A SEMANTIC GRAPH DATABASE FOR BIM-GIS INTEGRATED INFORMATION MODEL FOR AN INTELLIGENT URBAN MOBILITY WEB APPLICATION

A.E.Hadi Hor[1], S. Gunho[1*], P. Claudio [2], M. Jadidi[1], A. Afnan[1]

[1] Earth and Space Science and Engineering at York University, 4700 Keele St, Toronto, ON M3J 1P3, Canada
(elhadi, gsohn, mjadidi, aafnan) @yorku.ca
[2] Computer Science and Engineering at York University, 4700 Keele St, Toronto, ON M3J 1P3, Canada
pio@cse.yorku.ca

Commission IV, WG IV/4

KEY WORDS: Property, graph, database, model, IFC, CityGML, BIM, GIS, Neo4j, Web GIS Services, SPARQL

ABSTRACT:

Over the recent years, the usage of semantic web technologies and Resources Description Framework (RDF) data models have been notably increased in many fields. Multiple systems are using RDF data to describe information resources and semantic associations. RDF data plays a very important role in advanced information retrieval, and graphs are efficient ways to visualize and represent real world data by providing solutions to many real-time scenarios that can be simulated and implemented using graph databases, and efficiently query graphs with multiple attributes representing different domains of knowledge. Given that graph databases are schema less with efficient storage for semi-structured data, they can provide fast and deep traversals instead of slow RDBMS SQL based joins allowing Atomicity, Consistency, Isolation and durability (ACID) transactions with rollback support, and by utilizing mathematics of graph they can enormous potential for fast data extraction and storage of information in the form of nodes and relationships. In this paper, we are presenting an architectural design with complete implementation of BIM-GIS integrated RDF graph database. The proposed integration approach is composed of four main phases: ontological BIM and GIS model's construction, mapping and semantic integration using interoperable data formats, then an import into a graph database with querying and filtering capabilities. The workflows and transformations of IFC and CityGML schemas into object graph databases model are developed and applied to an intelligent urban mobility web application on a game engine platform validate the integration methodology.

1. INTRODUCTION

Industry foundations classes (IFC) and City Geographic Modelling language (CityGML) are both standards for encoding 3D models and their integration can open opportunities for applications in a broad range of areas such as urban planning, facility management, seamless indoor-outdoor routing, and in the environmental simulation (Ismail, A., Nahar, A., & Scherer, R. (2017)) and many more other applications. Even that, these two technologies have evolved differently, both can benefit from each other if they could exchange data effectively. As BIM technology is mainly focused on indoor environments, GIS can extend the benefits and applicability of existing building models to the outdoor environment, this is not an easy task to transfer no to exchange data between BIM and GIS without consideration of data format and meaning. Current state-of-the-art BIM (or GIS) tools enable the data exchange between the systems by using a common data format (Khalili, A., & Chua, D. 2015). Therefore, the users can access data from a different software program and share data within the BIM (or GIS) domain. However, it requires the user to have a thorough understanding of both systems and their functionalities. The integration tools and current standards lack the ability to help the user to convey meaning. To fully integrate GIS and BIM, there is a need to provide interoperability at the semantic level.

Given the fact that industry foundation classes (IFC) files and City Geography Mark-up Language (CityGML) both have capabilities for storing semantic properties and relationships between objects and elements, these models can be integrated into one information model the Resource Description Framework (RDF). This modern ontology language framework with web ontology language (OWL) are based on description logics. As description logics describe the domain in terms of concepts, roles, and individuals, OWL describes that in terms of classes instead of concepts, properties instead of roles, and individuals. A collection of these classes, their attributes and relationships can be stored as RDF statements or triples describing each individual resource, it's properties, and values which can be intuitively understood as a graph that can also be represented as nodes representing classes and connecting vertices representing relationships and values (A. Hadi. Hor, Jadidi, A., & Sohn, G. (2016). In this study we have investigated, designed, and developed a graph database framework for BIM-GIS RDF integrated model to take advantages of the powerful performance, flexibility and agility properties of graph databases and explore BIM-GIS integration patterns through large-scale graphs analysis and real-time applications like intelligent urban mobility.

2. METHODOLOGY AND CONCEPTUAL FRAMEWORK

The main purpose of this research work is to develop and implement a conceptual design for a BIM-GIS integrated information model using their IFC and CityGML source datasets using their semantics and geometric information represented in ontological taxonomies and graph database. The end-to-end system implementation for a semantic graph database for BIM-GIS integrated information model for intelligent Urban Mobility web application is mainly composed of four modules as shown in the diagram illustrated in Figure 1.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-IV-4-89-2018 | © Authors 2018. CC BY 4.0 License.

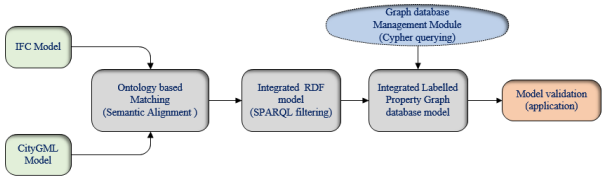


Figure 1. BIM-GIS semantic integrated model conceptual design

These four modules are described the section below

- **Source Models:** These are the BIM and GIS models expressed respectively in IFC and CityGML native formats exported into RDF representations for model integration.
- **Data translation and graph matching module:** the IFC and CityGML datasets are matched and aligned using a graph Matching algorithm for ontologies (GMO) algorithm, before migration into an integrated RDF graph model, to be then exported into a labelled property graph (LPG) datasets stored in a graph database management system.
- **BIM-GIS integrated Model management:** this module provides sets tools granting access to administration and development using the graph database.
- **Application:** an application was developed using the graph database to validate the BIM-GIS graph database.

The new proposed methodology introduces a workflow to develop and build IFC and CityGML based graphs models using their RDF ontological models developed (A. Hadi et al 2016), in this workflow the transformation is described in Figure 2.

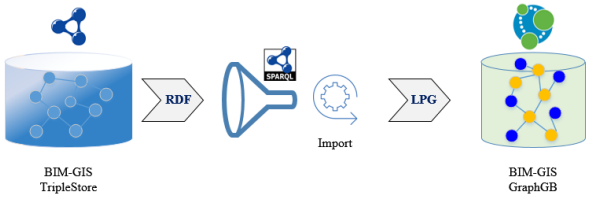


Figure 2. Transformation pipeline

The process of getting data from **R**esouce **D**escription **F**ramework (RDF) data into graph database can be completed by transforming the BIM-GIS integrated RDF model into CSV format. The CSV files can feed visualizations platforms and with its propositional form can be suitable for most data mining algorithms . queries were designed and executed to extract semantically mapped and aligned using semantic alignment techniques triples (A.Hadi et al., 2016) from the BIM-GIS RDF integrated model, before been exported to tabular view. The resulting RDF data can be then serialized to a CSV as specified in the W3C recommendation *https://www.w3.org/TR/sparql11-results-csv-tsv/*.

Once completed a data quality check is performed with a number of tools to help validating the imported dataset. In the course of this research we used a number of validation tools: mainly, the *CSVkit* a set of python tools that provide statistics on data, these statistics can allow inaccuracies detection, also we took advantage of LOAD CSV clause from Cypher, this powerful ETL and conversion tool can direct mapping of large data into complex graph/domain structure, merge data from heterogenous sources, their relationships structures and supportes well heavy computations.

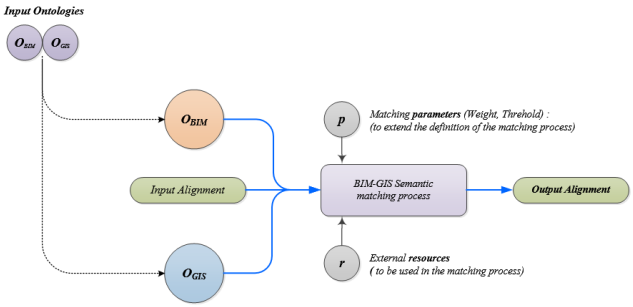


Figure 3. Semantic Alignment workflow of BIM-GIS integrated ontologies (A. Hadi. Hor, Jadidi, A., & Sohn, G., 2016).

Resource description Framework (RDF) triple stores and property graphs both have ways to explore and graphically depict connected data and provide methods to explore and query integrated data models. However, there are differences, and each has strengths when it comes to represent BIM and GIS data. This study is looking to combine the RDF and Property graphs to develop a graph data model able to integrate these two domains of knowledge that can provide the highest level of accuracy and data richness and with sets of admin tools allowing graph data management, however knowing that:

- *RDF does not uniquely identify instances of relationships of the same type: no two connections of the same type between a pair of nodes.*
- *RDF cannot qualify instances of relationships: no attributes on relationships.*
- *A property graph of **N** nodes, **p** properties/node, **r** relationships/node, **l** label/node, in RDF it will contain: **(p + r + l) * N** triples.*

A graph is a data structure composed of edges and vertices. A graph database technology is an effective set of tools to model, store, manipulate, query and manage graph data focusing on the attributes, values and relationships between the entities in the data model. Modeling objects such BIM IFC classes and GIS CityGML elements and the relationships between them means that everything can captured and represented in a corresponding graph. Graph databases make an extensive use of graph entities such as properties, nodes and edges (relationships). Data is mainly stored in nodes while the relationships between the data items is represented by edges between nodes. Graph databases are now a viable alternative to relational database systems and it has its applications in many fields in science and engineering such as biology, chemistry, social networking, semantic web and research engines.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-IV-4-89-2018 | © Authors 2018. CC BY 4.0 License.

3. GRAPH DATABASE MODELS

There are many supported graph types that can be utilized when it comes to modeling and bringing GIS and BIM models together into an integrated graph, one of these graph models is the property graph. Property graphs are attributed, labeled, directed multi-graphs provides a visual example of a property graph which represents interactions between people and objects. A benefit to the multi graph is that it is the most complex implementation because every other type of graph consists of subsets of the property graph implementation. This means a property graph can effectively model all other graph types. The graph database is optimized for the efficient processing of dense, interrelated datasets. This design allows the construction of predictive models, and detection of correlations and patterns. This highly dynamic data model in which all nodes are connected by relations allows for fast traversals along the edges between vertices. A benefit is the fact that traversals are localized and do not have to consider sets of unrelated data.

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*

Before deciding on the Graph database to use in this research work, we have conducted literature review of current graph databases along with testing and evaluations, the comparison is typically done by using a set of common features or/and by defining general model as a comparison base, the evaluation used and presented in this section is oriented to evaluate the data model provided by each graph databases, in terms of data structure, query language and integrity constraints.

3.1 AllegroGraph: is a persistent Resource Description Framework (RDF) graph database, it uses disk-based storage, enabling it to scale very well while maintaining high performance. AllegroGraph supports SPARQL, RDFS++, and provides a Representational State Transfer (REST) protocol architecture and can be used for geo-temporal reasoning and social networking analysis.

3.2 DEX: this is a High-performance and scalable graph database and mostly used in NoSQL applications, has an architecture of three layers and a query engine. The API provides application interface and the applications layer is used to extend core capabilities.

3.3 VertexDB: this is another High-performance graph database server that uses the HTTP protocol for requests and Java Script Object Notation (JSON) for its response data format.

3.4 Infinitegraph: A distributed object database with C++, java, C#, python. Infinitegraph is built on a highly scalable, distributed database architecture where both data and processing are distributed across the network. Infinitegraph can handle large transactions efficiently.

3.5 Hypergraphdb: An extensible, portable, distributed, embeddable, open-source data storage mechanism used mostly for knowledge representation, Artificial Intelligence (AI) and semantic web projects, it can also be used as an embedded object-oriented database for Java projects of all sizes.

3.6 Sones: Has a modular and well-designed for graph-oriented database logic, it provides an interface for using the database object efficiently. This graph database provides support for high-level data abstraction concepts for graphs and has own query language and an underlying distributed file system.

3.7 Infogrid: An open source java-based web graph database and web-oriented functions to web applications. it can be used as a standalone graph database or in addition to the other projects like user interface RESTful maps and content of a Graph Database to browser-accessible allowing developers to define individual objects and sub-graphs rendering.

3.8 FlockDB: A distributed graph database for storing adjacency lists, it is simpler than other graph databases. It scales horizontally and is designed for on-line, low-latency, high throughput environments such as web-sites.

3.9 Trinity: A graph database and graph designed to support large graphs, it has a set of utility tools and management tools built to be an efficient graph store and online query processing.

3.10 OrientDB: An open source NoSQL database management system to support schema-less, schema-full and schema-mixed modes and SQL as a query language. It uses indexing algorithm providing both fast insertions and ultra-fast lookups. It is also a transactional database and web ready, supporting HTTP, RESTful protocol, and JSON.

3.11 G-store: this graph database exploits the structure of the graph to derive a data placement optimization on disk for access patterns found in graph queries. It has its own built-in query engine that supports depth-first traversal, reachability testing, shortest path search, and shortest path tree search.

3.12 Cloudgraph: This is fully transactional .NET graph database providing a fast and scalable graph database that is both easy to deploy and maintain. It traverses graph with graph query language (GQL) and supports schema-less hypergraphs and key-value pairs. It is used for managing mainly web data.

3.13 Bigdata: this is an open source graph database. The Bigdata architecture provides a high-performance platform for data intensive distributed computing, indexing, and a high-level query on commodity clusters. It also provides SPARQL query language for fast load and query.

3.14 Neo4j: A high-performance NoSQL graph database with all the features of a mature and robust database Neo4j is the

most popular graph database, it is particularly developed for Java applications, but it also supports Python. Neo4j is an open source project available to public. The graph model in Neo4j consists of (1) Property (key- value pair) can be added to both node and edge; (2) Only edges can be associated with a type, (3) Edges can be specified as directed or undirected. Neo4j uses an index mechanism. Neo4j implements an object-oriented API, a native disk-based storage manager for graphs, and a framework for graph traversals with many features such as:

a) ***Intuitive**: to use graph model of data representation.*

b) ***Reliable**: fully transactional and upholds the database ACID properties.*

c) ***Durable and Embeddable**: using a custom disk-based and native storage engine with few java jar files.*

d) ***Massively scalable**: up to billions of nodes/relationships and properties.*

e) ***Highly available**: when distributed across machines*

f) ***Expressive**: with a powerful, readable declarative graph query language.*

g) ***Fast**: with powerful transversal framework for high speed graph queries.*

h) ***Simple**: accessible through REST API interface or any other object-oriented Java API.*

i) ***Indexed**: indexes are based on apache Lucene, and supports secondary indexes.*

j) ***Cross platform**, straight forward setup and configuration, open source and well documented.*

k) ***GPL** (General Public License) for community, and AGPL for enterprise and developers.*

l) ***Use Cypher**: is a declarative graph query language that allows for expressive and efficient querying and updating of property graphs.*

In this paper we are using the Neo4j Graph database to create, manipulate and maintain massive IFC and CityGML graphs. Neo4J is a native, high performance graph database built specially for storing and processing graphs. It takes advantage of connections between data stored in nodes and edge, thus speeding up queries by ignoring data that are not connected to relevant nodes. It is a fully ACID (*Atomicity, Consistency, Isolation, Durability*) transactional database. Neo4j stores graph data items in properties, nodes and relationships.

a) **Properties:** Flat data are stored in properties, every property must exist inside a node or a relationship, where they are uniquely identified by their respective names, meaning that no two words, no two properties have the same name in a node or relationship, properties in graphs correspond to column values in an equivalent relational database.

b) **Node:** Nodes are central entities in Neo4j. A node can contain an arbitrary number of labels and properties. Labelled nodes are indexed and be retrieved using one of their assigned labels (schemas indexing). Nodes are linked together by relationships. Graph nodes correspond to rows in entity tables from an equivalent relational database.

c) **Relationship:** Relationships connect nodes together. In Neo4j graph database, relationships are directed and thus point from a start to an end node. However, relationships can be traversed in both direction. Each relationship has type (like a node) and can contain an arbitrary number of properties. Graph relationships correspond to foreign keys in an equivalent relational database.

The simplest possible graph is s single **Node**, a record that has named values referred to as **Properties**. A node could start with a single property and grow to a few millions, it makes sense to distribute the data into multiple nodes organized with explicit relationships.

For performance and maintenance purposes Neo4j introduced schema for graphs consisting of indices and constraints

Index: To enable efficient querying of graph data, Neo4j creates redundant copy of graph entities and stores it is a database storage, this copy is called **index.** An index can be created automatically for properties of all nodes of the same label (schema indexing) or manually for nodes of different labels (legacy indexing).

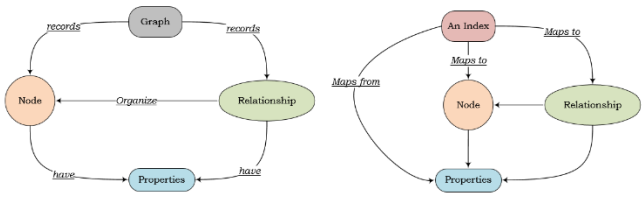


Figure 4. Graph node and relationship.

The drawbacks of indices are the additional required storage and slower disk read/write, therefore, indices should be only used when the number of queries requested on a given property exceeds that of modification operations.

Constraint: Another aspect of graph schema is constraint on nodes and relationships. Neo4j allows unique and existence constraints on nodes properties as well as existence constraints on relationships properties, in case of unique constraints, an index is implicitly created for affected data.

4. BIM-GIS INTEGRATED GRAPH DATABASE

4.1 IFC Graph Model

The standard Industry Foundation Classes (IFC) defines a rich object model for describing and sharing building models. In addition to the modeling of building components, the relationships between those are of peculiar interest. These relations are described by pointer structures between components. These models can only be insufficiently mapped to relational databases because of their heterogeneous structure. Therefore, graph databases will be tested for their ability to efficiently store of IFC models. In addition, the usefulness of the provided database query system is examined.

The IFC model is composed of IFC entities built up in a hierarchical order, where each IFC entity includes a fixed number of IFC attributes, with 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 as part of the IFC standard code (Ali Ismail and al., 2017).

The IFC data schema has three fundamental entity types described and shown here:

a) **Objects** are generalization of anything (or item)

b) **Relations** are the generalization of all relationships among things (or items)

c) **Properties** are the generalization of all characteristics (either types or partial types i.e. property sets) that may be assigned to objects.

That corresponds to the one of the following classes (Figure 5)

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.*

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.*

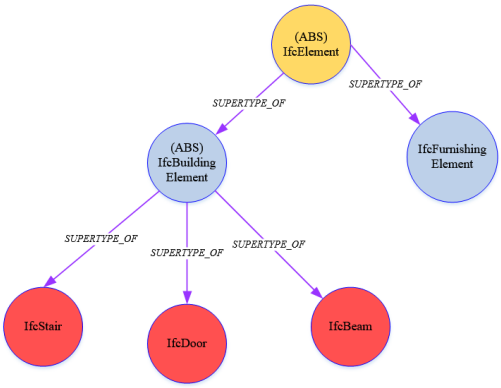


Figure 5. IFC IMG diagram

The BIM models contains huge amount of information and complex relationships between their elements, this information could remain inaccessible due to lack of suitable data management thus converting BIM models into an effective information retrievable model based on property graph database could significantly facilitate the exploration and analyzing the BIM highly connected data workflows, for an automatic transformation of IFC schema and IFC models into IFC object graph database, though RDF transformation then to graph data model using graph theory concepts to explore, manage and analyze all information inside the BIM models by:

1. Running queries for information retrieval and data mining.

2. Explore topology analysis of the model Integrated framework for BIM models and linking with any other additional project related information.

Knowing that RDF graph is a set of triples *(Subject, predicate, object)* where both the subject and the predicate are resources and the object can be either another resource or a literal, the only particularity about literals is that they cannot be the subject of another triple, in a tree structure they are the *nodes* and they are uniquely identified. To map RDF model into a graph database model we need to set the following rules:

- ***Rule 1:*** *Subjects of triples are modes to nodes in Neo4j GraphDB, the node representing an RDF resource will be labeled resource.*

- ***Rule 2:*** *Predicates of triples are mapped to nodes properties in Neo4j GraphDB if the object of the triple is a literal*

- ***Rule 3:*** *Predicates of triple are mapped to relationships in Neo4j GraphDB if the object of the triple is a resource.*

Given that IFC source model is an EXPRESS based entity-relationship model while EXPRESS is a standard data modeling language for product data, IFC is best for representing a model that is a collection of objects related to each other. Therefore, using this model as a data source for our BIM-GIS integrated information model, the resulting model should enable applications to retrieve specific objects based on their relation to other objects in the model. The relations between IFC entities in the graph database can be traced back to the connections found within a building and can be extended to its surrounding when CityGML elements are within the extended query.

4.2 CityGML Graph Model

CityGML is an open source data model and XML based format for the storage and the exchange of 3D city models, it is an application schema for the Geography Markup language version 3.1.1 (also known as GML3), the extensible international standard for spatial data exchange issued by OGC (Open Geospatial Consortium) and the ISO TC211. Most of common city objects and features, such as buildings, water, vegetation, traffic, terrain, bridges, etc. can be all described in CityGML. CityGML is capable not only for including 3D geometry and graphical appearances but also 3D topologies and semantic properties of city objects and features in five levels of details, namely from LOD0 to LOD4.

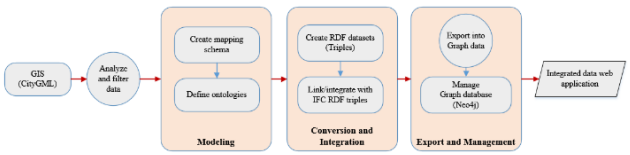


Figure 6. CityGML elements translation and export process

Knowing that even SPARQL query language can perform many kinds of queries on CityGML triples, it does not have spatial queries implemented, therefore we had to use GeoSPARQL in some of workflow transformations. GeoSPARQL defines vocabularies for representing geospatial data in RDF and provides an extension tools to SPARQL for processing Geospatial data, it allowed us to define geometric data in RDF triple-store and perform spatial queries and analytics, an example to of a GeoSPARQL to Find all features that feature my:A contains, where spatial calculations are based on *my:hasExactGeometry*. These types of queries are imbedded with other queries from BIM triples to extract and output a set of RDF triples to feed the application as a source data, we can notice

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-IV-4-89-2018 | © Authors 2018. CC BY 4.0 License.

simplicity and the flexibility of designing queries based on application requirements.

4.3 IFC-CityGML Integrated Model

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 like simple graphs; consist of nodes and relationships which are often expressed as vertices and edges. However, labelled property graphs provide additional characteristics to facilitate graph understanding, where, nodes could have a single or multiple label; 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. By using RDF graph as a common framework for the description of BIM and CityGML related elements, it is possible to use query language SPARQL from semantic web technologies to retrieve specific information from the integrated model. The proposed methodology introduces a workflow to develop and build an integrated information graph model using a transformation pipeline from IFC and CityGML RDF schemas, in this workflow the translation is semi-automatic and done achieved through a succession of processes from RDF representation to graph database schema as shown in the diagram below (Figure 11).

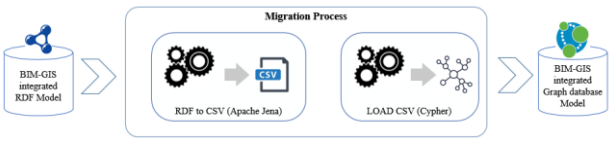


Figure 7. IFC-CityGML to graph database (Neo4j) conversion workflow

This approach a single graph Neo4j database to store various IFC and CityGML models integrated into one database model where each node (object) has attributes from either IFC or CityGML or from both resulting from the RDF integrated model created before extraction and import into Neo4j described in Figure 8, this way queries and analytics are not limited to one source model, this will allow to merge, and integrate different architecture, Engineering and construction (AEC) models from BIM with Geospatial domain of knowledge. By using semantic representation as a common framework for the description of BIM and GIS objects, it is possible to use semantic web technologies (A.Hadi. Hor, Jadidi, A., & Sohn, G., 2016). such as query language SPARQL to execute queries of IFC elements associated with corresponding elements from CityGML based on pre-defined vocabularies, relationships from integrated ontological model.

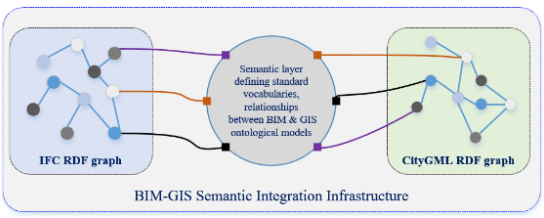


Figure 8. BIM-GIS integration infrastructure (ontological model)

Once the objects of the integrated graph model are imported into graph database, the attributes of the similar objects will be enhanced with each other attributes, topological and geometrical properties as described for the ifcSlab from BIM) and Slab from GIS seen in the Figure 9.

The mapping of IFC and CityGML into Meta Graph Model representing all IFC classes and CityGML elements, their attributes and relationships is completed through cypher scripts with LOAD CSV clauses and commands to validate data quality. Once all classes and attributes are mapped into nodes and connected using various relationships like ‘*has_property’* to connect a class node with direct attributes or ‘*subtype_of’* to connect the class with its corresponding element in CityGML and with its own sub-class resulting into a complete integrated graph model with all classes and elements stored a unified property graph in which the relationships for referenced, inversed and derived attributes for IFC classes will be created automatically including geospatial relationships from CityGML elements, the next step will consist in creating relationships between graph nodes and their connected information and dropping redundant relationships in the unified integrated model and using property normalization and non-direct attributes of BIM classes and GIS elements of the same object can be assigned as direct nodes attributes, further each node (object) can be assigned a set of labels of its parent class/element, them running pre-defined queries for the purpose of classification and normalization of objects representing building elements with accordance to their object type and property sets.

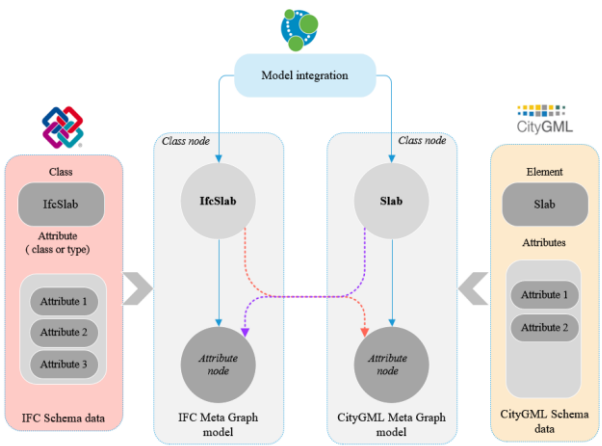


Figure 9. Enhanced attributes of BIM-GIS information using Graph Database

There are two aspects of interacting with Neo4j graphs querying datasets and manipulating graphs, for that purpose Neo4j provides its own declarative query language named Cypher and bolt protocol (Java Core API), in this paper we focused on using Cypher which is a declarative graph query language that allows for expressive and efficient querying and updating of property graphs, it is relatively simple and powerful query language. Complicated graph databases queries can easily be expressed through Cypher, However, different keywords such as MATCH, WHERE and RETURN are used to from queries, these are some of the examples to replace attribute and relations in a BIM class.

5. SYSTEM DESIGN ARCHITECTURE AND IMPLEMENTATION

The system architectured for a complete BIM-GIS integrated semantic model from a RDF representation to a graph database platform on Neo4j is represented in the diagram in Figure 10. It consists of four modules (1) Data sources consisting of BIM and GIS models in IFC and CityGML formats respectively, (2) Data translation from native formats followed with a RDF graph matching using Sematic Alignment Techniques, (3) BIM-GIS integrated model management console and providing a user interface to store, maintain and manage Graph datasets, and (4) Application : this is a web based application hosted using web based services hosted on a cloud platform.The datasets used are BIM and GIS models which we have pre-estimated, the amount of triples and number of nodes of each of the models was tested in this implementation.

After importing the integrated model into Neo4j graph database, a management console (also available as a web service) can be used to run queries for data retrieval, advanced analysis and graph data mining algorithms of BIM-GIS integrated graph model, these queries could be ranging from complex queries like the ones used for facilities management, emergency evacuation, evaluating inventories, cost estimation, and shortest path to the simplest like finding element’s information and indoor navigation paths.

Using Neo4j REST API import/export capabilities of graph data to multiple file format such as JSON format which has been used as an input facilitating connection to the BIMServer (http://bimserver.org/), using BIM Server JSON API (https://github.com/opensourceBIM/BIMserver/wiki/JSON-Queries). Through BIM Server, the data of the integrated semantic model represented in JSON file format that can be streamed to any of these file formats 3D Colloda, ifcXML, KML, etc. or as web REST or SOAP services such as REST or SOAP services, for the sake of this proof of concept, the model is streamed out to an IFC based schema for the purpose of comparison to check the number of the classes that have been carried out during the process workflow and to extract the multipatch geometries used in re-constructing and creating a scene Layer Package (SLPK), which is an optimized data format for visualizing a large amounts of three-dimensional data allowing the new model to be published on ArcGIS cloud platform as a web scene layer on ArcGIS Online cloud platform that can also be embedded with other features layers from ArcGIS Server. The resulting web services are then consumed into a Unity3D game engine application to help simulate an intelligent urban mobility (as shown in Figure 12), which beside the ability to identify individual objects from BIM and GIS, the application can also extract and filter combined attributes and metadata information that can be extracted and filtered from the integrated BIM and GIS domains such as the attributes of the Door object (Figure 12.c) where the information was originated from BIM and GIS.

6. EVALUATION AND CONCLUSION

In this paper, we discussed and presented a complete study with a complete implementation of an integration pipeline of integrated BIM-GIS semantic model. an approach to combine data models and instance data from BIM and GIS domains to provide a comprehensive graph data model with data analysis and mining capabilities using a graph database platform. The methodology and concepts were illustrated by the representation of a building model in both IFC and the CityGML and their translation with linkage into a unified integrated semantic model by applying semantic web technologies (RDF representation) then an extraction and import into a graph database system using SPARQL and Cypher scripting providing a promising future of data and application integration methods by adding more data sources by handling properties, attributes and relationships very effectively. In a semantically oriented integration based on RDF datasets and graph databases as presented in this research we applied many IFC and CityGML schemas and models to explore all capabilities of CityGML and IFC based graphs for advanced analytics and data filtering of a unified AEC-Geospatial models.

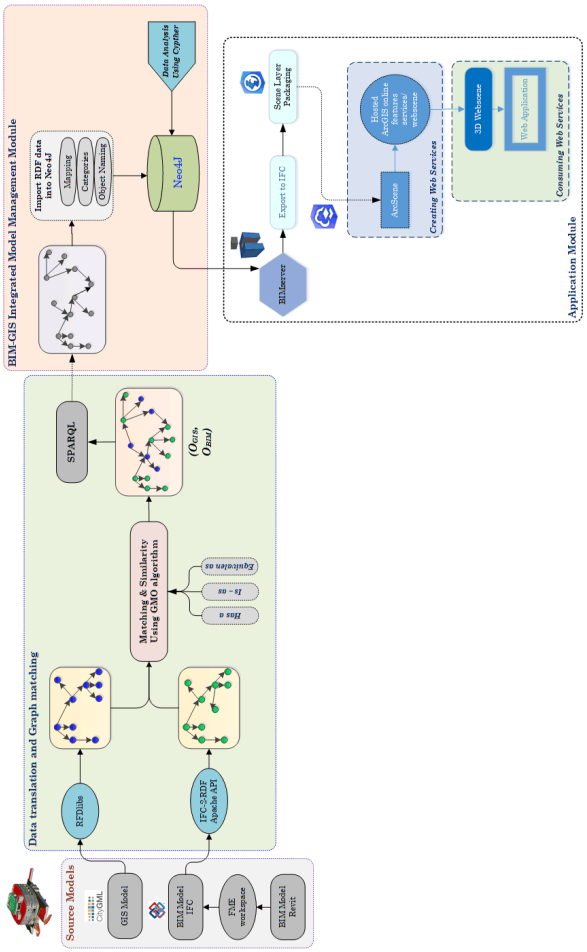


Figure 10. System design architecture

In our future research works we will target developing more cross-platform applications and web application for editing BIM-GIS information stored and managed in the graph databases and creating procedures in extracting graph datasets that can be exported as sub-models or merging models, exploring an cloud based web services integration based architecture using specified API to provide scalability, collaboration, flexibility and automatic, instant updates to models and supporting hardware and software resources. We also would like to investigate performance evaluation of queries execution plans metrics when running data import and data retrieval queries to make them much faster compared with solely using of Cypher commands and followed with a benchmark to compare the performance with other existing query approaches and we will also investigate developing tools in the graph database to validate geometries coming from any source data.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-IV-4-89-2018 | © Authors 2018. CC BY 4.0 License.

A semantic based BIM-GIS integration using graph data modeling tools as developed and presented in this work are considered great for managing information and creating workflows involving BIM and GIS components, however it will need deep understanding not only of BIM and GIS domains of knowledge data structure and schema characteristics but also graph mathematics, graph databases structure and tools therefore advanced data queries have to be written graph databases experts with help from BIM and GIS practitioners.

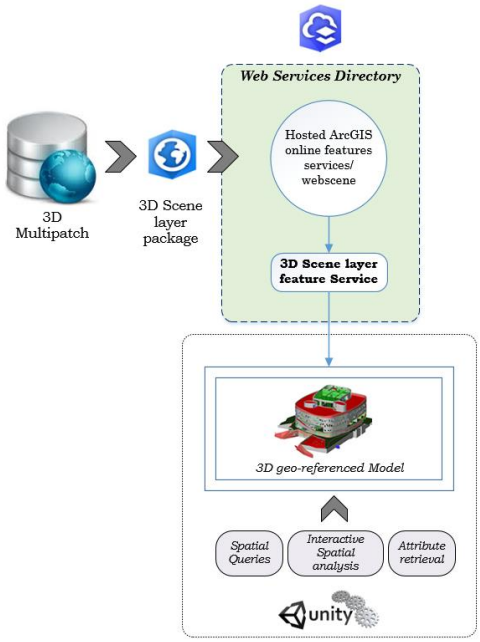
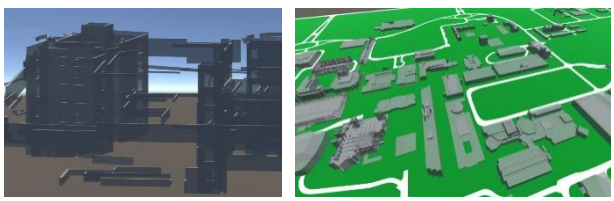
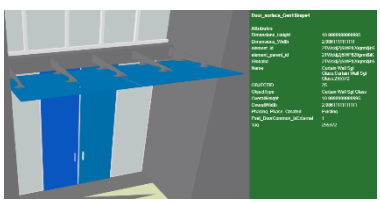


Figure 11. Authoring, publishing and consuming web service on Unity3D game Engine



(a) (b)



(c)

Figure 12. Outputs of BIM-GIS integrated model on Unity3D.

(a) Bergeron building only (b) York University campus

(c) attributes of Door object.

REFERENCES

A.Hadi. Hor, Jadidi, A., & Sohn, G. (2016). Bim-Gis Integrated Geospatial Information Model Using Semantic Web and Rdf Graphs. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *III-4*(July), 73–79. https://doi.org/10.5194/isprsannals-III-4-73-2016

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

Ismail, A., Nahar, A., & Scherer, R. (2017). Application of graph databases and graph theory concepts for advanced analysing of BIM models based on IFC standard. *24th International Workshop on Intelligent Computing in Engineering (EG-ICE 2017), At Nottingham, UK*, (July), 1–12.

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.

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