query languages introduced so far have certain limitations, particularly with respect to the high level of knowledge about the IFC object model and about data mapping mechanisms required by the user (Tauscher, Bargstädt, & Smarsly, 2016).

Evidently, graphs have shown great capabilities in understanding and accessing complex and rich datasets in many different domains. Graph models are extremely useful for representation and description of the complex relationships among building elements and data within BIMs (Isaac, Sadeghpour, & Navon, 2013), hence converting of BIM models based on the IFC standard into an effective information retrievable model based on graph databases could significantly facilitate the efforts of exploring and analysing BIM highly connected data.

A graph-based schema, termed the graph data model (GDM) was presented by (Khalili & Chua, 2015). This schema can be used to employ semantic information, to extract, analyse, and present the topological relationships among 3D objects in 3D space, and to perform topological queries faster. Another generic approach towards information retrieval using the IFC object model based on graph theory was presented recently by (Tauscher, Bargstädt, & Smarsly, 2016). In this approach a directed graph was generated that serve as semantic data pools facilitating generic queries. This approach is limited to apply queries on single IFC models.

This paper presents a workflow for complete and automatic transformation of IFC models into a labelled property graph-based model using the world leading graph database Neo4J [1] as a graph database framework.

The proposed approach targets two kinds of graph models:

- IFC Meta Graph (IMG) based on the IFC EXPRESS schema
- IFC Objects Graph (IOG) based on STEP Physical File (SPF) format.

Beside the automatic conversation of IFC models into graph database this paper aims to demonstrate the potential of using graph databases and concepts of graph theory in order to (1) explore, check and analyse the complex relationships inside one or multiple BIM models, (2) run complex queries for information retrieval and (3) carry out advanced analysis of the building topology like escape route analysis and comparing of different IFC models.

2. IFC data model

IFC is a data model developed by buildingSMART International to support the exchange of building information in Architecture, Engineering and Construction (AEC) industry to improve collaboration, scheduling, cost and delivery time activities throughout the whole life cycle of the building project. It provides an object-oriented and semantic data model for storing and exchanging building information. This enables actors from different disciplines to effectively use BIM data and to fully exchange information in construction or facility management projects. IFC schema is described using EXPRESS specification language that defined by ISO 10303-11, as conceptual information modelling language. The specification specified also a graphical representation known as EXPRESS-G to provide graphical subset. The IFC data schema can be saved as Extensible Markup Language (XML) file structure per ISO10303-21 or STEP Physical file (SPF) document structure following ISO10303-28.

Each IFC model is composed of IFC entities built up in a hierarchical order, where each IFC entity includes a fixed number of IFC attributes, plus any number of additional IFC properties. The IFC attributes are the main identifiers of the entities, while the names of these attributes are fixed, having been defined by buildingSMART (buildingSMART) as part of the IFC standard code. The IFC data schema has three fundamental entity types as shown in Figure 1. These entity types are the first level structure of the IfcRoot entities hierarchical.

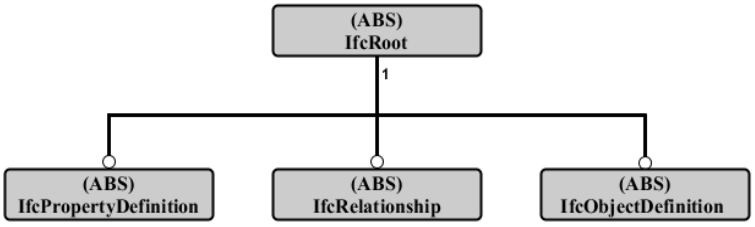


Figure 1: Fundamental entity types derived from IfcRoot class

- IfcPropertyDefinition describes all characteristics that may attach to objects. Thus, valuable information can be shared among multiple object instances. However, it may express the occurrence information of the actual object in the project context, in case that it is attached to a single object instance.

[1] www.neo4j.com

- IfcObjectDefinition stands for all handled objects or process. Where, all physical items and products such as roofs, windows, and slabs that can be touched and seen are classified as IfcObjectDefinition.
- IfcRelationship summarizes all the relationships among objects. This can enable users saving relationship specific properties directly at the relationship object and avoid duplication of relationship semantics from the object attributes. The abstract objectified relationship IfcRelationship and its subtype relationships are responsible for connectivity among objects, in which several properties can be attached to each relationship.

Figure 2 demonstrates the different types of objectified relationship within the IFC (Version 2x3) and their abstract super type IfcRelationship.

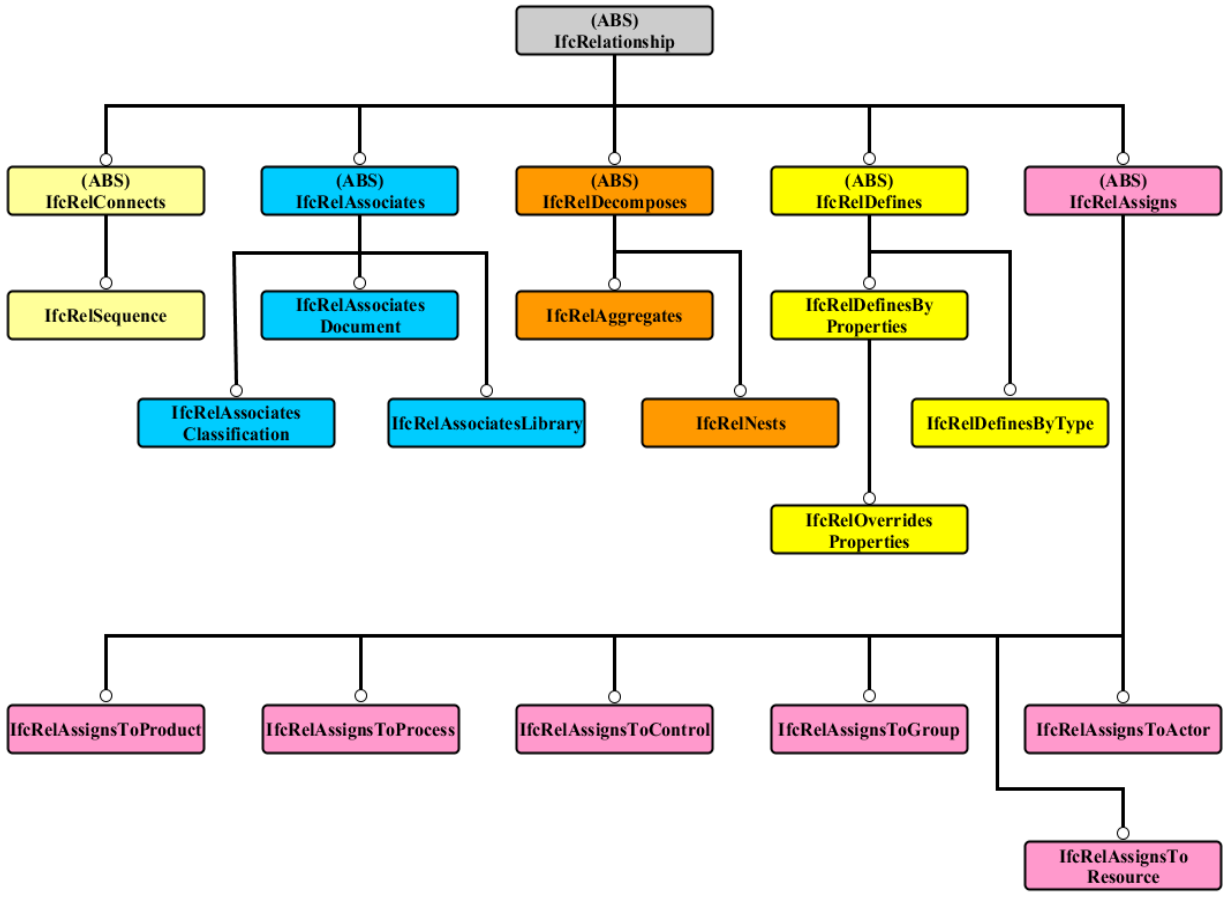


Figure 2: IFC abstract objectified relationship

IFC classes have direct attributes attached to them which could indicate a relationship to another object or they could just be attached as simple data type attribute, e.g. integer, string, logical, or Boolean. Therefore, IFC models distinguish between the attributes that are directly attached to object as entity attributes, and attributes that assigned to indicate a relationship to other objects. The second of two attributes are the most adopted approach to extend viable properties. However, as it is depicted Figure 3, each IFC class could have simple data attributes with referenced object attributes as in the case of the *IfcRoot*, or referenced object

with relationship attributes as in the case of IfcProduct, or entity attributes and relationship attributes together as in the case of IfcObject.

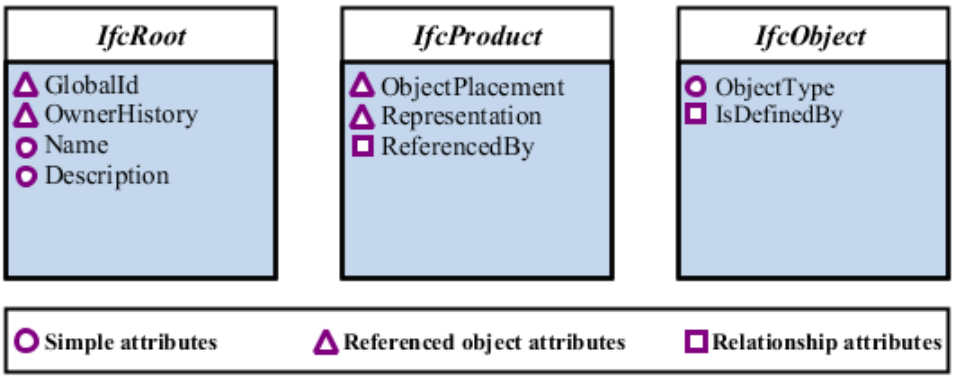


Figure 3 IFC Entity attributes and relationship attributes

In addition to the direct attached attributes, the objects can inherit attributes from their supertype classes, where, property pair can be assigned to almost any type of objects and thus to support enlarging their attributes base.

3. Graph data management

Graph theory is the branch of mathematics that dealing with the study of graphs (Hughes, 2016). While the concept of graphs itself is defined by graph theory as a diagrammatic representation of real-world scenarios in a form of points and lines (Wilson, 1996). The points are called vertices or nodes, while the lines that connect them are so-called edges, arcs or relationships. Each vertex in the graph is represented by drawing a dot or a circle, while the relationship between each pair of vertices is indicated by drawing an arc or line if they are connected by an edge.

The application of graphs has become an important technique to describe several scenarios in the real-world. One of the applications of graphs is to provide a simplified description of scenarios datasets in a way that produce a useful understanding of a complicated data. This has led to the birth of a special form of graph model, the so-called labelled property graph (Robinson, Webber , & Eifrem, 2015 "Second Edition"). Labelled property graphs are similar to simple graphs; consist of nodes and relationships which are often expressed as vertices and edges. However, labeled property graphs provide additional characteristics to facilitate graph understanding, where, nodes could have a single or multiple labels; in addition, they could have properties (key-value pairs). Relationships can also be named and contain properties while connecting each two nodes as start and end node.

Several graph processing systems have been developed in the last decade to meet the modern graph modelling and analysis tasks. Doekemeijer (Doekemeijer & Varbanescu, 2014- PDS-2014-003) has declared that more than 80 systems have been introduced in the period from 2004 to 2014, by academia and industry sectors together. However, the currently available systems can be divided into two main kinds, graph databases, and graph processing. In this section and for the objectives of the present study, we will express the concepts of graph database systems in general, with a focus on Neo4j graph database system particularly in some cases. However, all these efforts in the field of graph modelling express the importance

of graphs for real-world scenarios. Angles (Angles & Gutierrez, February 2008) summarized the advantages of using graphs as modelling mechanism for data management as following:

1. Graphs enable users to model data exactly as they are represented in the real-world scenario, this can significantly enhance the operations on data. Thus, graphs can keep all the information about an object in a single node and display the related information by relationships connected to it.
2. Queries can be developed based on the graph structure. For instance, the finding of the shortest path can be considered as sub graph from the original graph.
3. Operationally, graphs can be stored efficiently within databases using special graph storage structures, and functional graph algorithms for application of specific operations.

4. Methodology and Transformation Workflow

The proposed methodology introduces a workflow to develop and build IFC-based graph models using IFC EXPRESS schema and IFC models in STEP physical file format. In this workflow the whole transformation is automated and done through connected web services (dynamic EXPRESS parser, IFC web data server and web-based graph database import interface) without the need for any local tools. The transformation supports any valid IFC schema.

In our approach a single graph database is used to storage various IFC models at the same time, where each node (object) will has a special attribute to identify the belonged model. In this way queries and advanced analysis are not limited to one IFC model. This feature allows to do merging queries on different disciplines BIM models (architecture, structural design, MEP) of the same project, or to compare different versions of the same model over time.

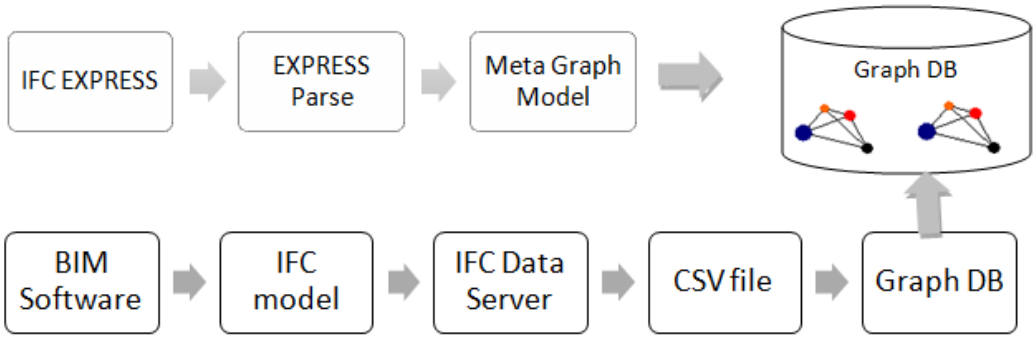


Figure 4: IFC to graph database conversation workflow

4.1 IFC Meta Graph Model

The workflow starts with analysing the IFC EXPRESS schema and developing a generic approach to generate an IFC Meta Graph Model (IMG), which represents all IFC classes, their attributes and the relationships between them (Figure 5).

IMG model will be used later in order to generate and check the relationships between IFC instance entities and validate IFC models. It can be also used to investigate changes between different IFC versions (e.g. IFC2X3 vs IFC4) or to run filters and information retrieval queries for a better understanding of IFC schema and analyzing its complex data connectivity.

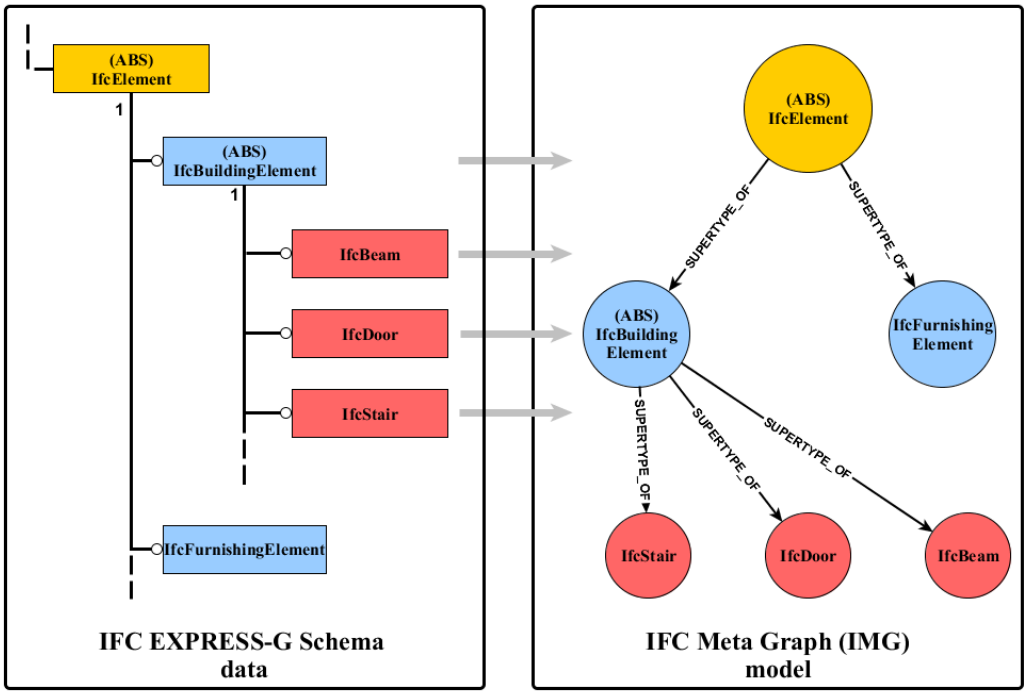


Figure 5 Mapping IFC EXPRESS into a Meta Graph Model

The mapping of IFC EXPRESS model into IMG is carried out through writing a special server script (in Ruby language) and run it on IFCWebServer.org (Ismail, 2011) which provides a dynamic EXPRESS parser and web script console. This script generates a set of Cyber commands, which create the Meta graph inside Neo4j database, where Cypher is a declarative, SQL-inspired language for describing patterns in graphs visually using an ascii-art syntax.

Figure 6 shows the scope of mapping for IFC classes, their attributes, data types and the relationships between classes. Both of IFC classes and the attributes are mapped into nodes and they are connected through various relationships like “has_property” to connect a class node with its direct attributes or “subtype_of” to connect a class with its sub-classes.

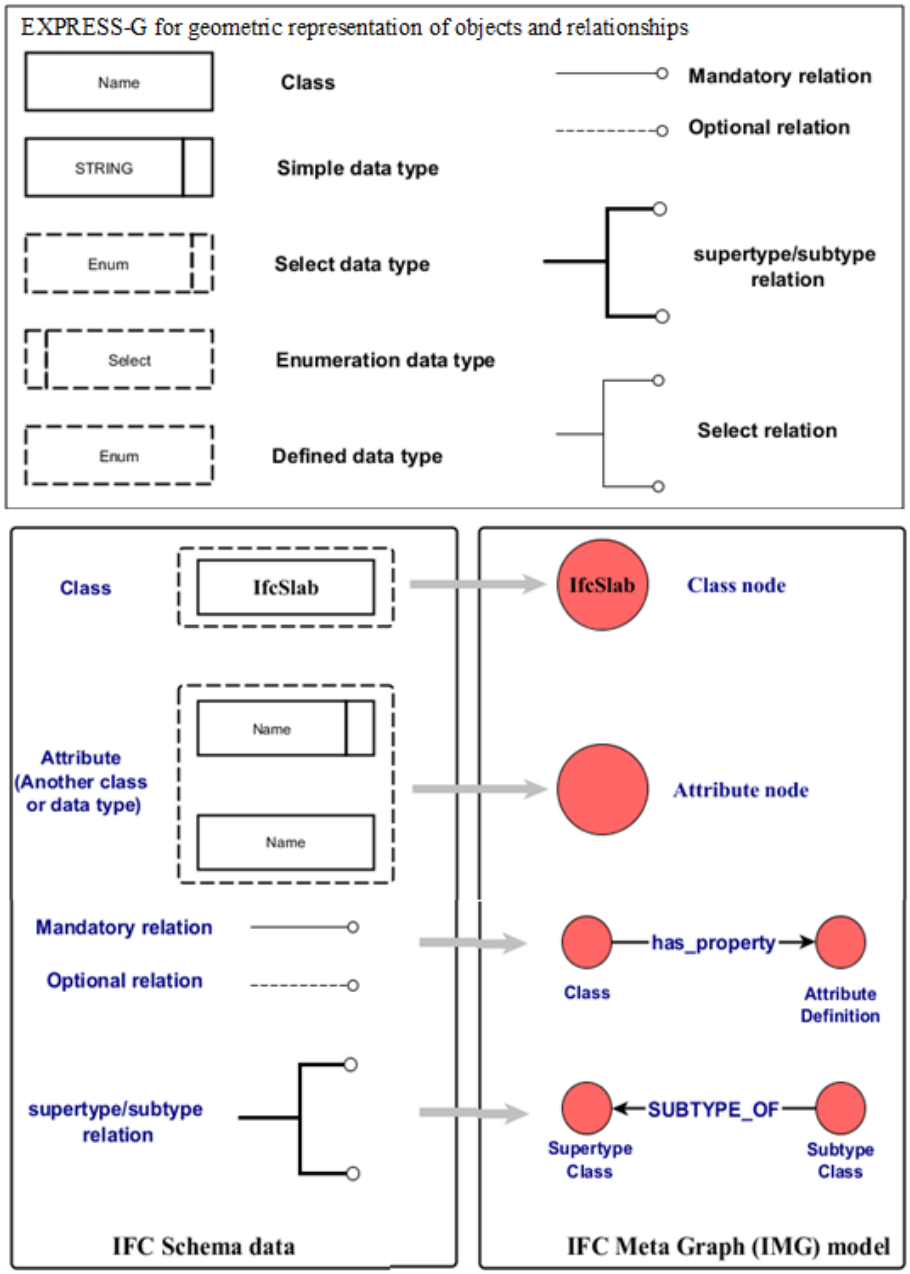


Figure 6 Mapping of classes, data types and relationships within IMG model

4.2 IFC Object Graph model

The next step in the workflow is to convert IFC models into IFC Object Graph (IOG). In this graph each IFC entity expect the relationships will be represented as a node and will hold information about the class of the entity and its basic attributes and will be connected to other entities through named relationships.

A special server script based on IFCWebServer.org has been developed in order to automate the process of extracting IFC data (objects, relationships) and creating all necessary Cypher commands for the data import and also relationship generation. The script include some options to exclude certain IFC classes, for example all low level geometry classes.

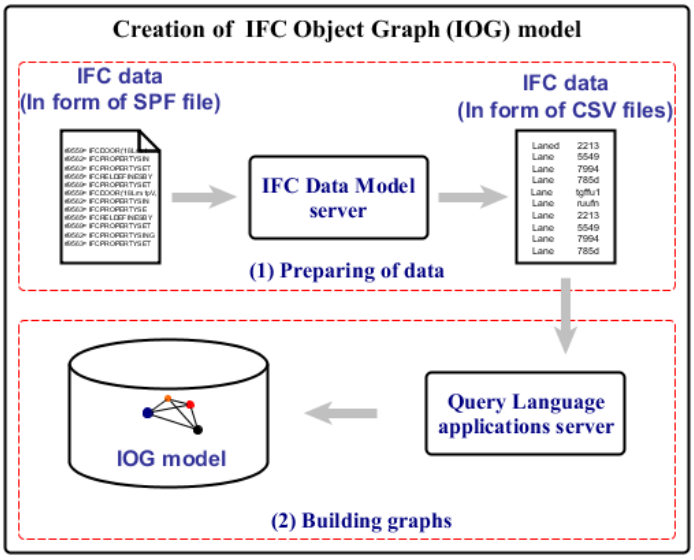


Figure 7 Creation of IFC Object Graph

IFC data are extracted and saved at first in CSV format then imported into Neo4j to create the graph-based models (Figure 7).

The Cypher snippet bellow shows the import command for IfcBuilding.

```
Load csv with headers from 'http://www.ifcwebserver.org/
/ahmednahar/Muster003.ifc/IfcBuilding.csv' as line FIELDTERMINATOR '    '
Create (u: IfcBuilding {Model: line.Model, label: line.label, IFCID: line.IFCID,
globalId: line.globalId, ownerHistory: line.ownerHistory, name: line.name,
description: line.description, objectType: line.objectType, objectPlacement:
line.objectPlacement, representation: line.representation, longName:
line.longName, compositionType: line.compositionType, elevationOfRefHeight:
line.elevationOfRefHeight, elevationOfTerrain: line.elevationOfTerrain,
buildingAddress: line.buildingAddress });
```

After importing the IFC data and with help of the Meta graph model the relationships for referenced, inversed and derived attributes will be created automatically. The scope of relationships transformation includes all IFC relationships, for example: (1) aggregation, (2) spatial and element composition, (3) spatial and element decomposition, (4) zone and group assignment, (5) building system assignment, (6) spatial structure, (7) element and path connectivity and (8) element filling.

The next step creates direct relationships between graph nodes and their connected information and deletes all redundant relationship. For example the building element nodes will be connected directly with their corresponding properties through a "isDefinedByProperties" relationships and all IfcRelDefinesByProperties and IfcRelDefinesByType can be deleted. As a result, the graph queries can be simplified by normalizing the property sets and non-direct attributes of building elements to be assigned as direct node attributes.

A further step will be assigning each node in the graph with a set of labels of all its parent classes and running a set of pre-defined queries for further classification and normalization of the building elements attributes according to their classes and property sets. For example all

walls which have the property “Load bearing” set to true will be linked into a node “Load bearing walls” through a relationship “is-load-bearing”.

5. Validation and Case Studies

After converting IFC models into a graph database it can be used as an IFC data server to run data retrieval queries and advanced analysis of BIM models taking in account the advantages of applying graph theory concepts and algorithms like path finding and shortest path method. In the following sections we present some simple and advanced examples.

5.1 Examples of Simple Queries

- **Retrieve the assigned property sets of a certain object:** This query returns all propery sets which are assigned to a ceartin object through its globalId value.

```
MATCH
(wall:IfcWallStandardCase{Model:'M.ifc',GUID:'3wP5TgnCHEPwsgi3SxIcXs'})-
[rel]-(property:IfcPropertySet)
RETURN DISTINCT wall,rel,property
```

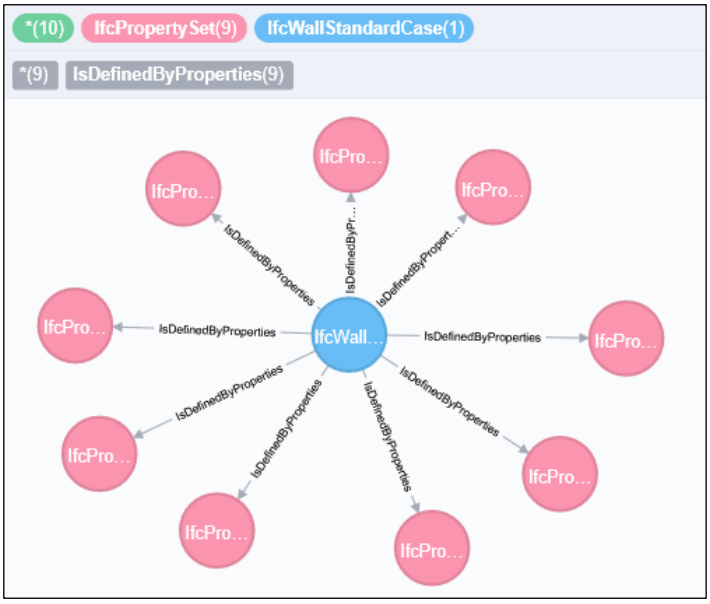


Figure 8 getting the assigned PropertySets for a certain object

- **Filter objects based on material:** This query returns a list of all doors and their material names. The connection between doors and material nodes is done automatically by the graph database engine (through IfcDoor–[*1..5]-IfcMaterial relation command) without the need to know how both IfcDoor and IfcMaterial are related to each other in the IFC schema.

```
MATCH (door:IfcDoor{Model:'Muster003.ifc'})-[*1..5]-
(material:IfcMaterial)
RETURN DISTINCT (material.IFCID)AS IFCID,(material.name)AS Material_name
```

- **Create new relationships to assign objects to layers:** This query creats the relationship “assignedItem” between defined layers in the model and their coresponding elements.

```
MATCH (n:IfcPresentationLayerAssignment{Model: "Office_A.ifc"})
UNWIND split(replace(replace(n.assignedItems,"(",""),")",""),",") as o
MERGE (assignedItems {Model: "Office_A.ifc",IFCID: replace(o,"#","")})
MERGE (n)-[:assignedItems]->(assignedItems);
```

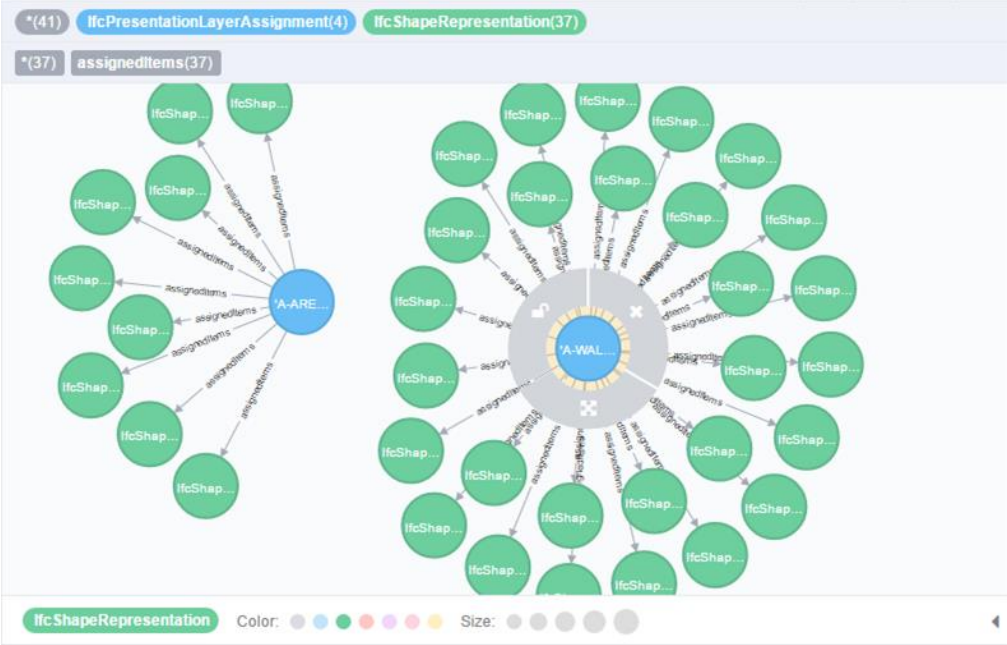


Figure 9 Assign objects to layers

5.2 Analysis of BIMs for Emergency Routes

In this example we used the graph database to do a simplified topology analysis of BIM models in order to analyse and generate emergency routes. The construction of possible escape paths is done in 2 steps:

(1) Through the relationship “BoundedBy” between IfcSpace and IfcDoor entities

(2) Through the relationship “RelatingSpace” between 2 IfcSpace neighbour objects.

```
MATCH p=(d:IfcDoor{IFCID:'9824',Model:'Muster003.ifc'})-[r:BoundedBy]-
>(s:IfcSpace)
RETURN p
UNION
MATCH p=(s1:IfcSpace{Model:'Muster003.ifc'})-
[r1:BoundedBy]-(d1:IfcDoor)-[r2:BoundedBy]-(s2:IfcSpace)
WHERE s1.IFCID > s2.IFCID
RETURN p
UNION
MATCH p=(s1:IfcSpace {Model:'Muster003.ifc'}) <-[r1:RelatingSpace]-
(r:IfcRelSpaceBoundary)-[r2:RelatingSpace]->(s2:IfcSpace)
```

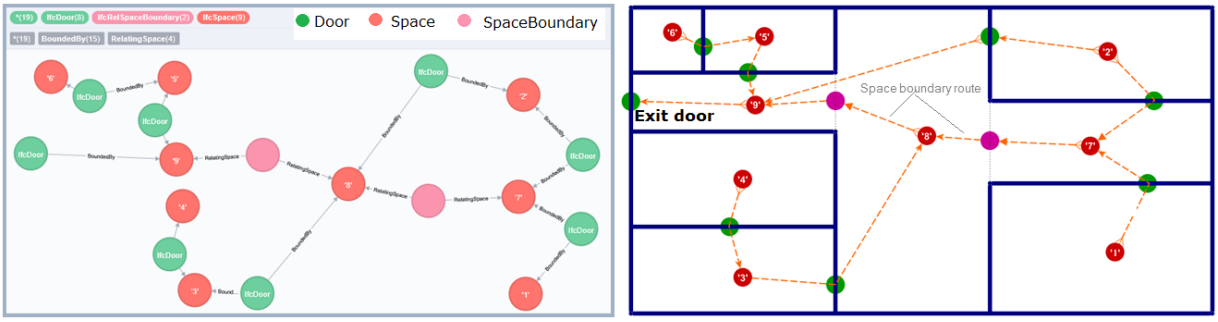
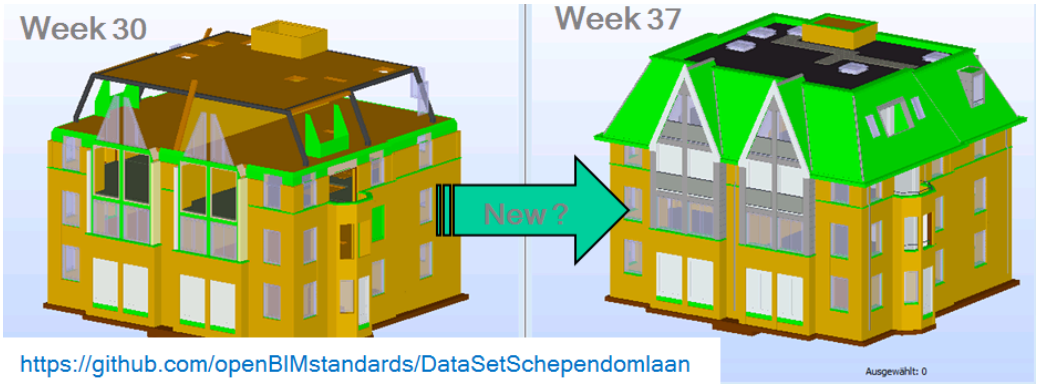


Figure 10 Creation of escape routes through graph query

5.3 Compare Different IFC Models

The following example compares 2 versions of IFC models for the same building and returns a list of all windows which have been added in the second model.




```
MATCH (W30 { Model: 'Week30.ifc' }),(W37 { Model: 'Week37.ifc' })
WHERE W30.tag = W37.tag
WITH collect(distinct W30.tag) as tags
MATCH (new) WHERE new.tag <> "" AND NOT new.tag  IN tags
AND new.label = "IfcWindow"
RETURN distinct new.label as Class, new.globalId, new.IFCID as
STEPID,new.name, new.tag ORDER BY new.tag
```

```
$ MATCH (w30 { Model: 'Week30.ifc' }),(w37 { Model: 'Week37.ifc' }) where w30.tag = w37.tag WITH collect(distinct w30.tag) as tags M
```

Rows | Text | Code

Class	GUID	STEPID	Name	Tag
IfcWindow	'2KsoxVt15Es8dH3L6KMVj0'	#108847	'dakkoepel 1000x_(769613)'	'281480A1-040A-48D1-A601-A1D6110BB331'
IfcWindow	'1aiPGZMJTBuOCj8nflqbyU'	#70771	'velux K08_(944800)'	'3461CF7D-134D-4C49-8212-00B3CB2FAA89'
IfcWindow	'0fexbeEOj7GxM6YwAainL9'	#293103	'dakkoepel 1000x_(1010566)'	'3EA47F39-44A3-44C7-A8F2-48A6D7B6F079'
IfcWindow	'2ZTlq5qx98Gg00NEGgsCu7'	#94339	'velux K08_(967504)'	'416FC727-C9B1-48B9-9170-B149BE80A18F'
IfcWindow	'0FbXSSF5b7$Or4awT$ablF'	#47203	'velux K08_(926256)'	'45C00328-2D0A-4E32-8D62-D9E365398AA8'
IfcWindow	'0y5r$haCX78Pq2BDkdcGi6'	#302403	'dakkoepel 1000x_(1018927)'	'481DE70F-AC2D-4599-864B-80BCD74EA453'
IfcWindow	'3OsF8vqm52GRcuICVSAR6F'	#290003	'dakkoepel 1000x_(1019167)'	'727B48D4-9217-4F38-984E-DDD4AFF7252C'
IfcWindow	'2PhZ0Ahyn399ojlU1JuiEf'	#23635	'velux K08_(948940)'	'E0CC4923-AC54-46D1-96DF-3F7628C03F8A'
IfcWindow	'1uSpy8O7X6qvqZOAWOaRLe'	#296203	'dakkoepel 1000x_(1019087)'	'EEB2ECF2-163A-4985-A831-5E9F2FFE50EB'
IfcWindow	'1TLojM7ir6OReaPEG9rR1A'	#299303	'dakkoepel 1000x_(1019007)'	'FAB3F4BB-3B1F-4E1D-98BE-3FDFD0EA66A1'

Started streaming 10 records after 205 ms and completed after 206 ms.

Figure 11 Compare IFC models using graph query

6. Conclusions and Future Work

This paper presented a workflow for automatic transformation of IFC schema and models into a property graph database. The suggested approach have been applied on many IFC schema/models in order to explore the capabilities of the IFC-based graph models for data query and advanced analysis of the BIM models.

The current scope of transformation and queries doesn't take in account the geometry information or the process of creating geometry objects based on parameters or Boolean operations. A future research is needed in order to include the geometry information to be a part of the pre-defined queries by developing an interface between the graph database and an IFC geometry engine.

Future research could involve developing a user friendly web application for editing the information of graph database and developing an IFC export interface for sub-models or merging models. The future work will include also developing special stored procedures which allow accessing the Java API of Neo4j directly and running the import and data retrieval queries much faster compared with solely using of Cypher commands and followed with a benchmark to compare the performance with other existing query approaches.

Graph models can be considered a satisfactory tool to manage building information. However, writing user-defined queries for users without IFC and graph skills still quite challenging where advanced queries should be written by IFC and graph DB experts. This research can be considered as an initial step in the direction of IFC-based graph models formation management.

References

Angles, R., & Gutierrez, C. (2008). Survey of Graph Database Models. ACM Computing Surveys, 40(1).

Doekemeijer, N., & Varbanescu, A. (2014). A Survey of Parallel Graph Processing Frameworks. Delft: Parallel and Distributed Systems Group- Delft University of Technology.

Hughes, J. (2016). ECE 3020 Mathematical Foundations of Computer Engineering. Atlanta: Lecture notes from Georgia Institute of Technology.

Isaac, S., Sadeghpour, F., & Navon, R. (2013). Analyzing Building Information using Graph Theory. International Association for Automation and Robotics in Construction (IAARC)- 30th ISARC, (S. 1013-1020). Montreal.

Ismail, A. (2011). *IFCWebServer*. IFC Data model server and online viewer: http://ifcwebserver.org

Khalili, A., & Chua, D. (2015). IFC-Based Graph Data Model for Topological Queries on Building Elements . American Society of Civil Engineers, Vol.29 Issue3.

Robinson, I., Webber , J., & Eifrem, E. (2015). Graph Databases. Sebastopol: O'Reilly Media.

Tauscher, E., Bargstädt, H.-J., & Smarsly, K. (2016). Generic BIM queries based on the IFC object model using graph theory. The 16th International Conference on Computing in Civil and Building Engineering. Osaka,Japan.

Wilson, R. (1996). Introduction to Graph Theory (Fourth edition Ausg.). Harlow- England: Longman Group Ltd.