



Session 3.1-3.2

A hands-on BYOBIM workshop on Linked Building Data

Mads Holten Rasmussen, NIRAS A/S

Alexander Schlachter, Per Aarsleff A/S (co-speaker)

Class Description

Wouldn't it be cool if we could exchange "information" over "documents" in an interconnected manner with better insights? With Linked Building Data (LBD), we can, and in this workshop, you will learn why this is important and how you can get started with the technology today.

The Architecture, Engineering Construction and Operations (AECO) industry is fragmented as several actors consume, process and further develop a common project material throughout the building's life cycle. This, in combination with continuous design iterations and a highly document-centric working manner, challenges the design process. In his PhD, Mads has investigated the possibilities of creating a web-based infrastructure that supports the need for dynamic, interconnected information exchanges. Semantic web technologies are used to capture knowledge about a building, its spatial and physical elements and their internal relationships in an extendable, distributed data model. Such a model is referred to as an AEC Knowledge Graph and it describes the building information in an unambiguous, machine-readable manner where the processing of design information can to some extent be automated. In the workshop, Mads will initially describe the challenges of the industry and the shortcomings of Today's BIM. He will then give an introduction to the semantic web and linked data technologies and describe how these could be used to accommodate some of the identified challenges by structuring data in AEC Knowledge Graphs.

Through visual examples, it will be demonstrated what an AEC Knowledge Graph looks like, how it captures knowledge and how this knowledge can be accessed through structured queries that traverse the graph. But as a participant, you will also be able to parse one of your own IFC files into an AEC Knowledge Graph using the LD-BIM web app. During the hands-on session, you will get an introduction to the SPARQL query

3.1-3.2 - A hands-on BYOBIM workshop on Linked Building Data



Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

language and learn how to make sophisticated data requests such as “all the rooms on the 3rd floor that have south-facing windows”.

NIRAS has been using the technologies for some years and it is our hope to inspire other companies to develop minimal apps that consume and extend an AEC Knowledge Graph following our example. We have previously seen how strong communities can help to spread and mature technologies and it is our belief that the industry as a whole will benefit from such a community for AEC Knowledge Graphs.

About the Speakers:



In his role as Business Development Director in NIRAS, **Mads** is in charge of an initiative focused on establishing FAIR Data (Findable, Accessible, Interoperable & Reusable) both internally and for clients. He has a background in Architectural Engineering with a focus on design and planning on building installations. Later in his career, he finalized an industrial PhD in Linked Building Data, and as a result, Today, he has a double foundation of construction and coding experience.



Alex is MSc in Architectural Engineering from the Technical University of Denmark (2020) and BEng in Civil Engineering from Germany with a strong interest in smart methods to improve the process of construction, such as location-based scheduling. Investigating how to better plan temporary construction items, Alex dived in to the digital world of construction for his master thesis and has tried to develop and implement digital solutions for construction ever since. After two years working at NIRAS and developing Linked Data related solutions for different use cases, together with Mads, Alex is now a VDC specialist at Per Aarsleff A/S, a contractor in Denmark, and continues the same path in close collaboration with the executing party of construction.

3.1-3.2 - A hands-on BYOBIM workshop on Linked Building Data



Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

Introduction

All assignments are based on the LD-BIM web app. It is not a commercial software and is entirely built for learning and teaching purposes. It is a playground where we test out new functionality and hence it is rapidly changing. These assignments might therefore be slightly outdated in the future. Nevertheless, we believe it is a great platform for learning Linked Building Data, so let's get started!

Material

The slide deck for BILT is available at <https://bit.ly/3h59mvA>

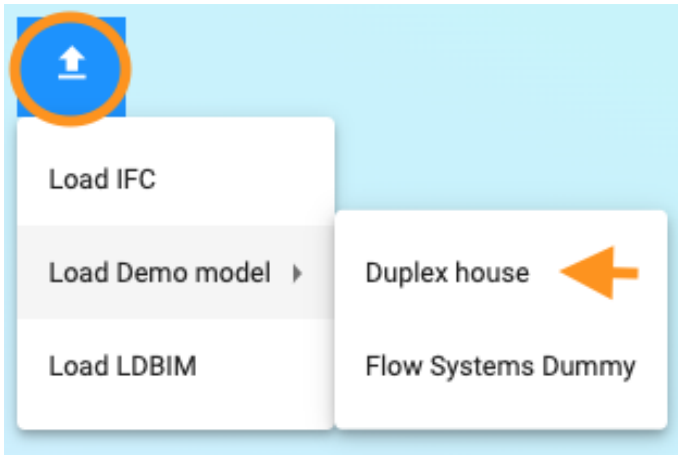
LD-BIM is available at <https://ld-bim.web.app>

LBD-school online material (work in progress) <https://bit.ly/3E7VycX>

Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

Assignment 1 – Queries on the Duplex house

In this assignment, we will be doing some queries in LD-BIM (<https://ld-bim.web.app>). We will use the Duplex House demo model.



SPARQL 101

Let's first do a simple query where we ask for all instances of [bot:Space](#).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bot: <https://w3id.org/bot#>

SELECT ?space
WHERE{
  ?space rdf:type bot:Space .
}
```

Have a look at the query above. We can divide it into three areas of interest:

- First, we have a set of prefixes. A prefix is an abbreviation used in the [Turtle](#) RDF serialization to simplify query writing. Remember that everything in Linked Data is identified with HTTP URIs i.e. web addresses? Providing the information *PREFIX bot: <https://w3id.org/bot#>* tells the interpreter that every time it sees *bot:* it is an abbreviation for that URI. *bot:Space* is therefore interpreted as *<https://w3id.org/bot#Space>*. It is also possible to write the full URIs in *<triangle brackets>*, so the below query will yield the same results.

```
SELECT ?space
WHERE{
  ?space <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://w3id.org/bot#Space> .
}
```

3.1-3.2 - A hands-on BYOBIM workshop on Linked Building Data



Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

- Next part is the *SELECT* clause which defines what we wish to return. *SELECT* will return a table with all the variables listed. There are also other clauses like *CONSTRUCT*, *ASK*, *INSERT* and *DELETE*.
- The last part is the *WHERE* clause. Here we define a triple pattern that should be matched. Anything prefixed with a ? (questionmark) indicates a variable, so the triple pattern “*?anySubject ?anyRelationship ?anyObject*” would return everything in the graph since we only provide variables. This is typically not what we want, so the result can be restricted by replacing one of the three variables with a constant. In the above example, the relationship is restricted to *rdf:type* (what relates an instance with its type) and the object is limited to *bot:Space*. Therefore, *?space* will bind to all instances of *bot:Space* in the graph.

When we execute the query in LD-BIM we get a list of space URIs prefixed with *inst:* which in LD-BIM is used for all triples in the instance namespace. Try clicking the eye icon that shows up next to the space column header when hovering. This will highlight all spaces in random colours as shown in the image below. Now toggle the “Append new” setting, hover a row in the results list and click the eye at an individual space. This will highlight that single space. Click another space and that space will be highlighted. Toggle “Append new” on and as you continue to click spaces, they are appended to the scene. You can also click on an empty place in the scene to reset colors.

The screenshot shows the LD-BIM interface. At the top, a 3D model of a building is displayed with various colored blocks representing different spaces. Below the model, the interface is divided into several sections:

- Top Left:** "22317 facts available"
- Top Right:** "Result count: 21 (completed)" and a toggle for "Append new" (currently on).
- Left Panel:** "Predefined queries" section containing a query:

```
1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX bot: <https://w3id.org/bot#>
3
4 SELECT ?space
5 WHERE{
6   ?space rdf:type bot:Space .
7 }
8
```

A "Run query" button is located at the bottom of this panel.
- Right Panel:** A table of query results with a column header "space" and an eye icon. The table lists several instance URIs, each with a small colored square next to it. A "Show elements" button is visible below the header.

Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

Now let's simplify the query a bit by replacing *rdf:type* with *a* like shown below. This is syntactic sugar in Turtle and SPARQL that means the same thing. The *bot* prefix is still necessary but throughout this assignment we leave out prefixes that have already been specified previously to save space.

```
SELECT ?space
WHERE{
  ?space a bot:Space .
}
```

Returning more variables

Let's remove one of the constants in the query and replace it with a variable so we can return more results.

```
SELECT ?something ?class
WHERE{
  ?something a ?class .
}
```

What we are basically saying here is that we wish to return anything and the class it belongs to. "a" is used to relate an instance to its class, so this makes sense. We explicitly state the names of the classes, but we could also just use an asterisk like shown below to return all results.

```
SELECT *
WHERE{
  ?something a ?class .
}
```

Extending the pattern

We are not reduced to only using single-line triple pattern matches. The dot at the end of the line indicates that the triple is over, and we are free to add another condition that should also be met. We could for example ask for all spaces including all relationships starting from a space and going to other objects. Notice that we must use the same variable name to bind to all the spaces that were retrieved from the first pattern match.

Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

```
SELECT *
WHERE{
  ?space a bot:Space .
  ?space ?property ?value .
}
```

This way we can traverse the graph and discover new information. We also extend the SELECT clause to only return unique properties. This way we will know all the different relationships that exist on spaces in our dataset.

This is a useful query pattern that can be used at any step when defining your full query. It can be used at any point to ask the question "where can I get from here" since it reveals all outgoing relationships from the current position (in this case all outgoing relationship from any space).

```
SELECT DISTINCT ?property
WHERE{
  ?space a bot:Space .
  ?space ?property ?value .
}
```

Semicolon and comma

A dot indicates that a triple is over, but we can also use semicolon which means that the subject of the previous triple is still valid for the next triple. The below query is therefore equal to the one above.

```
SELECT DISTINCT ?property
WHERE{
  ?space a bot:Space ;
  ?property ?value .
}
```

Using a colon indicates that both the subject and the predicate should be repeated. For example, the query below will return all instance of bot:Element and will further return the other classes that the element belongs to.

```
SELECT *
WHERE{
  ?element a bot:Element , ?class .
}
```

Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

Limiting results

The above query will also bind the bot:Element class to the ?class variable since it also matches the pattern. We might, however, not be interested in this. We can use a filter to leave these out from the returned results.

```
SELECT *
WHERE{
  ?element a bot:Element , ?class .
  FILTER(?class != bot:Element)
}
```

Construct queries

A construct query returns the sub-graph in a graph like data structure like Turtle or JSON-LD. The sub-graph contains only the triples that are matched by the WHERE clause. The query below will return all instances of bot:Space and all their relationships to all instances of bot:Element. Notice that LD-BIM shows the results in a graph.

```
CONSTRUCT
WHERE{
  ?space a bot:Space ;
  ?someRelationship ?element .
  ?element a bot:Element .
}
```

The screenshot shows the LD-BIM interface. At the top, there is a 3D model of a building with a blue sky background. Below the model, there is a toolbar with icons for home, search, and other functions. The main area is divided into two sections. On the left, there is a 'redefined queries' section with a text editor containing the following query:

```
1 PREFIX bot: <https://w3id.org/bot#>
2 CONSTRUCT
3 WHERE {
4   ?space a bot:Space ;
5   ?someRelationship ?element .
6   ?element a bot:Element .
7 }
8
```

On the right, there is a graph visualization of the query results. The graph shows a central node labeled 'bot.Space' connected to several other nodes representing instances of bot:Space and bot:Element. The nodes are connected by edges labeled 'bot:adjacentElement' and 'rdf:type'. The graph is displayed in a 'Graph' view, with a 'Raw' view also available. The interface also shows '2317 facts available' and a 'C' icon for clearing the graph.

Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

When you hover a node in that graph that represents an object in the IFC model that element will be highlighted. You can also click the Raw button to get the results in JSON-LD serialized RDF.

In the construct clause itself it is also possible to limit the results. We could for example omit the relationships to the classes and only return the instances and their relationships. When you use such advanced features in the WHERE clause it is no longer possible to just write CONSTRUCT. You also need to explicitly specify what should be returned.

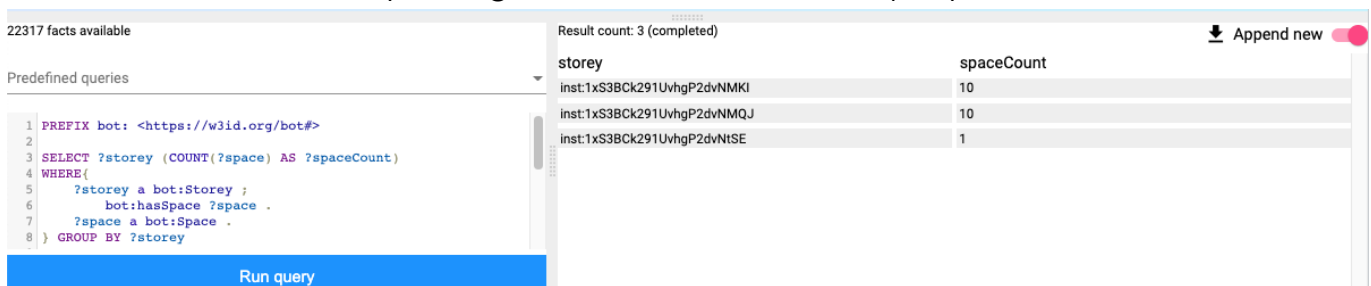
```
CONSTRUCT{
  ?space ?someRelationship ?element .
}
WHERE{
  ?space a bot:Space ;
    ?someRelationship ?element .
  ?element a bot:Element .
}
```

Aggregation functions

Let's run a simple aggregation function that counts the number of spaces in the graph. What we would like to find is the number of spaces per storey. We do this by querying relationships between storeys and spaces and grouping them by storey. In the SELECT clause we use the aggregate function COUNT() to count the number of space occurrences per storey and finally we bind the result to a new variable ?spaceCount.

```
SELECT ?storey (COUNT(?space) AS ?spaceCount)
WHERE{
  ?storey a bot:Storey ;
    bot:hasSpace ?space .
  ?space a bot:Space .
} GROUP BY ?storey
```

At the current state LD-BIM uses a SPARQL implementation based on N3/Comunica which is not so stable so you might have to execute the query a few times.



The screenshot shows a SPARQL query interface. On the left, there is a text area with the following query:

```
1 PREFIX bot: <https://w3id.org/bot#>
2
3 SELECT ?storey (COUNT(?space) AS ?spaceCount)
4 WHERE{
5   ?storey a bot:Storey ;
6     bot:hasSpace ?space .
7   ?space a bot:Space .
8 } GROUP BY ?storey
```

Below the query is a blue button labeled "Run query". On the right, the results are displayed in a table. The table has two columns: "storey" and "spaceCount". The results are:

storey	spaceCount
inst:1xS3BCK291UvhgP2dvNMKI	10
inst:1xS3BCK291UvhgP2dvNMQJ	10
inst:1xS3BCK291UvhgP2dvNtSE	1

At the top right of the results area, it says "Result count: 3 (completed)" and there is a "Append new" button with a red toggle.

3.1-3.2 - A hands-on BYOBIM workshop on Linked Building Data



Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

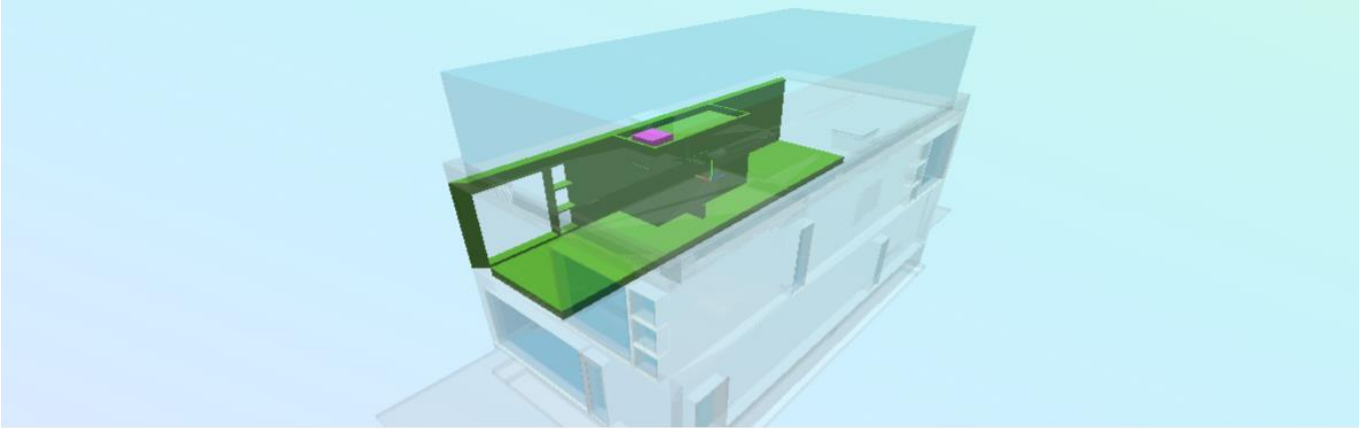
GROUP_CONCAT is another aggregate query that can be used to return all the individual spaces at each group. Again, the implementation is not strong in these advanced query patterns. It will likely change in near future since we are considering moving to Oxigraph which is also faster (see [preliminary test results](#)).

```
SELECT ?storey (GROUP_CONCAT(?space) AS ?spaces)
WHERE{
  ?storey a bot:Storey ;
    bot:hasSpace ?space .
  ?space a bot:Space .
} GROUP BY ?storey
```

Querying for space's adjacent elements seems to be more stable and the concept is the same. So if you can't get the above query to work, try this:

```
SELECT ?space (GROUP_CONCAT(?adjE1) AS ?adjacent)
WHERE {
  ?space a bot:Space ;
    bot:adjacentElement ?adjE1
} GROUP BY ?space ?area
```

LD-BIM understands that the adjacent-column contains a list of elements, so try first clicking the eye to color a space in the scene and then afterwards click the eye in the adjacent column on the same row. Try that with a few spaces and toggle "Append new" in-between to remove the previous results from the scene.



22317 facts available

Result count: 17 (completed)

Append new

space	adjacent
inst:0BTBFw6f90Nfh9rP1dIXri	inst:1hOSvn6df7F8_7GcBWIRrM inst:202Fr%24t4X7Zf8N0ew3FL9r inst:202Fr%24t4X7Zf8N0ew3FLR9 inst:202Fr%24t4X7Zf8N0ew3FL96 inst:20Brcmyk58NupXoVOHUvV inst:202Fr%24t4X7Zf8N0ew3FL8v inst:1hOSvn6df7F8_7GcBWISDM inst:20Brcmyk58NupXoVOHUvPL inst:1aj%24VJZFn2TjepZUBcKpac inst:20Brcmyk58NupXoVOHU5W inst:3bXICStxP6Fgxdej%24yc55M

Predefined queries

```
1 PREFIX bot: <https://w3id.org/bot#>
2
3 SELECT ?space (GROUP_CONCAT(?adjE1) AS ?adjacent)
4 WHERE {
5   ?space a bot:Space ;
6     bot:adjacentElement ?adjE1
7 } GROUP BY ?space ?area
```

Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

Well-known text (advanced)

LD-BIM can also visualize [Well-known Text geometries](#) like Points, linestrings and polygons from the result table. In this example, we are building quite a complex query.

We are looking at a particular space that we know exists in the Duplex house (*inst:0BTBFw6f90Nfh9rP1dl_3A*) and we use a BIND clause to bind this space URI to a variable `?space`. This approach is clean since it puts the variable in the very top of the query making it easier to understand. If you wish to do the same with another space, simply replace the URI.

We are then looking for space boundaries, which are in LD-BIM modelled as instances of [bot:Interface](#). An interface is related to the interfacing objects that are interfacing using the [bot:interfaceOf](#) relationship. We are looking for relationships to spaces and elements, so we bind to representative variables. For the interface, we are also interested in the 3D vertices and we would like to restrict the results to only vertical space boundaries. We further restrict the element variable to only hold instances of [bot:Element](#).

The last thing we do is that we use a BIND clause to build our WKT text strings in the form `POLYGON Z(x1 y1 z1, xn yn zn)` and bind them to the variable `?boundaryGeometry`.

```
PREFIX ex: <https://example.com/>
PREFIX kg: <https://w3id.org/kobl/geometry#>
PREFIX bot: <https://w3id.org/bot#>
PREFIX inst: <https://web-bim/resources/>

SELECT DISTINCT ?boundary ?boundaryGeometry ?element
WHERE{
  BIND(inst:0BTBFw6f90Nfh9rP1dl_3A AS ?space)

  ?boundary bot:interfaceOf ?space , ?element ;
    kg:vertices3D ?ver ;
    ex:isVertical true .
  ?element a bot:Element .
  BIND(CONCAT("POLYGON Z (", STR( ?ver ), ")") AS ?boundaryGeometry )
}
```

When you have executed the query, try first showing the full element column by clicking the eye.

3.1-3.2 - A hands-on BYOBIM workshop on Linked Building Data



Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

22317 facts available

Result count: 11 (completed)

Append new

Predefined queries

```

6 SELECT DISTINCT ?boundary ?boundaryGeometry ?element
7 WHERE{
8   BIND(inst:0BTBFw6f90Nfh9rP1dl_3A AS ?space)
9
10  ?boundary bot:interfaceOf ?space , ?element ;
11    kg:vertices3D ?ver ;
12    ex:isVertical true .
13  ?element a bot:Element .
14  BIND(CONCAT("POLYGON Z (", STR( ?ver ), ")") AS ?
15  boundaryGeometry )

```

boundaryGeometry	element	boundary
POLYGON Z (0.208 14.583 3.428, 0.208 17.383 3.428, 0.208 17.383 5.699999999999999, 0.208 14.583 5.699999999999999, 0.208 14.583 3.428)	inst:1f0GAJrTFv8%24zmKJOH4%24e	inst:03zEL46e54kwMort0K0hk
POLYGON Z (2.444 11.072 3.1, 2.444 11.612 3.1, 2.444 11.612 5.7, 2.444 11.072 5.7, 2.444 11.072 3.1)	inst:202Fr%24t4X7Zf8NOew3FLIE	inst:07La4RUKb3ve2KmgyyWv7l
POLYGON Z (2.444 11.612 3.1, 0.208 11.612 3.1, 0.208	inst:202Fr%24t4X7Zf8NOew3FLJd	inst:00Kj7TimCK9PekLFKbtZW

You will see all elements that are adjacent to the space and you will probably understand why we need space boundaries if we wish to address only the part of the element that is interfacing with the space. Now deselect “Append new” and show the boundaryGeometry column.

22317 facts available

Result count: 11 (completed)

Append new

Predefined queries

```

6 SELECT DISTINCT ?boundary ?boundaryGeometry ?element
7 WHERE{
8   BIND(inst:0BTBFw6f90Nfh9rP1dl_3A AS ?space)
9
10  ?boundary bot:interfaceOf ?space , ?element ;
11    kg:vertices3D ?ver ;
12    ex:isVertical true .
13  ?element a bot:Element .
14  BIND(CONCAT("POLYGON Z (", STR( ?ver ), ")") AS ?
15  boundaryGeometry )

```

boundaryGeometry	element	boundary
POLYGON Z (0.208 14.583 3.428, 0.208 17.383 3.428, 0.208 17.383 5.699999999999999, 0.208 14.583 5.699999999999999, 0.208 14.583 3.428)	inst:1f0GAJrTFv8%24zmKJOH4%24e	inst:03zEL46e54kwMort0K0hk
POLYGON Z (2.444 11.072 3.1, 2.444 11.612 3.1, 2.444 11.612 5.7, 2.444 11.072 5.7, 2.444 11.072 3.1)	inst:202Fr%24t4X7Zf8NOew3FLIE	inst:07La4RUKb3ve2KmgyyWv7l

You can also click the individual elements and their related boundary one by one to investigate the results.

Now try replacing *inst:0BTBFw6f90Nfh9rP1dl_3A* with the URI of another space and see the results.

Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

Lastly, try changing the “POLYGON Z” part of the BIND clause to “LINESTRING Z” to watch the results as linestrings instead of surfaces.

Try yourself

With what you have now learned, try formulating some queries on your own. Remember the trick you learned where you ask for outgoing relationships? Try that! You can also try clicking the “query resource” button in a cell like one of the space boundaries for example to create a query that will return everything we know about that resource. You can also try the Flow Systems demo model that contains different relationships like “Pipe supplies fluid to Fitting” and “System has component Pipe”.

Thoughts and perspectives

What you have been querying is the information that we have decided to extract from the IFC-file. It is also possible to get a full conversion of the IFC to [ifcOWL](#), but this data structure is a 1-1 representation of how it is modelled in the IFC which is really complicated to query. We just didn't want to scare you away! Also, the LBD community is arguing that this complex representation is way too complex and carries on too much legacy, so after all maybe we don't even need it in the future? LBD uses a modular approach where different domains can model their individual perspectives and bring them together as a federated whole and one might argue that this scales better.

You might have stumbled upon the fact that all properties are put in the instance namespace. This was a modelling decision made when building the IFC parser for properties. It would make sense to use the IFC namespace for all properties that are from PSets issued by BuildingSMART and then the rest of them could be described in the instance namespace.

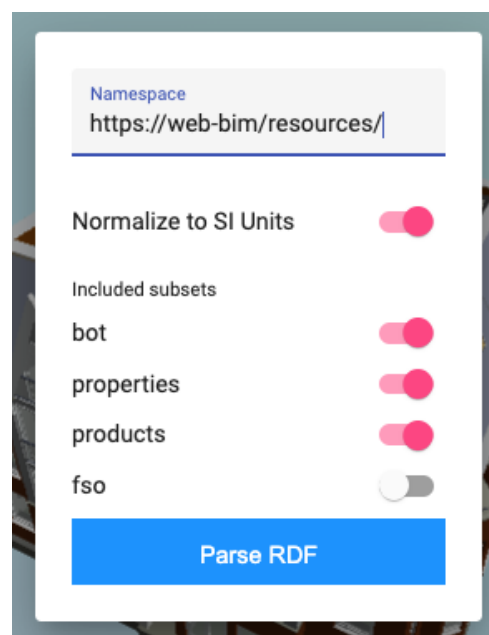
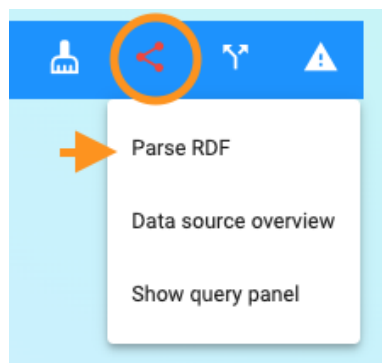
Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

Assignment 2 – Convert your own BIM model to an AEC Knowledge Graph

Load your own IFC into LD-BIM. We have experienced problems with some models so please let us know if you are experiencing problems. The parsers are built on [IFCjs](#) which is a community driven open-source project that keeps evolving and everything is in rapid development! Also keep in mind that LD-BIM didn't exist one year ago and is purely a playground.



When you see the model, click the red graph icon and select "Parse RDF" which will bring up the parser settings panel.



In the parsing panel you can set the namespace in which everything will be described (just go with the default setting).

3.1-3.2 - A hands-on BYOBIM workshop on Linked Building Data



Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

Per default all quantity units are converted to SI units but you can specify that the model units should be used.

Lastly you can choose the subsets that should be included.

BOT parses Building Topology Ontology relationships as described in the documentation: <https://w3id.org/bot>

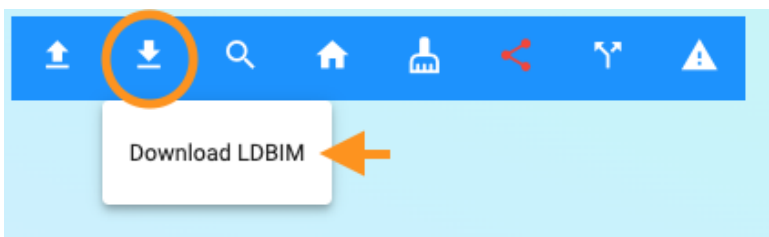
Properties parses all direct and PSet properties from the model. The way these are modelled might change in the future.

Products will include IFC Products like <x> a ifc:IfcDoor, <y> a ifc:IfcWindow etc.

FSO is only relevant for MEP models since it parses Flow Systems Ontology relationships as described in the documentation: <https://w3id.org/fso>

When you click "Parse RDF" the model is being parsed and you can see the status either in the top bar or in the browser console. When finished you can query the model just as you did with the Duplex model in assignment 1.

Since model processing is a demanding task, especially for large models you have the option to save the resulting LD-BIM model from the top menu.



Try downloading the .ldbim file and upload it again. You will see that this time it is faster.

Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

Assignment 3 – Extend the Knowledge Graph

The graph structure is quite a flexible data structure since everything is only loosely coupled. This means that you can take part of the dataset that describes the architectural design, and it will, in itself, be meaningful. You can also take the part describing the ventilation system, and it will be meaningful, and even more powerful is the fact that you can bring them together for a specific purpose on demand.

If we go back to the example from Assignment 1 with the Duplex house, this model only describes the architectural design of the house. But what if we wanted to extend this with design constraints for the mechanical design? The ultimate scenario would be that all tools in the AEC world consume and extract RDF data, but unfortunately, we are not there yet. In this assignment, however, you will learn two approaches to extending the knowledge graph.

The familiar approach: Spreadsheets!

Let's first formulate a query that will give us a list of all the spaces and their name.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bot: <https://w3id.org/bot#>
PREFIX inst: <https://web-bim/resources/>

SELECT *
WHERE{
  ?space a bot:Space ;
         rdfs:label ?name ;
         inst:areaPSetRevitDimensions ?area
}
```

Execute the query and click the download button to get a CSV you can open in Excel.

3.1-3.2 - A hands-on BYOBIM workshop on Linked Building Data



Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

22317 facts available

Result count: 21 (completed)

Append new

area	name	space
17.93623674999994	Foyer	inst:0BTBFw6f90Nfh9rP1dL_3
1.72839494131539	Utility	inst:2gRXFgjRn2HPE%24YoDLX3FC
26.11931424999996	Bedroom 1	inst:0BTBFw6f90Nfh9rP1dLXr
30.14164524999997	Living Room	inst:0BTBFw6f90Nfh9rP1dL_CZ
7.799954699999879	Hallway	inst:0BTBFw6f90Nfh9rP1dLXr
7.799954699999978	Hallway	inst:0BTBFw6f90Nfh9rP1dL_3G
26.11931424999978	Bedroom 1	inst:0BTBFw6f90Nfh9rP1dL_3A

```

1 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2 PREFIX bot: <https://w3id.org/bot#>
3 PREFIX inst: <https://web-bim/resources/>
4 SELECT *
5 WHERE {
6   ?space a bot:Space ;
7         rdfs:label ?name ;
8         inst:areaPSetRevitDimensions ?area
9 }
    
```

Run query

When you open the file in Excel it should look something like this:

	A	B	C
1	area	name	space
2	17,9362367	Foyer	https://web-bim/resources/0BTBFw6f90Nfh9rP1dL_3Q
3	1,72839494	Utility	https://web-bim/resources/2gRXFgjRn2HPE%24YoDLX3FC
4	26,1193142	Bedroom 1	https://web-bim/resources/0BTBFw6f90Nfh9rP1dLXr
5	30,1416452	Living Room	https://web-bim/resources/0BTBFw6f90Nfh9rP1dL_CZ
6	7,7999547	Hallway	https://web-bim/resources/0BTBFw6f90Nfh9rP1dLXr
7	7,7999547	Hallway	https://web-bim/resources/0BTBFw6f90Nfh9rP1dL_3G
8	26,1193142	Bedroom 1	https://web-bim/resources/0BTBFw6f90Nfh9rP1dL_3A
9	5,4158194	Bathroom 2	https://web-bim/resources/0BTBFw6f90Nfh9rP1dLXr
10	26,1779942	Bedroom 2	https://web-bim/resources/0BTBFw6f90Nfh9rP1dLXrb
11	17,9362367	Foyer	https://web-bim/resources/0BTBFw6f90Nfh9rP1dLXrr
12	4,9221725	Room	https://web-bim/resources/10mJSDZj9gPS2PrQaxa4o
13	26,1779942	Bedroom 2	https://web-bim/resources/0BTBFw6f90Nfh9rP1dL_39
14	3,997752	Bathroom 1	https://web-bim/resources/0BTBFw6f90Nfh9rP1dLXru
15	13,897501	Kitchen	https://web-bim/resources/0BTBFw6f90Nfh9rP1dLXr%24
16	30,1416452	Living Room	https://web-bim/resources/0BTBFw6f90Nfh9rP1dLXr2
17	5,44147306	Bathroom 2	https://web-bim/resources/0BTBFw6f90Nfh9rP1dL_3C
18	3,997752	Bathroom 1	https://web-bim/resources/0BTBFw6f90Nfh9rP1dL_3P
19	13,897501	Kitchen	https://web-bim/resources/0BTBFw6f90Nfh9rP1dL_3S
20	145,721689	Roof	https://web-bim/resources/0pNy6pOyf7JpMxRLgxs3sW
21	1,7540486	Utility	https://web-bim/resources/2gRXFgjRn2HPE%24YoDLX3FV
22	4,9221725	Stair	https://web-bim/resources/10mJSDZj9gPS2PrQaxa3z
23			

It is perfectly fine to rearrange the columns as you prefer. Typically you would probably hide the column containing the space URI since this is not so pretty for a non-advanced user.

In addition to what we have from LD-BIM we specify a new column titled "ventilation demand". To make it simple, we simply specify 100 m³/h for all rooms except for the Roof, Foyer, Hallway, Stair, Utility and the one called "Room".

3.1-3.2 - A hands-on BYOBIM workshop on Linked Building Data



Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

	A	B	C	D
1	space	name	area	ventilation demand
2	https://web-bim/resources/OBTBFw6f90Nfh9rP1dIXru	Bathroom 1	3,997752	=ROUND(C2*100; 0)
3	https://web-bim/resources/OBTBFw6f90Nfh9rP1dl_3P	Bathroom 1	3,997752	400
4	https://web-bim/resources/OBTBFw6f90Nfh9rP1dIXre	Bathroom 2	5,4158194	542
5	https://web-bim/resources/OBTBFw6f90Nfh9rP1dl_3C	Bathroom 2	5,44147306	544
6	https://web-bim/resources/OBTBFw6f90Nfh9rP1dIXrc	Bedroom 1	26,1193142	2612
7	https://web-bim/resources/OBTBFw6f90Nfh9rP1dl_3A	Bedroom 1	26,1193142	2612
8	https://web-bim/resources/OBTBFw6f90Nfh9rP1dIXrb	Bedroom 2	26,1779942	2618
9	https://web-bim/resources/OBTBFw6f90Nfh9rP1dl_39	Bedroom 2	26,1779942	2618
10	https://web-bim/resources/OBTBFw6f90Nfh9rP1dl_3Q	Foyer	17,9362367	0
11	https://web-bim/resources/OBTBFw6f90Nfh9rP1dIXrr	Foyer	17,9362367	0
12	https://web-bim/resources/OBTBFw6f90Nfh9rP1dIXri	Hallway	7,7999547	0
13	https://web-bim/resources/OBTBFw6f90Nfh9rP1dl_3G	Hallway	7,7999547	0
14	https://web-bim/resources/OBTBFw6f90Nfh9rP1dIXr%24	Kitchen	13,897501	1390
15	https://web-bim/resources/OBTBFw6f90Nfh9rP1dl_3S	Kitchen	13,897501	1390
16	https://web-bim/resources/OBTBFw6f90Nfh9rP1dl_CZ	Living Room	30,1416452	3014
17	https://web-bim/resources/OBTBFw6f90Nfh9rP1dIXr2	Living Room	30,1416452	3014
18	https://web-bim/resources/OpNy6pOyf7JpMxRLgxs3sW	Roof	145,721689	0
19	https://web-bim/resources/10mjSDZJj9gPS2PrQaxa4o	Room	4,9221725	0
20	https://web-bim/resources/10mjSDZJj9gPS2PrQaxa3z	Stair	4,9221725	0
21	https://web-bim/resources/2gRXFgjRn2HPE%24YoDLX3FC	Utility	1,72839494	0
22	https://web-bim/resources/2gRXFgjRn2HPE%24YoDLX3FV	Utility	1,7540486	0
23				

By the way, wonder what that "Room" is? Do a simple CONSTRUCT query on the element and check!

The screenshot shows a BIM visualization tool interface. At the top, there is a blue toolbar with icons for home, search, and other functions. Below the toolbar is a 3D model of a building with a yellow rectangular area highlighted on the ground floor. Below the model, there is a section for "Predefined queries" with a query editor. The query is as follows:

```

1 CONSTRUCT
2 WHERE {
3   <https://web-bim/resources/10mjSDZJj9gPS2PrQaxa4o> ?p
4   ?o
5 }

```

Below the query editor is a "Run query" button. To the right of the query editor is a graph visualization showing a central node "Inst:10mjSDZJj9gPS2PrQaxa4o" connected to various other nodes and properties. The graph includes nodes like "FireAlarmZoneName", "Stairway", "LoadCapacity", and "Inst:20".

Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

Bonus assignment

The room name is actually a mistake. It should have been a Stair. Let's correct this with this UPDATE query:

```
DELETE{
  <https://web-bim/resources/10mjSDZJj9gPS2PrQaxa4o> rdfs:label ?oldName
}
INSERT{
  <https://web-bim/resources/10mjSDZJj9gPS2PrQaxa4o> rdfs:label "Stair"
}
WHERE{
  <https://web-bim/resources/10mjSDZJj9gPS2PrQaxa4o> rdfs:label ?oldName
}
```

Writing the full URL three times is a bit ugly, so let's use a BIND to def a variable ?space for it:

```
DELETE{
  ?space rdfs:label ?oldName
}
INSERT{
  ?space rdfs:label "Stair"
}
WHERE{
  BIND(<https://web-bim/resources/10mjSDZJj9gPS2PrQaxa4o> AS ?space)
  ?space rdfs:label ?oldName
}
```

Don't bother with the query result feedback that says nothing was inserted or deleted. This is because of the simplified method for detecting changes which is basically comparing the store size. So to check that the value was actually updated, query for the space name:

```
SELECT *
WHERE{
  <https://web-bim/resources/10mjSDZJj9gPS2PrQaxa4o> rdfs:label ?name
}
```

Sorry for that little detour :D The last thing we need to do in the CSV file is to insert a new row that is used to identify what is to be inserted back into LD-BIM. The column containing the space URIs is given the value "id", which the importer uses to detect the resource to which the new properties should be added. The new ventilation demand column is given the full URI we would like to identify that property with. In our example, let's just use a fictive namespace "https://example.com#" and the identifier

3.1-3.2 - A hands-on BYOBIM workshop on Linked Building Data



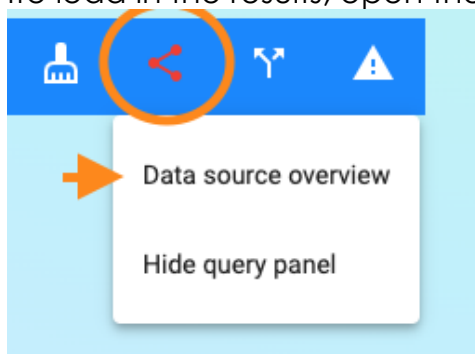
Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

“ventilationDemand”. In your project, company, country, standardization committee etc. you should specify your own properties and provide metadata for them.

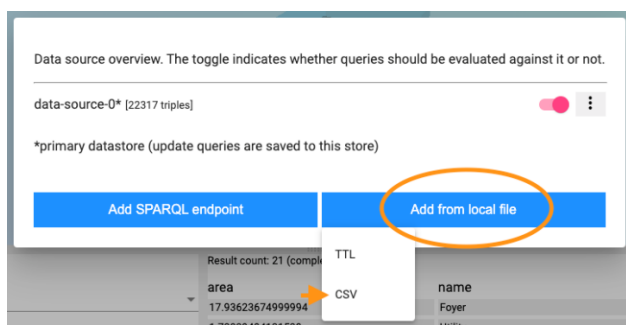
	A	B	C	D	E	F
1	space	name	area	ventilation demand		
2	id			https://example.com#ventilationDemand		
3	https://web-bim/resources/0BTBFw6f90Nfh9rP1dlXru	Bathroom 1	3,997752	400		
4	https://web-bim/resources/0BTBFw6f90Nfh9rP1dl_3P	Bathroom 1	3,997752	400		
5	https://web-bim/resources/0BTBFw6f90Nfh9rP1dlXre	Bathroom 2	5,4158194	542		
6	https://web-bim/resources/0BTBFw6f90Nfh9rP1dl_3C	Bathroom 2	5,44147306	544		
7	https://web-bim/resources/0BTBFw6f90Nfh9rP1dlXrc	Bedroom 1	26,1193142	2612		
8	https://web-bim/resources/0BTBFw6f90Nfh9rP1dl_3A	Bedroom 1	26,1193142	2612		
9	https://web-bim/resources/0BTBFw6f90Nfh9rP1dlXrb	Bedroom 2	26,1779942	2618		
10	https://web-bim/resources/0BTBFw6f90Nfh9rP1dl_39	Bedroom 2	26,1779942	2618		
11	https://web-bim/resources/0BTBFw6f90Nfh9rP1dl_3Q	Foyer	17,9362367	0		
12	https://web-bim/resources/0BTBFw6f90Nfh9rP1dlXrr	Foyer	17,9362367	0		
13	https://web-bim/resources/0BTBFw6f90Nfh9rP1dlXri	Hallway	7,7999547	0		
14	https://web-bim/resources/0BTBFw6f90Nfh9rP1dl_3G	Hallway	7,7999547	0		
15	https://web-bim/resources/0BTBFw6f90Nfh9rP1dlXr%24	Kitchen	13,897501	1390		
16	https://web-bim/resources/0BTBFw6f90Nfh9rP1dl_3S	Kitchen	13,897501	1390		
17	https://web-bim/resources/0BTBFw6f90Nfh9rP1dl_CZ	Living Room	30,1416452	3014		
18	https://web-bim/resources/0BTBFw6f90Nfh9rP1dlXr2	Living Room	30,1416452	3014		
19	https://web-bim/resources/0pNy6pOyf7JPmXRLgxs3sW	Roof	145,721689	0		
20	https://web-bim/resources/10mjSDZJj9gPS2PrQaxa4o	Room	4,9221725	0		
21	https://web-bim/resources/10mjSDZJj9gPS2PrQaxa3z	Stair	4,9221725	0		
22	https://web-bim/resources/2gRXFgjRn2HPE%24YoDLX3FC	Utility	1,72839494	0		
23	https://web-bim/resources/2gRXFgjRn2HPE%24YoDLX3FV	Utility	1,7540486	0		
24						

Once we have our information we save it in a new CSV file “ventilation-demands.csv”.

To load in the results, open the data source overview.



Then click Add from file and select CSV.



3.1-3.2 - A hands-on BYOBIM workshop on Linked Building Data



Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

From the CSV upload panel set the separator and load it in. You should now see the triples that will be inserted in JSON-LD.

```
1 [
2   {
3     "@id": "https://web-bim/resources/0BTBFw6f90Nfh9rP1dlXru",
4     "https://example.com#ventilationDemand": {
5       "@value": "400",
6       "@type": "xsd:decimal"
7     }
8   },
9   {
10    "@id": "https://web-bim/resources/0BTBFw6f90Nfh9rP1dl_3P",
11    "https://example.com#ventilationDemand": {
12      "@value": "400",
13      "@type": "xsd:decimal"
14    }
15  },
16  {
17    "@id": "https://web-bim/resources/0BTBFw6f90Nfh9rP1dlXre",
18    "https://example.com#ventilationDemand": {
19      "@value": "542",
```

Cancel Save results

Finally click Save results to write the new triples to the store.

To confirm that the new information was added, let's query for the ventilation demands.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bot: <https://w3id.org/bot#>
PREFIX ex: <https://example.com#>
PREFIX inst: <https://web-bim/resources/>

SELECT *
WHERE{
  ?space a bot:Space ;
  rdfs:label ?name ;
  inst:areaPSetRevitDimensions ?area ;
  ex:ventilationDemand ?vd .
}
```

Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

The sophisticated one: Update queries

If you did the previous step you will need to refresh the page to start from scratch.

Let's first formulate a query that will calculate ventilation demands as the space area multiplied by 100. The query below will do exactly that. What it does is that it first finds bindings for `?space`, `?name` and `?area` as we have seen before. In addition to that, however, it uses a BIND clause to create a new variable in the binding holding the result of area multiplied by 100 in a new variable `?vd`.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bot: <https://w3id.org/bot#>
PREFIX ex: <https://example.com#>
PREFIX inst: <https://web-bim/resources/>

SELECT *
WHERE{
  ?space a bot:Space ;
         rdfs:label ?name ;
         inst:areaPSetRevitDimensions ?area
         BIND(?area*100 AS ?vd)
}
```

Since we would like to round the result, we can extend the BIND a bit:

```
SELECT *
WHERE{
  ?space a bot:Space ;
         rdfs:label ?name ;
         inst:areaPSetRevitDimensions ?area
         BIND(ROUND(?area*100) AS ?vd)
}
```

There were also some rooms that we did not wish to ventilate, so let's set up some filters that will filter out spaces with names containing either "roof", "hallway", "foyer" or "utility". "i" means that the match is incasesensitive. You can probably imagine that such filtering could also be reversed to apply the ventilation demand only to certain rooms.

3.1-3.2 - A hands-on BYOBIM workshop on Linked Building Data



Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

```
SELECT *
WHERE{
  ?space a bot:Space ;
  rdfs:label ?name ;
  inst:areaPSetRevitDimensions ?area .

  FILTER(!REGEX("roof", ?name, "i"))
  FILTER(!REGEX("hallway", ?name, "i"))
  FILTER(!REGEX("foyer", ?name, "i"))
  FILTER(!REGEX("utility", ?name, "i"))

  BIND(ROUND(?area*100) AS ?vd)
}
```

Until now, we have just been returning the results, but the end goal is to actually insert the ventilation demands. Before we do that, let's use a CONSTRUCT query to return the new triples that we are building. Remember in assignment 1 where we formulated a CONSTRUCT query that would return a sub-graph? In the CONSTRUCT clause, you can actually restructure the results before returning them! So let's define a CONSTRUCT query that will return the ventilation demands:

```
CONSTRUCT{
  ?space ex:ventilationDemand ?vd
}
WHERE{
  ?space a bot:Space ;
  rdfs:label ?name ;
  inst:areaPSetRevitDimensions ?area .

  FILTER(!REGEX("roof", ?name, "i"))
  FILTER(!REGEX("hallway", ?name, "i"))
  FILTER(!REGEX("foyer", ?name, "i"))
  FILTER(!REGEX("utility", ?name, "i"))

  BIND(ROUND(?area*100) AS ?vd)
}
```


3.1-3.2 - A hands-on BYOBIM workshop on Linked Building Data



Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

22317 facts available

Predefined queries

```
3 PREFIX ex: <https://example.com/#>
4 PREFIX inst: <https://web-bim/resources/>
5
6 CONSTRUCT{
7   ?space ex:ventilationDemand ?vd
8 }
9 WHERE{
10  ?space a bot:Space ;
11   rdfs:label ?name ;
12   inst:areaPSetRevitDimensions ?area .
13
14  FILTER(!REGEX("roof", ?name, "i"))
15  FILTER(!REGEX("hallway", ?name, "i"))
16  FILTER(!REGEX("Elevator", ?name, "i"))

```

Run query

Graph

Raw

This returns a visual graph representation of the results, but if you click the Raw button, you will see the RDF serialized as JSON-LD.

```
},
"@graph": [
  {
    "@id": "inst:0BTBFw6f90Nfh9rP1dlXrc",
    "https://example.com/#ventilationDemand": {
      "@type": "xsd:decimal",
      "@value": "2612"
    }
  },
  {
    "@id": "inst:0BTBFw6f90Nfh9rP1dl_CZ",
    "https://example.com/#ventilationDemand": {
      "@type": "xsd:decimal",
      "@value": "3014"
    }
  }
]
```

In the above snippet you see that each resource is assigned a ventilation demand as we wished. It was even automatically detected that the datatype is decimal numbers.

Let's materialize this simply by replacing CONSTRUCT with INSERT.

```
INSERT{
  ?space ex:ventilationDemand ?vd
}
WHERE{
  ?space a bot:Space ;
  rdfs:label ?name ;

```


Mads Holten Rasmussen, NIRAS A/S NIRAS A/S

```
inst:areaPSetRevitDimensions ?area .  
  
FILTER(!REGEX("roof", ?name, "i"))  
FILTER(!REGEX("hallway", ?name, "i"))  
FILTER(!REGEX("foyer", ?name, "i"))  
FILTER(!REGEX("utility", ?name, "i"))  
  
BIND(ROUND(?area*100) AS ?vd)  
}
```

To confirm that the new information was added, let's query for the ventilation demands.

```
SELECT *  
WHERE{  
  ?space a bot:Space ;  
  rdfs:label ?name ;  
  inst:areaPSetRevitDimensions ?area ;  
  ex:ventilationDemand ?vd .  
}
```

Other approaches

Many tools in the AEC industry have open APIs and serializing RDF is not hard after what you have learned here. How about letting your Dynamo nodes or Revit plugins import and export RDF so the data can be used in a larger context?

Downloading results

LD-BIM is capable of importing and exporting an .ldbim-file which is basically a zip containing a GLTF serialization of the IFC and ttl-files with LBD data. If you download this file it will include the new data you added.

It is also possible to just download the RDF data from the data source overview.