A Simplified BIM Model Server on a Big Data Platform

Wawan Solihin, wawan.solihin@gatech.edu
Invicara Pte. Ltd., Singapore

Charles Eastman, charles.eastman@coa.gatech.ac.edu
College of Architecture, Georgia Institute of Technology, United States

Abstract
Today's BIM models are increasingly complex and large. There are at least two trends that will make BIM several magnitudes larger in size. The two trends are the integration of building sensor data that can be used to optimize the operations of the building, and the increasing demand to consolidate multiple projects into one easily accessed model server to apply various analyses for comprehensive urban planning or design taking into account the impact of adjacent buildings. This pattern of data fits the characteristics of a big data. A simplified schema BIMRL that has been developed in a relational database environment is proposed and adapted for an efficient use of the big data platform for BIM. The adapted schema allows a centralized collection of all building models in one database. This research explores suitable forms of the big data platform for BIM, challenges in implementing them, and the potential benefits of such approach. As a proof-of-concept, an ETL process is used to transform the relational database BIMRL data into the multi-mode NoSQL database OrientDB. Sample queries are shown to compare how such schema works by comparing it with the original relational based BIMRL.

Keywords: BIM, Big-data, Query-able Building Model Database

1 Introduction
With the wide acceptance of BIM in our industry, researchers have begun looking into the future of BIM. Besides various applications using BIM that increasingly looks into the lifecycle support of buildings, there are also two trends in BIM that will require ability to support much larger model, collection of models, and increasing amount of data being collected. The first trend concerns integration of multiple buildings into one integrated collection for the purpose of better building operation and maintenance for owners or integrating buildings into a city-wide collection for various analyses [1], and the other is the increasing popularity of laser scan point cloud data for existing buildings and sensor data for use in optimizing the operation of the buildings [1, 2]. In both cases, there is a need to develop a strategy to store and access large data efficiently. This problem fits nicely into the big data problem. The general consensus of the definition of the big data is given by Gartner: "Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation". This definition was coined in 2001 in a research report by the META group (now Gartner) and has become known as the 3V's [3]. The simpler definition of it is that data which is too big, too fast, or too hard for existing tools to process [4].

Armed with the increasing popularity of the big data topic and the availability of tools to deal with the problem, this research looks into what would be needed for BIM, in particular, to address the above trends using the big data platform. It also seeks to identify limitations and other issues with the current tools before such use is ready for practical use. The rest of the paper is organized in the following manner, first, we will present a short discussion about the current BIM. This is followed by the trends for the future BIM in the area of data management and access, and what requirements that must be fulfilled to meet the demand for the future BIM. Next, we will describe a brief overview of the big data platform and the various flavors of the tools. We then look into what should the future BIM look like if it makes use of the big data platform to address the requirements. A brief discussion on the prototype implementation on the selected platform that fit the need for the future BIM will be covered in the next section. From the experimental tests, we will discuss the challenges and opportunities for the future BIM in the big data platform, followed by conclusions and future works.

2 The Current BIM

With the increasing awareness of the value of BIM in bringing improvement to the quality of buildings and to reduce cost, the use of BIM has been increasingly widespread. The many initiatives by various governments around the world to push for the mandatory use of BIM are the testament to this [2, 3]. There are several issues with BIM currently that still need works. Among those are the integration issue [4] and access performance. Model servers are supposedly the answer to these issues. Using a model server, models from various disciplines or various versions can be put together into one repository and accessed using a uniform interface. IFC is the standard format in many of the model servers, with the exception of those that are part of a single vendor ecosystem such as Revit server (Autodesk), or Graphisoft BIMCloud. BIMServer is increasingly popular since it is an open source project supporting IFC [5]. EDM Model server is another popular but commercial offering also using IFC [6]. There are also several model server experiments using relational database [7], and object-relational database [8]. All these model servers map IFC entities into their internal structure one-to-one. Because of the inherent complex structure of the IFC schema, a one-to-one mapping of IFC to the database means the use of many tables, one for each IFC entity. In IFC2x3, there are 653 entities and 327 types (117 defined types, 164 enumeration types, and 46 select types). In IFC4, the numbers have grown to 766 entities and 391 types. Thus, creating a 1-to-1 mapping to database tables clearly imposes a heavy burden on the relational model. It also requires a large number of tables in a join statement just to be able to access even relatively simple data. The select type also causes additional complexity for queries as it has to be determined during runtime before further queries can be made to the data. The approach also is insufficient to handle geometry since IFC geometry information is suitable as an “instruction” to reconstruct the geometry, and therefore is not ready for spatial queries.

3 The Future BIM

3.1 Current trends

With the ever increasing awareness of sustainability, designs, constructions, and operations of buildings will need to improve not just evolutionary but by a big leap. In order to achieve this, a lot more data as input to the construction process and operations will need to be handled to allow to derive intelligent information for use in the entire process. It includes awareness of the surroundings, such as the GIS, weather, and adjacent buildings data. All these are required at the design phase. Further on during the construction phase, all the construction activities, logistics, movements of people and materials, and construction progress will further feed a large amount of data. In the operations phase, sensor data from building automation systems and other types of sensors may provide a good input for efficient building operations and comfort to the occupants. A result of it will be a large amount of continuous stream of sensor data. All these show that at least two key concepts are critical, i.e. integration of data from various sources, and potentially a large amount of data that need to be handled.

The rise of popularity of the big data platform provides a good framework and tools suitable for handling a large amount of data. However, there are many factors to consider before a suitable platform can be selected especially in the current climate where there are a plethora of options to choose from and seemingly continuous proliferation of tools popping out regularly in the recent years.

3.2 Requirements

To enable the right decision in choosing the appropriate platform, the requirements must be first established. The two broad concepts that need to be supported as discussed above are the integrated data and handling a large amount of data.

1. Integrated data

BIM big data platform will be used to aggregate various sources of data for use of diverse applications throughout the lifecycle of the buildings. For this, the typical requirements for BIM in the current environment are required plus several others:

- Programming interface to allow loading data from BIM authoring tools. Generally, exporting IFC should be the first choice since it is a standard that is supported by major BIM tools. In theory, other proprietary formats can also be supported as long as it is transformed into the same schema.
- The common schema that captures the data. In most cases today, the common schema is the IFC format. However, this format is not suitable for efficient real-time queries [9]
- A suitable schema that allows fast and efficient queries. This is essential if the data is to be used efficiently in practice. One example is the use of the data for rule checking in various phases during the lifecycle of the buildings [9]
- Integrated geometry and spatial operators in the same database, which will allow the use of the data for diverse applications including clash detection, 4D simulation, rule checking, etc.
- Generally, the requirement for the data is "write once, read many", i.e. focus will be given to enabling the data for high-performance read.
- Data must include its geographical reference location including the elevation to ensure that various building data will be in the correct placement on earth and against each other.

2. Large amount of data

The aggregation of building data is not only the contributing factor to a large amount of data. Additional data such as point cloud for the existing buildings and sensor data may provide a massive amount of data over time.

- A Large amount of data. This type of data generally takes a simple key-value type. In other cases, there may be a lot of unstructured data as well such as documents.
- The data may come in streams that require a fast insertion into the database
- Queries may require data across different buildings
- Efficient visualization may be required
- Cloud environment may be required not only to allow online, distributed access but also to free up local resources by utilizing the cloud computing power.

4 Characteristics of the big data platform

The general consensus of the big data refers to data that is large and require fast access, which the current database and tools are not suitable or able to manage. This has led to the development of various databases to address the problem. These tools are collectively known as NoSQL databases, which is a departure from the standard SQL that is used in the traditional relational database world. Grolinger and other authors have reviewed various big data NoSQL databases, and recently a new breed of databases that combine the NoSQL and features of SQL called NewSQL [10-16]. The NoSQL databases can be grouped into four general categories:

1. Key-value store
2. Document store
3. Column store
4. Graph database

Contrast to the traditional Relational Database System (RDBMS) that guarantees ACID (Atomicity, Consistency, Isolation, Durability), the NoSQL databases are often identified as BASE (Basically Available, Soft-state, Eventual consistency). This is usually achieved by relaxing one of the component in the CAP theorem [17], usually Consistency in favor of the Availability and Partitions. In each of the category, there are multiple tools and vendors offering their wares with various types of licenses, often makes it really hard to decide the right tool to do the job.

5 BIM in the big data platform

One of the important aspects of the big data platform is high-velocity. To achieve it, it is crucial to define a data model that is suitable for the purpose. Unfortunately, IFC data model without a tweak does not fit well for this purpose. BIMRL simplified schema which is derived from IFC, on the other hand, is designed for high-velocity in mind. It is modeled based on a data warehouse star-schema that reduces the link between tables to mostly one level [18, 19]. This way the information is always within a single link in most cases paving the way for simpler and faster queries. Although the schema is named simplified schema, it by no mean only handles simple queries. In fact, as will be shown in the query examples in section 6, BIMRL supports anything but simple queries. The ultimate aim of BIMRL is to provide a tool to define complex building rules relatively easy and without much programming.

5.1 The simplified schema

The BIMRL simplified schema is originally designed for a relational database system (Figure 1). The schema conforms to MVD CV2.0, which is the most basic and wide coverage of IFC MVD. It is used for the IFC certification [20]. The schema flattens many of the IFC hierarchy into just 20 tables and maintains the information from the main table in only one step away for most cases. All building objects are kept in a single BIMRL_ELEMENT table. This is consistent with the concept of data warehousing [18]. To translate the IFC data to BIMRL simplified schema, the data will go through an ETL process (Extract, Transform, Load). The details of the BIMRL schema and the ETL process is described in [19].

For the use of the big data model the schema needs to be adjusted slightly, keeping all the information intact. The outcome may vary depending on which type of the NoSQL database.

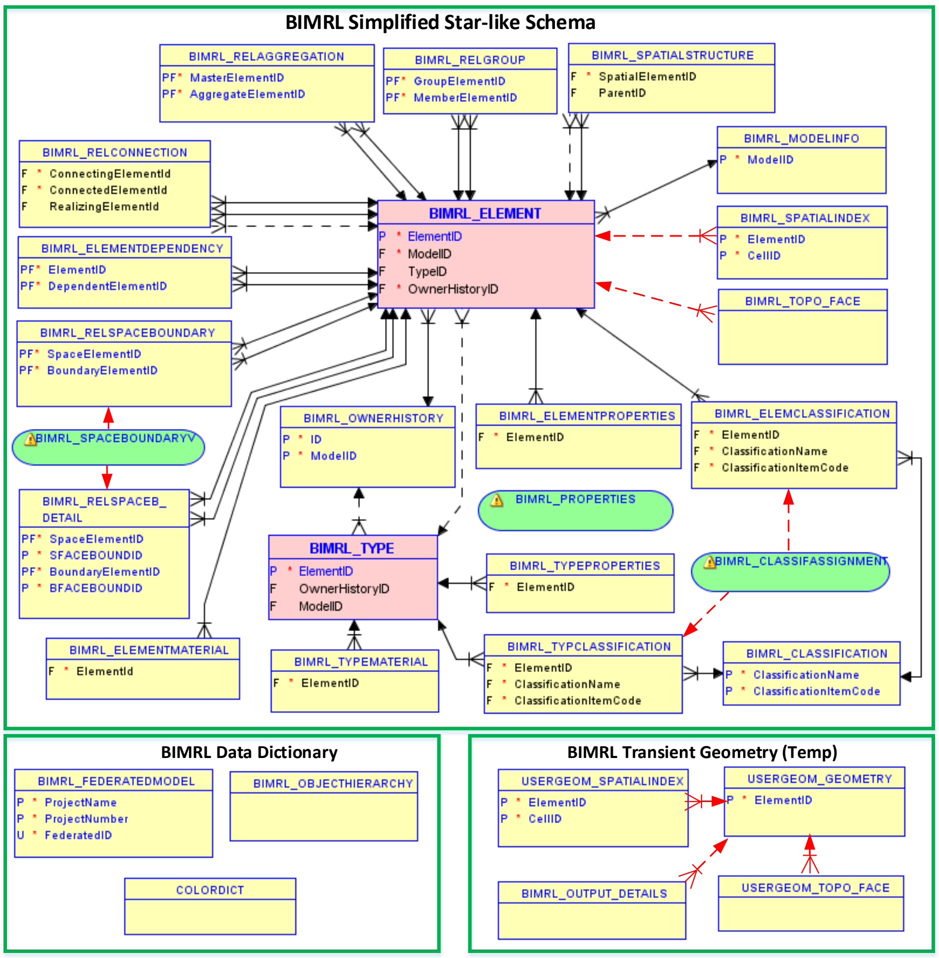


Figure 1 - BIMRL Simplified Schema

5.2 Possible approaches

There are two important considerations when choosing the NoSQL platform, i.e. understanding the characteristic of the data and what type of queries will be used against it. The first step in this research is to consider which of the four general classifications of the NoSQL platform that will be suitable for BIM data.

The key-value NoSQL platform is ruled out early because it is only suitable for data with a very simple structure of a pair of key and value. This is not the structure of the BIM data. The other three categories in some ways may be used. The most popular category of the NoSQL platform is the document store. It is a very useful platform to deal with unstructured data, but it can also be used for a structured data. The main approach for the document store is that the data should be denormalized and put into a single document whenever possible. A join is usually discouraged as it will affect performance severely. Defining a document that captures information of the main BIMRL_ELEMENT table is relatively straightforward. Its detailed tables are included by nesting them into the document. This applied to the following tables: BIMRL_ELEMENTPROPERTIES, BIMRL_ELEMCLASSIFICATION, BIMRL_ELEMENTMATERIAL, BIMRL_SPATIALINDEX, BIMRL_TOPO_FACE. These tables are those with a single arrow pointing to BIMRL_ELEMENT. In addition, small tables such as BIMRL_OWNERHISTORY, and BIMRL_MODELINFO can be denormalized and duplicated in all the individual document. BIMRL_TYPE is expected to be denormalized with this approach. But in experience, this causes an issue that the document becomes bloated because BIMRL_TYPE (representing IfcTypeObject) may own a large number of properties, materials, and classification. Duplicating the entire set of the type information causes data to swell affecting data transfer performance badly that makes it a less practical option. Besides this issue, denormalizing all the relationship tables also poses problems because it is not possible to duplicate the same element tables many times over. The relationship tables must be modeled separately and JOIN becomes unavoidable. Therefore the use of the document store NoSQL many be good for BIM data store in a traditional sense, but offers only partial benefit to be used for an efficient query-based usage without shifting a lot of responsibility to the application client to deal with the query optimization.

The next category, i.e. column store, looks more promising since it is recommended for data warehouse and OLTP types of application, which the original BIMRL schema is designed after. The column store manages the data differently than the typical row-store in the relational database. It keeps the same column data adjacent to each other instead of various column data in a row. The result, it is much more efficient for inserting, updating and querying the data when the criteria often are column based. [21-23] describe the details for the column store database comparing it to the traditional relational database's row store. Among many of the column store platform, there is a compelling option with Cassandra [24]. Cassandra allows defining a table like a document but with the well-defined schema. Internally it manages the data using column store. The previous description for the document store is applicable to this approach. However, the problems are also similar, i.e. it does not recommend JOIN. But because of the nature of BIM data, the relation is hardly avoidable. There is a possibility of using Apache Spark to handle queries with a data source in Cassandra. Spark supports a simple JOIN between two tables. This research first took this approach. But once the mapping is started, it was also realized that the highly relational data in IFC, especially for the use of relatively complex query such as ones required in rule checking, prevents the effective use of Cassandra – Spark platform [24, 25] for the same reason as the document store, i.e. shifting the responsibility of query optimization for a complex BIM queries to the applications.

This left with the last category, i.e. a graph database. Graph database stores data in vertices and edges. It seems to be suitable to deal with various relationships supported in IFC and BIMRL. Since there are also other types of data that are mainly tabular, the graph store database must be able to support document in addition to the graph. In the recent years, there is a new breed of NoSQL database is emerging that calls them themselves multi-mode NoSQL databases. The most popular graph database is Neo4J [26]. However, it does not handle other modes well. OrientDB [27] and ArangoDB [28] are among the multi-mode databases. In this research, OrientDB is selected due its popularity and completeness of the features.

5.3 The simplified schema in the graph database

Defining the graph database for the simplified schema is relatively straightforward after the initial readjustment of the mindset. This is especially relevant when defining the relationship. The graph database allows the relationship to be directly linked by an edge and therefore there is no need for primary and foreign keys for defining the relationship. Figure 2 shows the BIMRL simplified schema as defined in the graph database. Most of the relationships that are represented as different tables in the original relational database model are sufficiently represented by edges (the diagram on the right in the figure).

The multi-mode database is required to allow additional information that pertains to the specific entity to be stored as a document attached to the vertex or edge. A key-value store also will be needed to handle additional data such as sensor data that has a very simple structure.

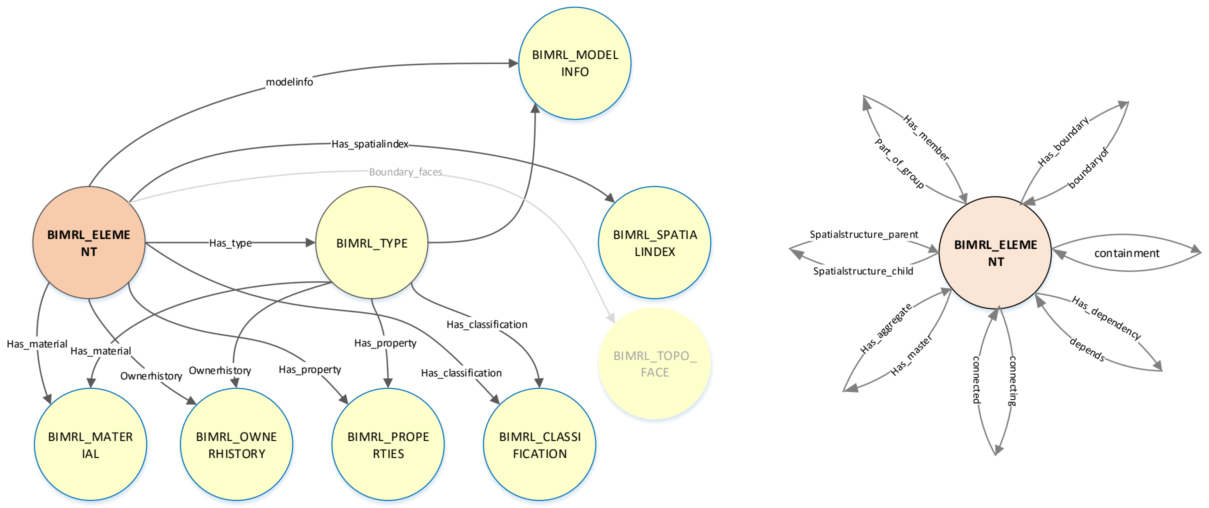


Figure 2 - BIMRL simplified schema in the graph database

6 Comparisons of the BIMRL simplified schema in RDBMS and Graph Database

To validate the effectiveness of the graph database to perform queries to the database, several queries are selected and tested against the graph database. They are compared side by side with the original relational based BIMRL (Table 1). The queries were run against a test model purpose-built to contain as many relationships typical to the BIM data as possible (Figure 3).

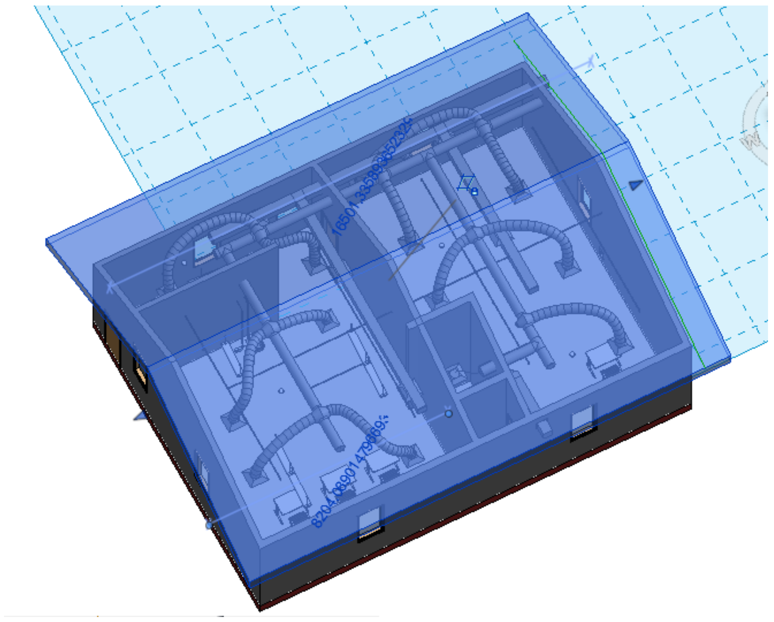


Figure 3 - The Sample Model

Table 1 - Example Queries on the BIMRL Simplified Schemas for Both Graph and Relational Databases

Query #1: Check for potential issue with a space boundary, i.e. space is isolated without a means to access it through a door, opening or direct connection via another space

BIMRL graphDB:
 	
Select @rid, elementid, name, longname, $boundary.out
from bimrl_element
   let $boundary = (select out('has_boundary')[elementtype='IFCDOOR'

```
    or elementtype='IFCOPENINGELEMENT' or elementtype='IFCSPACE'] from $parent.$current)
where elementtype='IFCSPACE' and $boundary.size()=0
```

BIMRL: (using a direct SQL statement)

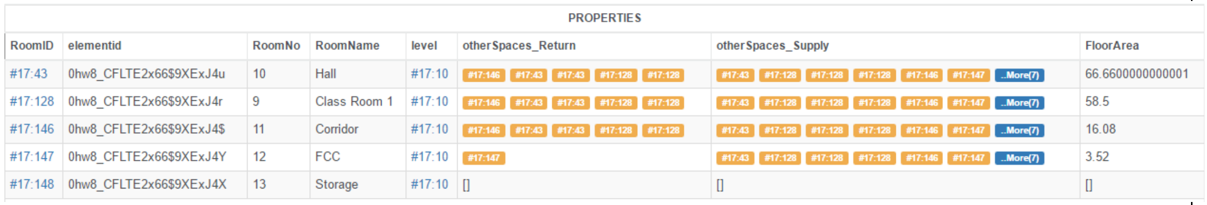
```
SQL select a.elementid, a.longname, b.parentid, c.longname
  from bimrl_element a, bimrl_spatialstructure b, bimrl_element c
  where c.elementid=b.parentid and b.spatialelementid=a.elementid
    and a.elementtype='IFCSPACE' and b.levelremoved=1 and a.elementid not in
    (select spaceelementid from bimrl_relspaceboundary
      where boundaryElementType in ('IFCDOOR','IFCOPENINGELEMENT','IFCSPACE'));;
```

Query #2:

Check for every space at ‘Level 1’ that is larger than 3.5 m^2 and is served by both Supply Air system and Return Air system.
For each of the system serving that space, check also whether the system serves the space exclusively.

BIMRL graphDB:

```
Select @rid as RoomID, elementid, name as RoomNo, longname as RoomName,
      out('spatialstructure_parent')[name='Level 1'] as level,
      out('containment')[elementtype='IFCFLOWTERMINAL'].out('part_of_group')[objecttype like
      'Return%'].out('has_member')[elementtype='IFCFLOWTERMINAL'].in('containment') as
      otherSpaces_Return,
      out('containment')[elementtype='IFCFLOWTERMINAL'].out('part_of_group')[objecttype like
      'Supply%'].out('has_member')[elementtype='IFCFLOWTERMINAL'].in('containment') as
      otherSpaces_Supply,
      out('has_property')[propertyname='NetFloorArea' and propertyvalue.asDecimal()>3.5].propertyvalue
      as FloorArea
      from bimrl_element where elementtype='IFCSPACE'
```



PROPERTIES

RoomID	elementid	RoomNo	RoomName	level	otherSpaces_Return	otherSpaces_Supply	FloorArea
#17:43	0hw8_CFLTE2x66$9XExJ4u	10	Hall	#17:10	#17:146 #17:43 #17:43 #17:128 #17:128	#17:43 #17:128 #17:128 #17:128 #17:146 #17:147 ..More(7)	66.6600000000001
#17:128	0hw8_CFLTE2x66$9XExJ4r	9	Class Room 1	#17:10	#17:146 #17:43 #17:43 #17:128 #17:128	#17:43 #17:128 #17:128 #17:128 #17:146 #17:147 ..More(7)	58.5
#17:146	0hw8_CFLTE2x66$9XExJ4$	11	Corridor	#17:10	#17:146 #17:43 #17:43 #17:128 #17:128	#17:43 #17:128 #17:128 #17:128 #17:146 #17:147 ..More(7)	16.08
#17:147	0hw8_CFLTE2x66$9XExJ4Y	12	FCC	#17:10	#17:147	#17:43 #17:128 #17:128 #17:128 #17:146 #17:147 ..More(7)	3.52
#17:148	0hw8_CFLTE2x66$9XExJ4X	13	Storage	#17:10	[]	[]	[]

BIMRL:

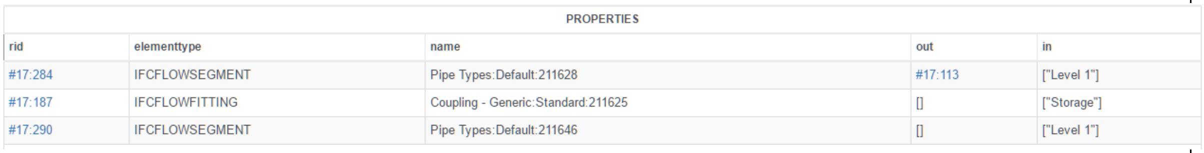
```
check ifcspace s, ifcsystem f1, ifcflowterminal t
      where container(s).name='Level 1' and systemof(t,f1,"where (f1.objecttype like 'Supply%' or
      f1.objecttype like 'Return%')")
            and contains(s,t) and property(s,NetFloorArea,to_number)>3.5
      collect s.elementid, s.name, s.longname, f1.name systemname;
evaluate nothing();
action print result;
```

Query #3:

For every space that is not ventilated (search by an existence of a diffuser), a storage or pump room, get information on other systems that pass through the space. They must be insulated (or with sleeve).

BIMRL graphDB:

```
select @rid,elementtype,name,out('connecting')[elementtype='IFCCOVERING'],in('containment').longname
   from (select expand(out('has_spatialindex').in()[elementtype like 'IFCFLOW%'])
         from (select from bimrl_element
         where (longname.toLowerCase() like '%storage%' or longname.toLowerCase() like '%pump%')
         or (out('containment')[elementtype='IFCFLOWTERMINAL'].out('has_type')
               [ifctype='IFCAIRTERMINALTYPE'].size()>0
               and out('containment')[elementtype='IFCFLOWTERMINAL'].out('part_of_group')
               [name.toLowerCase() like '%return%' or name.toLowerCase() like '%supply%'].size()=0)
         and elementtype='IFCSPACE'))
```



PROPERTIES

rid	elementtype	name	out	in
#17:284	IFCFLOWSEGMENT	Pipe Types:Default:211628	#17:113	["Level 1"]
#17:187	IFCFLOWFITTING	Coupling - Generic:Standard:211625	[]	["Storage"]
#17:290	IFCFLOWSEGMENT	Pipe Types:Default:211646	[]	["Level 1"]

BIMRL: (using chain rules)

<u>Sub-rule 1:</u>

```
check
 ifcspace s, ifcflowterminal t
   where contains(s,t,usegeometry)=.t. and t.name like '%Diffuser%'
   collect s.elementid spaceid, s.name spacenumber, s.longname spacename, count(s.elementid) cnt
   group by s.elementid, s.name, s.longname;
evaluate nothing() from (select elementid spaceid, name spacenumber, longname spacename
   from bimrl_element where elementtype='IFCSPACE' and elementid not in (select spaceid from set1)
   or lower(longname) like '%storage%' or lower(longname) like '%pump%');
action print result save into table spacenomep;
```

<u>Sub-rule 2:</u>

```
check
   { ifcspace s3, ifcdistributionelement d, ifccovering cv
      where s3.elementid in (select spaceid from spacenomep)
      and (contains(s3,d,usegeometry)=.t.) and (connectedto(d,cv,
         "where relationshiptype='IFCRELCOVERSBLDGELEMENTS'")=.t.)
      collect s3.elementid spaceid, s3.name spacenumber, s3.longname spacename, d.name flowelement,
         cv.name covering
      group by s3.elementid, s3.name, s3.longname, d.name, cv.name
   } as setd;
   { ifcspace s4, ifcdistributionelement d2
      where s4.elementid in (select spaceid from spacenomep) and (contains(s4,d2,usegeometry)=.t.)
      collect s4.elementid spaceid, s4.name spacenumber, s4.longname spacename, d2.name flowelement
      group by s4.elementid, s4.name, s4.longname, d2.name
   } as setc;
evaluate nothing() from setd right join setc using (spaceid, spacenumber, spacename, flowelement);
action print result;
```

7 Challenges and opportunities

In section 6, the potential use of BIMRL using graph database looks promising. They are not always simpler compared to the original BIMRL relational schema, but in some cases, it is easier to query certain information from the graph database than in the relational database. The statements in BIMRL examples appear much simpler than the actual underlying SQL statements because BIMRL provides built-in shortcuts to JOIN statements using built-in functions. In theory, a graph database should be also more efficient because all relationships are traversed directly (by pointer), which allow the theoretical complexity of O(1) as compared to b-tree index base in the relational database, which is O(log(n)). With such speed and other built-in support that come with the NoSQL databases to deal with a large data such automatic sharding, clustering, near linear multi-node scaling (in some databases), it potentially allows very large data being handled effectively.

However there many challenges that await those who jump to the leading edge of such technology. Although NoSQL databases have been around for more than 10 years, most are still relatively new. For example, OrientDB is less than 5 years old. This poses a problem that the technology is still going through rapid changes. Several considerations that need to be taken when deciding to get into the NoSQL bandwagon are:

1. Currently, it is difficult to choose from the plethora of options out there. There are simply difficult to compare because they are different in term of technology, features, stability, licensing model, and programming interface, to name a few. Worse still there is no agreed standard (unlike SQL), which makes each tool provides its own language. Choosing one product will mean getting stuck to it without any possibility of easily port the application to another product. For a graph database, there is a glimmer hope for a standardization of a graph traversal language with Gremlin, but it is still early and only for graph traversal for the time being [29].
2. Stability in term of both product and vendor. Beyond prototyping, there are still a lot of uncertainties in readiness of these products for a production environment. Also, one would not want to start with a product that the vendor will disappear from the market just a few years down the road.
3. Most NoSQL solutions shift responsibilities in handling the data and query to the client applications. This leads to a lot of redundant and expensive efforts by each application to optimize queries. In the relational worlds, the SQL standard has helped a lot of optimizations to be handled by the database and the client does not have to worry about it [30].

8 Conclusions

The potential of NoSQL databases, in particular, graph databases for BIM data looks promising since they can handle relationships very effectively for building information. This alone is not sufficient unless a suitable schema is used that fits very well with the idea of high-performance queries. The BIMRL simplified schema has been demonstrated to be very effective in both relational environment and the NoSQL highly distributed environment with minor adjustment without changing the fundamental design of the schema. Only the form of the schema changes with the adaptation to the graph database. With the capability to support complex queries including geometry, this solution offers an exciting future to the BIM data as a truly open database that is available for a range of uses regardless of the data size.

9 Future works

There are still works to be done to use the schema for the production quality use, they are:

1. Full integration of the geometry support
2. Testing it into a real large collection of models in a distributed environment

This research serves as a starting point toward the goals of making BIM data openly accessible and enables a wide range of applications that one may not even foresee today.

Acknowledgements

Part of the work presented here is an investigation work done in Invicara Pte. Ltd. where the main author currently works. The approach taken in this research is not directly related to the products Invicara is developing.

References

1. Forum, W.E., *Shaping the Future of Construction: A Breakthrough in Mindset and Technology*. 2016, World Economic Forum: http://www3.weforum.org/docs/WEF_Shaping_the_Future_of_Construction_full_report__.pdf.
2. Britain, D.B., *Level 3 Building Information Modelling - Strategic Plan*. 2015: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/410096/bis-15-155-digital-built-britain-level-3-strategy.pdf.
3. BCA. *BIM e-Submission*. [cited 2016 16 July].
4. Solihin, W., C. Eastman, and Y.C. Lee, *A framework for fully integrated building information models in a federated environment.* Advanced Engineering Informatics, 2016. **30**(2): p. 168-189.
5. Beetz, J., et al. *BIMserver. org–An open source IFC model server*. in *Proceedings of the CIP W78 conference*. 2010.
6. EPM. *EDM Model Server (IFC)*. http://www.epmtech.jotne.com/products/edm-model-server-ifc 2014 [cited 2014.
7. You, S.-J., D. Yang, and C.M. Eastman. *Relational DB Implementation of STEP based product model*. in *CIB World Building Congress 2004*. 2004.
8. Kang, H.-s. and G. Lee, *Development of an object-relational ifc server.* ICCEM/ICCPM, 2009.
9. Wawan, S., *(Ph.D. Thesis) A Simplified BIM Data Representation Using a Relational Database Schema for an Efficient Rule Checking System and Its Associated Rule Checking Language*, in *School of Architecture*. 2015-12-29, Georgia Institute of Technology.
10. Stonebraker, M., *SQL databases v. NoSQL databases.* Commun. ACM, 2010. **53**(4): p. 10-11.
11. Cattell, R., *Scalable SQL and NoSQL data stores.* SIGMOD Rec., 2011. **39**(4): p. 12-27.
12. Cuzzocrea, A., I.-Y. Song, and K.C. Davis, *Analytics over large-scale multidimensional data: the big data revolution!*, in *Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP*. 2011, ACM: Glasgow, Scotland, UK. p. 101-104.
13. Han, J., et al. *Survey on NoSQL database*. in *Pervasive computing and applications (ICPCA), 2011 6th international conference on*. 2011. IEEE.
14. Hecht, R. and S. Jablonski. *Nosql evaluation*. in *International conference on cloud and service computing*. 2011. IEEE.
15. Grolinger, K., et al., *Data management in cloud environments: NoSQL and NewSQL data stores.* Journal of Cloud Computing: Advances, Systems and Applications, 2013. **2**(1): p. 1.
16. Moniruzzaman, A. and S.A. Hossain, *Nosql database: New era of databases for big data analytics-classification, characteristics and comparison.* arXiv preprint arXiv:1307.0191, 2013.
17. Brewer, E.A. *Towards robust distributed systems*. in *PODC*. 2000.
18. Adamson, C., *Star schema*. 2010: McGraw-Hill.
19. Kimball, R. and M. Ross, *The data warehouse toolkit: the complete guide to dimensional modeling*. 2011: John Wiley & Sons.
20. BuildingSMART, *IFC2x3 CV2.0 Certification sub-schema (MVD)*, in *http://www.buildingsmart-tech.org/specifications/ifc-view-definition/coordination-view-v2.0/sub-schema*.
21. Stonebraker, M., et al. *C-store: a column-oriented DBMS*. in *Proceedings of the 31st international conference on Very large data bases*. 2005. VLDB Endowment.
22. Abadi, D.J., S.R. Madden, and N. Hachem, *Column-stores vs. row-stores: how different are they really?*, in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 2008, ACM: Vancouver, Canada. p. 967-980.
23. Matei, G. and R.C. Bank, *Column-oriented databases, an alternative for analytical environment.* Database Systems Journal, 2010. **1**(2): p. 3-16.
24. Apache. *Apache Cassandra*. 2016 [cited 2016 17 July 2016]; Available from: http://cassandra.apache.org/.
25. Apache. *Apache Spark*. 2016 [cited 2016 17 July 2016]; Available from: http://spark.apache.org/.
26. neo4j. *neo4j*. 2016 [cited 2016 17 July 2016]; Available from: https://neo4j.com/.
27. OrientDB. *OrientDB*. 2016 [cited 2016 17 July 2016]; Available from: http://orientdb.com/orientdb/.
28. ArangoDB. *ArangoDB*. 2016 [cited 2016 17 July 2016]; Available from: https://www.arangodb.com/.
29. Apache. *Apache TinkerPop™ -The Gremlin Graph Traversal Machine and Language*. [cited 2016 16 July].
30. Mohan, C. *History repeats itself: sensible and NonsenSQL aspects of the NoSQL hoopla*. in *Proceedings of the 16th International Conference on Extending Database Technology*. 2013. ACM.