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

Ali Ismail MS.c.
Ahmed Nahar MS.c.
Prof. Dr.-Ing. Raimar J. Scherer

EG-ICE 2017 Nottingham, 10.07.2017
- Motivation and objectives
- IFC to labeled property graph workflow:
  - IFC Meta Graph(IMG) model
  - IFC Object Graph(IOG) models
- Example of filters and queries based on the graph DB
- Conclusion and outlook
Motivation & Objectives

Motivation

- BIM models contain huge amount of information and complex relationships between their elements. This information could remain inaccessible due to the absent of suitable data management tools.
- Converting BIM models into an effective information retrievable model based on Labeld Property Graph (LPG) databases could significantly facilitate exploring and analyzing the BIM highly connected data.
Objectives

- Workflow for automatic transformation of IFC EXPRESS schema and IFC models into IFC Meta and Object Graph databases

- Utilizing graph theory concepts to explore, manage and analyse all information inside BIM models by:
  - Running queries for information retrieval.
  - Topology analysis of the model.
  - Integrated formwork for BIM models and linking the with other project information
Workflow and tools

Demo:

http://ifcwebsserver.org/ifc2neo4j

(Cypher Query Language)
.neo4j features

- intuitive, using a graph model for data representation
- reliable, fully transactional, upholds ACID
- durable and fast, using a custom disk-based, native storage engine
- massively scalable, up to several billion nodes/relationships/properties
- highly-available, when distributed across multiple machines
- expressive, with a powerful, human readable declarative graph query language
- fast, with a powerful traversal framework for high-speed graph queries
- embeddable, with a few small jars
- simple, accessible by a convenient REST API interface or an object-oriented JAVA API
- indexes are based on Apache Lucene, supports Secondary Indexes
- has been in commercial development for 10 years and in production for over 7 years; since 2003;
- Cross-platform; Simple set-up; Well documented; Open source;
- GPL for Community, AGPL for Enterprise
Cypher is a declarative graph query language that allows for expressive and efficient querying and updating of a property graph. Cypher is a relatively simple but still very powerful language. Very complicated database queries can easily be expressed through Cypher.

- powerful graph query language
- relatively simple
- declarative grammar (say what you want, not how)
- humane query language
- self-explanatory (based on English prose and neat iconography)
- written in Scala
- pattern-matching (borrows expression approaches from SPARQL)
- aggregation, ordering, limits
- create, update, delete
- structure and most of keywords inspired by SQL
- changing rather rapidly (CYPER 1.9 START …)

“Makes the simple things easy, and the complex things possible”
The labeled property graph model consists of a set of nodes and edges (relationships). An edge is always related to exactly two nodes with a fixed direction from a start to an end node, defining the property graph as a directed graph. Apart from that, two nodes can be connected by multiple edges at the same time, thus being a multigraph. Both, nodes and edges, can store a set of key-value pairs, called properties and nodes can be tagged with labels additionally.
IFC Meta Graph (IMG) model

EXPRESS -> EXPRESS Parser -> Cypher -> Graph DB

- **IFC classes**
  - (ABS) IfcElement
    - IfcBuildingElement
      - IfcBeam
      - IfcDoor
      - IfcStair
      - IfcFurnishingElement

- **Attributes**

- **Relationships**

IFC EXPRESS-G Schema data

IFC Meta Graph (IMG) model
Mapping mechanism

(1) Mapping of classes and attributes

- **Class**: IfcSlab
- **Attribute** (Another class or data type)
  - Name
  - Name

IFC Schema data

- **Class node**: IfcSlab
- **Attribute node**

IFC Meta Graph (IMG) model
Mapping mechanism

(2) Mapping of the relationships

IFC Meta Graph (IMG) model

Mandatory relation
Optional relation
supertype/subtype relation

IFC Schema data

has_property
Optional=true

Class
Attribute Definition

Supertype Class

SUBTYPE_OF

Subtype Class

IFC Meta Graph (IMG) model
Path analysis

MATCH (a:IfcRoot{Version:'IFC2x3'})-[r*]-(b:IfcSlab)
RETURN r
BIM Software -> IFC -> IFCWebserver -> Cypher -> Graph DB

(1) Preparing IFC data

(2) Generate DB
Enhancement of Graph Database

Filtration based on Multi-labeled

MATCH (n: IfcBuildingElement) RETURN n
Filters and Queries

- Verification of graph database
- Evaluation of graph capabilities
- Topology analysis
Verification of Graph DB

- IFC class types and entity list

Cypher command

MATCH (n {Model: 'Muster003.ifc'})
WITH DISTINCT LABELS (n) AS LABELS, COUNT (n) AS temp
UNWIND LABELS AS Class_name
RETURN DISTINCT Class_name, SUM (temp) AS Number_of_entities
ORDER BY Class_name

<table>
<thead>
<tr>
<th>Class_name</th>
<th>Number_of_entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>IfcApplication</td>
<td>1</td>
</tr>
<tr>
<td>IfcArbitraryOpenProfileDef</td>
<td>57</td>
</tr>
<tr>
<td>IfcAxis2Placement2D</td>
<td>33</td>
</tr>
<tr>
<td>IfcAxis2Placement3D</td>
<td>153</td>
</tr>
<tr>
<td>IfcBuilding</td>
<td>1</td>
</tr>
<tr>
<td>IfcBuildingStorey</td>
<td>1</td>
</tr>
<tr>
<td>IfcCartesianPoint</td>
<td>2534</td>
</tr>
</tbody>
</table>

Outcomes
Cypher command

```
MATCH (n (Model: 'Muster003.ifc'))
WHERE NOT ((n)--())
WITH DISTINCT LABELS (n) AS LABELS, COUNT (n) AS temp
UNWIND LABELS AS Class_name
RETURN DISTINCT Class_name, SUM(temp) AS Number_of_unconnected_entities
ORDER BY Class_name
```

Outcomes

<table>
<thead>
<tr>
<th>Class_name</th>
<th>Number_of_unconnected_entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>IfcDerivedUnit</td>
<td>2</td>
</tr>
<tr>
<td>IfcMaterial</td>
<td>3</td>
</tr>
<tr>
<td>IfcPresentationLayerAssignment</td>
<td>4</td>
</tr>
<tr>
<td>IfcPresentationStyleAssignment</td>
<td>16</td>
</tr>
<tr>
<td>IfcProductRepresentation</td>
<td>9</td>
</tr>
</tbody>
</table>
Filters and Queries

Evaluation of graph capabilities

- Filter objects based on property values

MATCH (wall: IfcWallStandardCase {Model: 'Muster003.ifc', IFCID: '2091'})-[rel]- (property: IfcPropertySet)
RETURN DISTINCT wall, rel, property
Evaluation of graph capabilities

- Get name of assigned materials of certain object(s)

MATCH
(d: IfcDoor {Model: 'Muster003.ifc', globalId:xyz})-[[*1..5]-(m: IfcMaterial)
RETURN DISTINCT (m.IFCID) AS IFCID, (m.name) AS material_name

<table>
<thead>
<tr>
<th>IFCID</th>
<th>Material_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>9472</td>
<td>'Metall - Lackiert - Grau'</td>
</tr>
<tr>
<td>9471</td>
<td>'Holz - Birke'</td>
</tr>
<tr>
<td>9470</td>
<td>'Metall - Chrom'</td>
</tr>
</tbody>
</table>
Simplified analysis of model topology for emergency routes

Construction of possible paths

Navigation routes through accesses

MATCH p=(s1:IfcSpace)- [r1:BoundedBy]- (d:IfcDoor)- [r2:BoundedBy]- (s2:IfcSpace) WHERE s1.IFCID > s2.IFCID
RETURN p
Navigation routes through space boundaries

MATCH p=(s1:IfcSpace) <-[r1:RelatingSpace]- (b:IfcRelSpaceBoundary) -[r2:RelatingSpace]- > (s2:IfcSpace) WHERE s1.IFCID > s2.IFCID RETURN p
Retrieval query for emergency routes:

- Emergency exit door.
- Accesses navigation routes.
- Space boundary routes.

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)
WHERE s1.IFCID > s2.IFCID RETURN p
Filters and Queries

‘Office_A’ model - First floor layout
Filters and Queries

‘Office_A’ model - Neo4j
Compare 2 Versions of IFC models

Find out the new windows installed in the Week37 compared with the Week30

https://github.com/openBIMstandards/DataSetSchependonlaan
Filters and Queries

MATCH (w30 { Model: 'Week30.ifc' }), (w37 { Model: 'Week37.ifc' }) where w30.tag = w37.tag
WITH collect(distinct w30.tag) as tags
MATCH (w37new) where w37new.tag <> "" and NOT w37new.tag IN tags and w37new.label = "IfcWindow"
RETURN distinct w37new.label as Class, w37new.globalId as GUID, "#" + w37new.IFCID as STEPID, w37new.name as Name, w37new.tag as Tag order by w37new.tag

<table>
<thead>
<tr>
<th>Class</th>
<th>GUID</th>
<th>STEPID</th>
<th>Name</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>IfcWindow</td>
<td>'2KsoxVt16Es8dH8L6KMvO'</td>
<td>#108847</td>
<td>'dakkoepel 1000x_(769613)'</td>
<td>'281480A1-040A-48D1-A601-A1D8110BB831'</td>
</tr>
<tr>
<td>IfcWindow</td>
<td>'1aIGZMJT6BuOcJfNgqbyU'</td>
<td>#70771</td>
<td>'velux K08_(944800)'</td>
<td>'3461CF7D-134D-4C48-8212-00B3CB2FAA89'</td>
</tr>
<tr>
<td>IfcWindow</td>
<td>'0fexbeE7QgM6YwAqinL9'</td>
<td>#293103</td>
<td>'dakkoepel 1000x_(101656)'</td>
<td>'3EA47F39-44A3-44C7-A8F2-46AD76BF079'</td>
</tr>
<tr>
<td>IfcWindow</td>
<td>'2ZTq6p98Gg0G63g6Cu7'</td>
<td>#94339</td>
<td>'velux K08_(967504)'</td>
<td>'416FC727-C981-4888-917D-B145B80A16F'</td>
</tr>
<tr>
<td>IfcWindow</td>
<td>'0FbXSSS5b75OR4wT5ab1F'</td>
<td>#47203</td>
<td>'velux K08_(926256)'</td>
<td>'45C00326-2D0A-4E32-6D62-D8E365388A8A'</td>
</tr>
<tr>
<td>IfcWindow</td>
<td>'0ySrKna6CX7Fipq2BdkG9Gi6'</td>
<td>#302403</td>
<td>'dakkoepel 1000x_(1018927)'</td>
<td>'481DE70F-AC2D-4599-864B-80ACD04EA453'</td>
</tr>
<tr>
<td>IfcWindow</td>
<td>'0SoF6Yq652GRucY3mMRA6F'</td>
<td>#290033</td>
<td>'dakkoepel 1000x_(1019167)'</td>
<td>'727B84D4-9217-4F38-9B4E-0D4A4FF7252C'</td>
</tr>
<tr>
<td>IfcWindow</td>
<td>'2PH20JYB399qJUUT1UjeF'</td>
<td>#23635</td>
<td>'velux K08_(946940)'</td>
<td>'E0CC4923-AC54-46D1-96DF-3F628C3F86A'</td>
</tr>
<tr>
<td>IfcWindow</td>
<td>'1uspy90FX6qyZDOAVQoaRe'</td>
<td>#269203</td>
<td>'dakkoepel 1000x_(1019087)'</td>
<td>'EEB2ECF2-163A-4931-5E6F2FEE508B'</td>
</tr>
<tr>
<td>IfcWindow</td>
<td>'1TLjuiM71OReaPEGSR1A'</td>
<td>#259303</td>
<td>'dakkoepel 1000x_(1018007)'</td>
<td>'FAB3F4BB-381F-4E1D-98BE-3FDFD0EA66A1'</td>
</tr>
</tbody>
</table>

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

Achievements

- Create IFC Meta Graph (IMG) model
- Create IFC Object Graph (IOG) models
- Support understanding of IFC data schema
- Analyse building information inside BIMs
- Provide realistic answers for typical queries
- Run advanced analysis of BIM models
Outlook

- Improve the mapping and converting process to cover 100% of all relationships and inverse relationships.
- Integration with a geometry engine and include geometry information in the graph DB (positions, BBOX, volumes, etc.)
- User friendly web application for accessing and managing the graph database
- Performance and benchmark tests
Thank you for your Attention

Questions?

e-mail: ali.ismail@tu-dresden.de