Unlock the Full Potential of Your MEP Data: The Case for a Unified Data Model

Will Reynolds

Principal Digital Applications Developer

AUTODESK UNIVERSITY

HOARE LEA H.

About the speaker

Will Reynolds

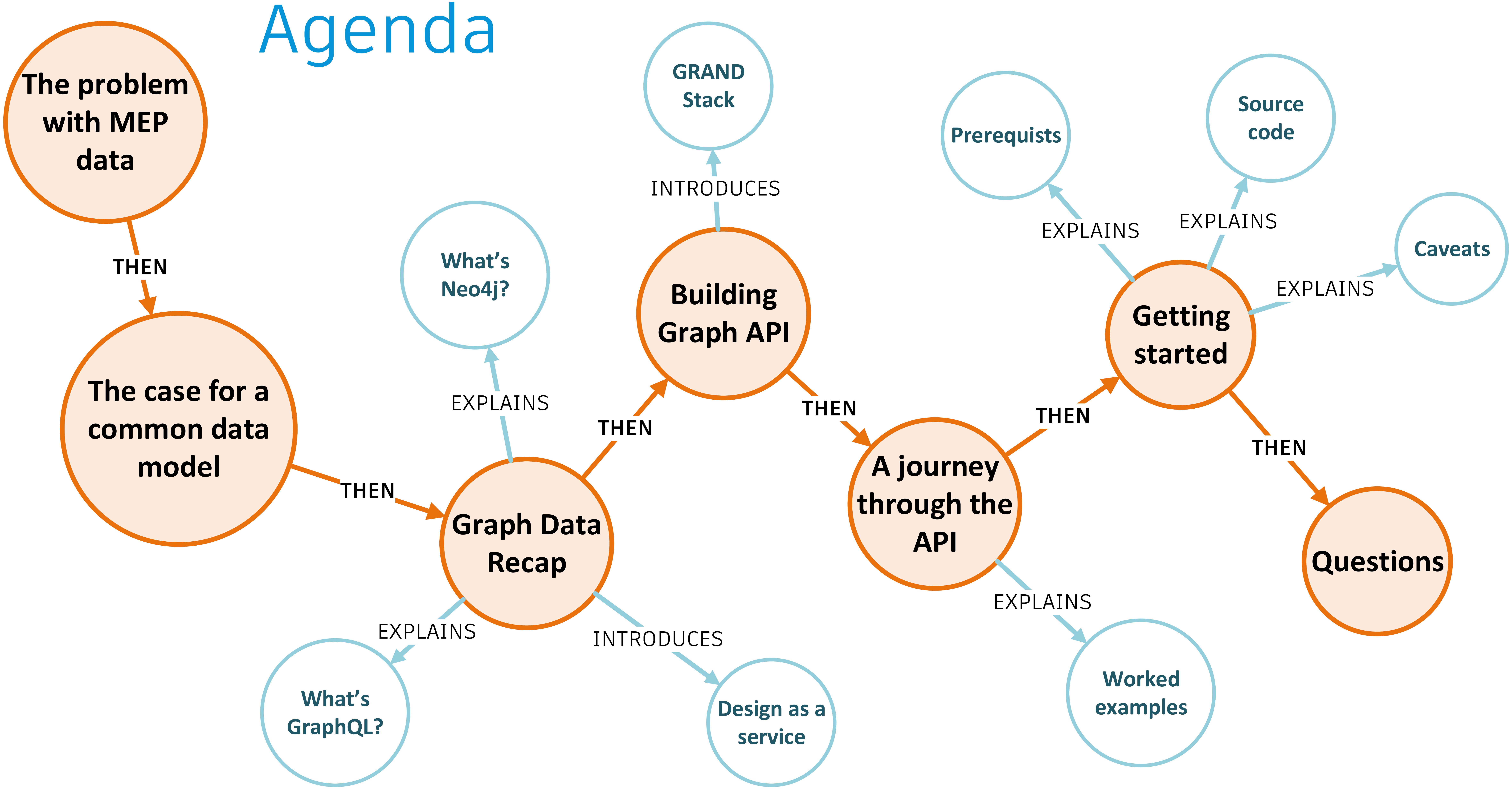
- 14 Years with the UK's largest MEP Consultancy + Specialists
- Principal digital applications developer - Digital Innovation group
- 6 years Revit app developer
- Background in electronics and digital systems

✉ WillReynolds@HoareLea.com

https://twitter.com/d2liYmxl

https://www.instagram.com/d2liymxl/

Agenda



The problem with MEP data

THEN

The case for a common data model

THEN

Graph Data Recap

EXPLAINS

What's Neo4j?

EXPLAINS

What's GraphQL?

INTRODUCES

Design as a service

THEN

Building Graph API

INTRODUCES

GRAND Stack

THEN

A journey through the API

EXPLAINS

Worked examples

THEN

Getting started

EXPLAINS

Prerequists

EXPLAINS

Source code

EXPLAINS

Caveats

THEN

Questions

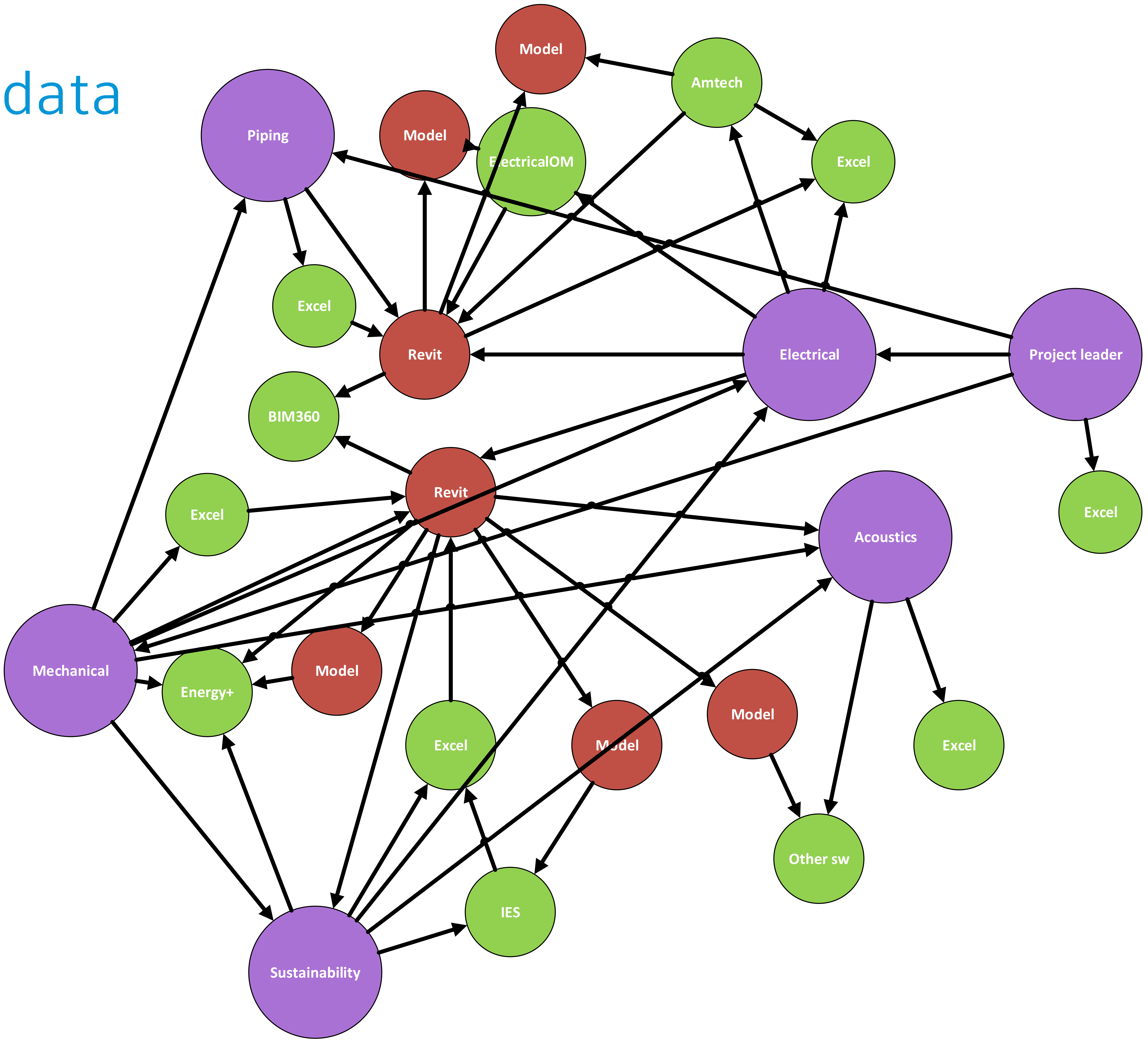
HOARE LEA

The problem with MEP data

THEN

The problem with MEP data

- Many teams
- With different applications
- Each with their own models or view on data
- Often with different names for the same parameters



The problem with MEP data

- Additional workload in exporting, translating and importing data
- Data replicated in multiple places
- Inconsistencies across data siloes
- Reliance on Excel
- Knowledge, calcs, techniques known by individuals, lost for other teams

HOARE LEA

data

THEN

The case for a common data model

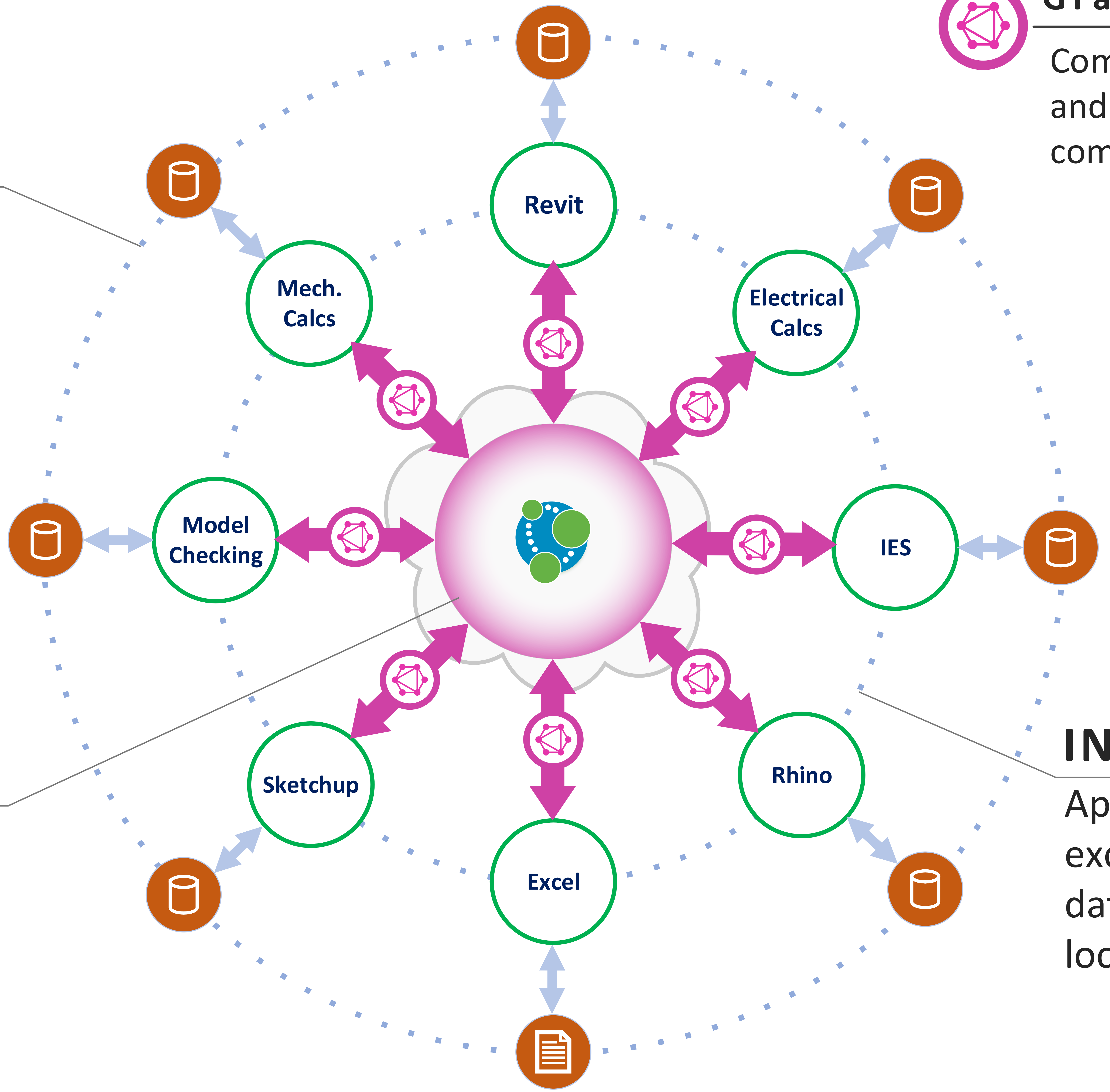
THEN

Graph Data Recap

THEN

The da schem

The case for a common data model



GraphQL

Common query language and schema for communicating data.

Neo4j

Revolutionary database for interrelated data.

DISPARTE RING

File exchange between applications. Currently, the primary way to exchange building geometry.

DESIGN NUCLEUS

Cloud distributed single source of truth for the current and developing MEP design.

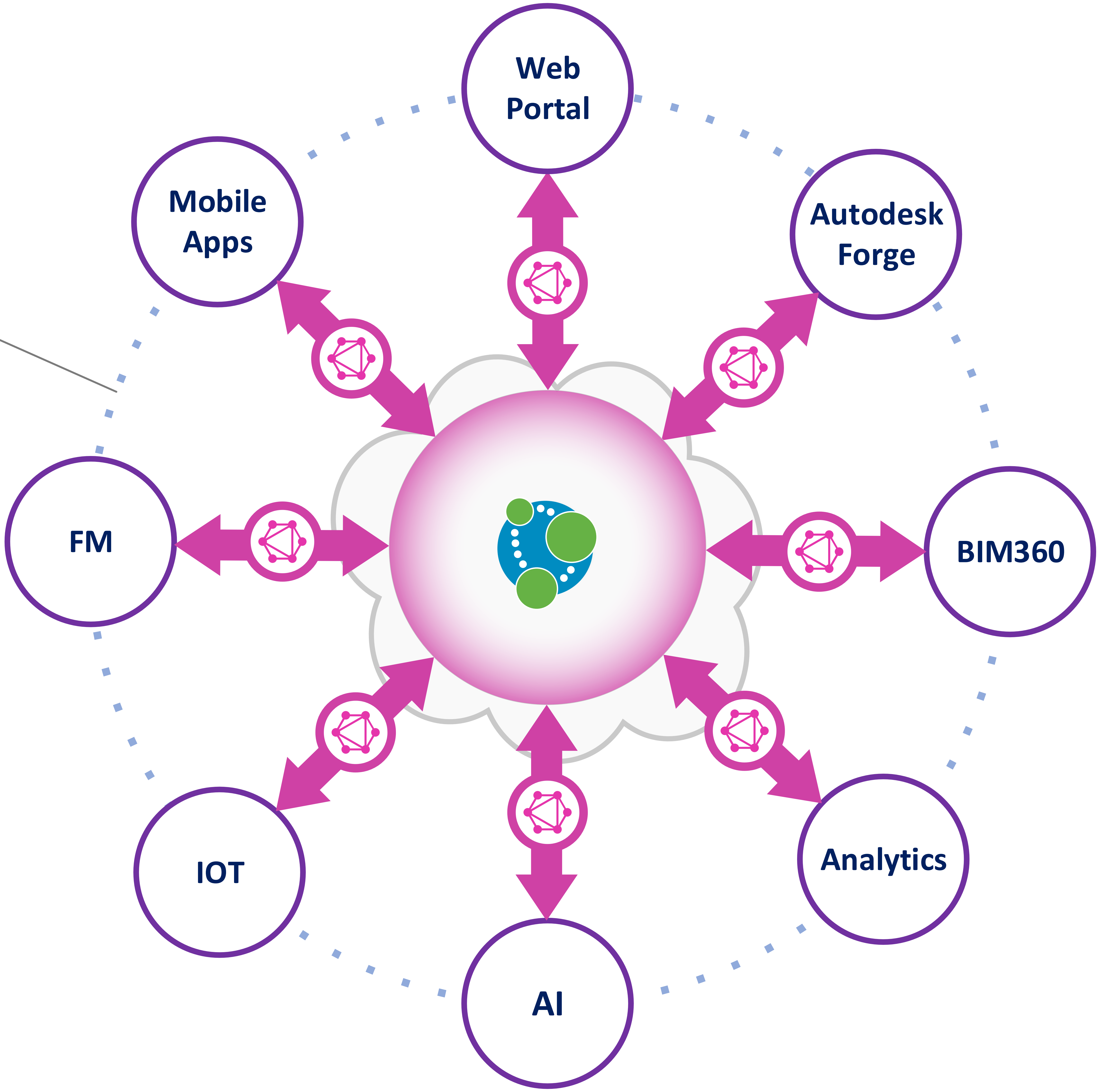
INTEROP RING

App-to-app data exchange. Realtime data exchange for local iterative design.

Introducing: Design, or calcs, as a service (Daas?)

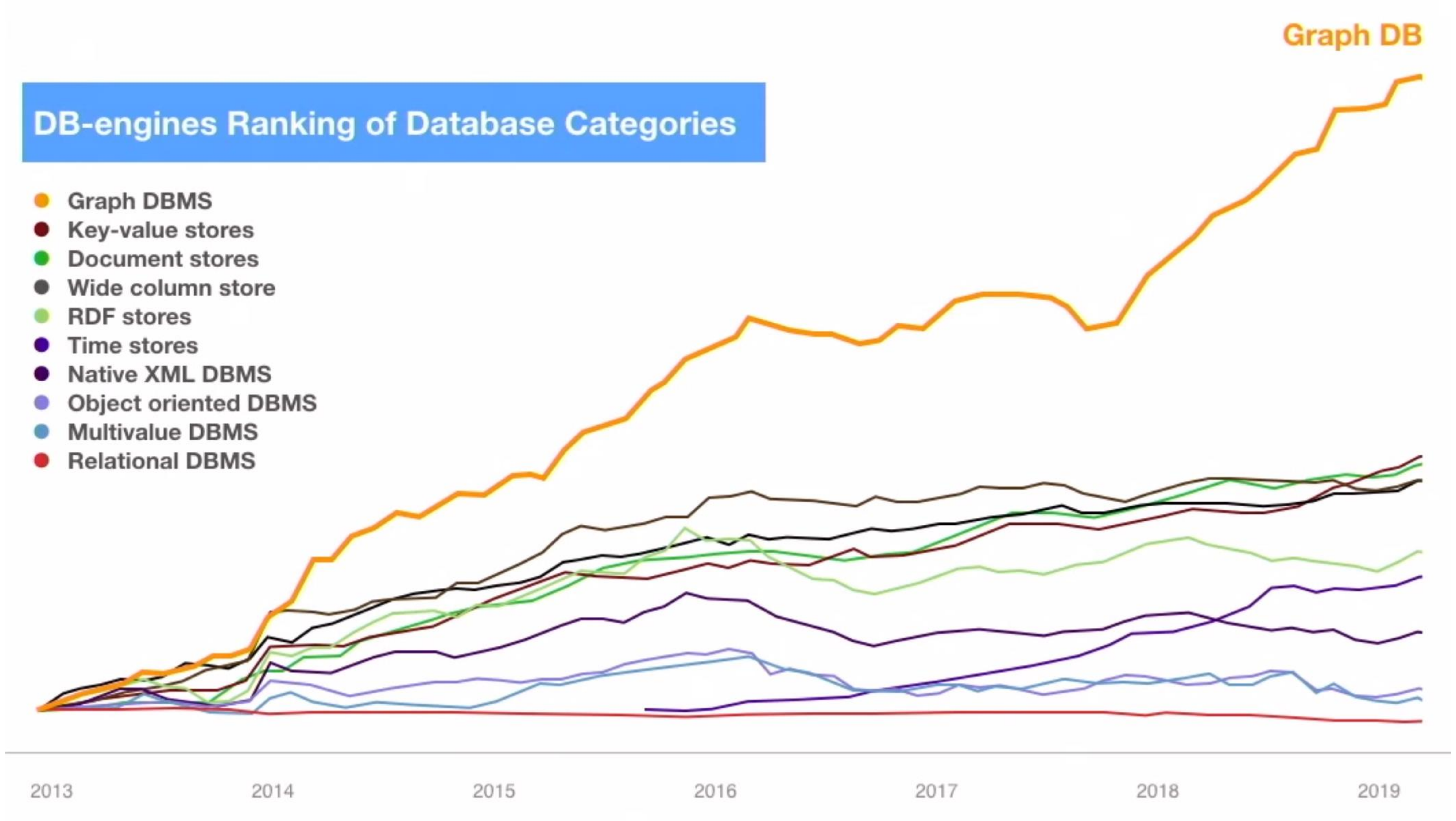
SERVICES RING

Direct access to building data for other services and APIs



What is Neo4j

https://neo4j.com



Neo4j NODES 2019 – Keynote
https://youtu.be/AfhJcyys108?t=529

- Graph native database
- Developer focused
- Plugin framework, graph traversal, geometry + more
- Well supported and documented
- Working to create GQL query language standard

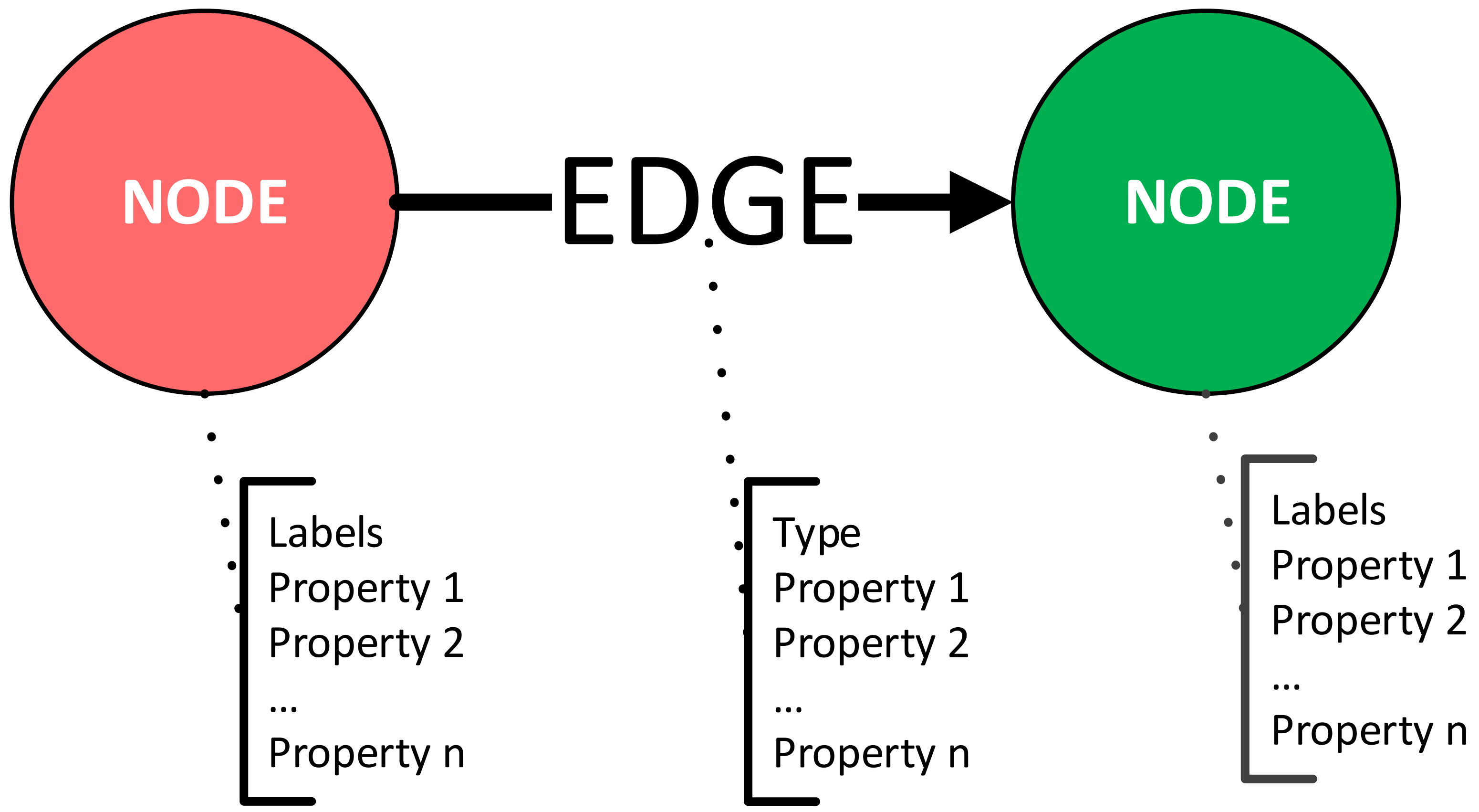
Graph Data – quick recap and comparison with RDBS

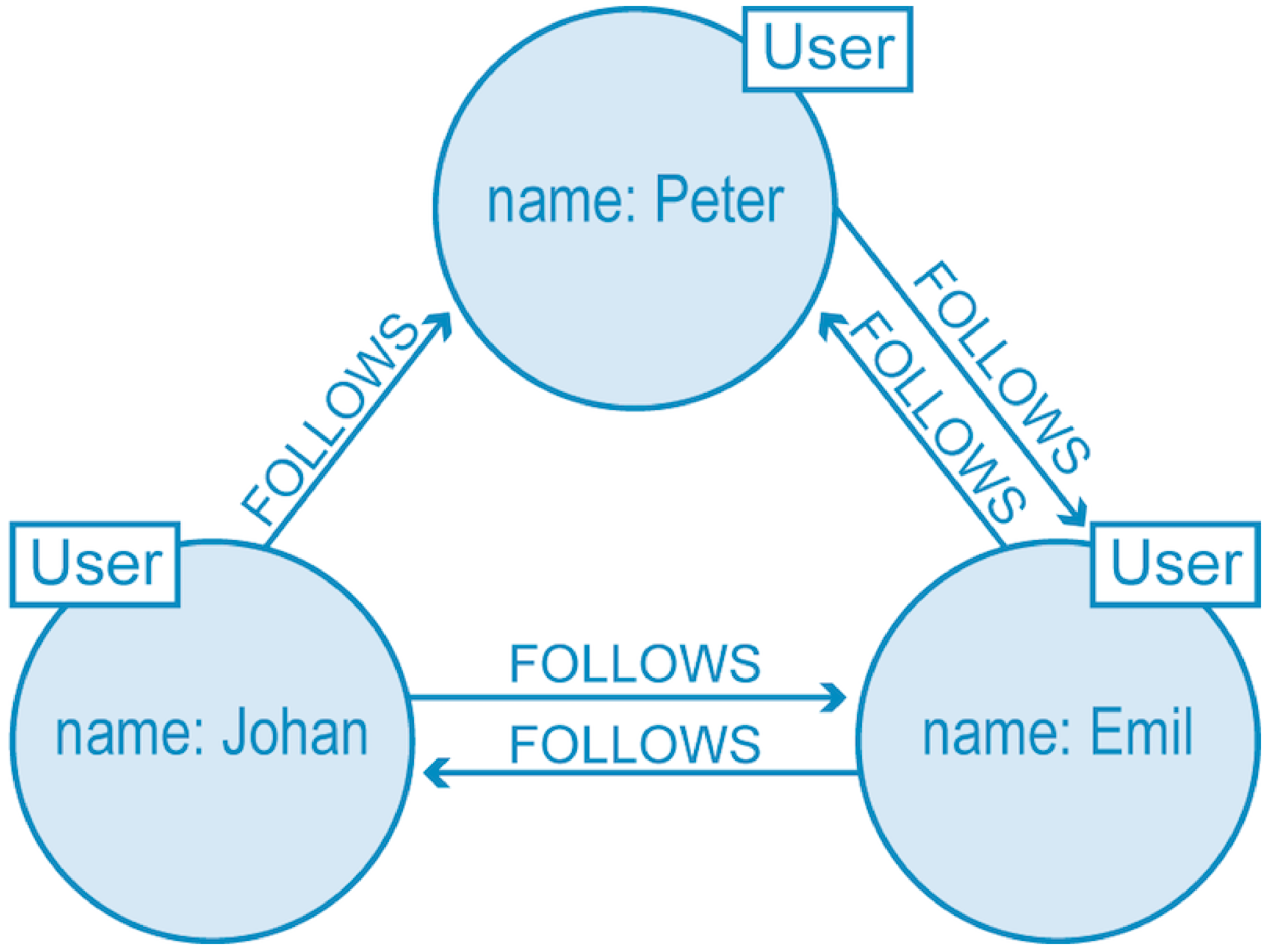
- In graph databases the emphasis is on data relationships

- Querying complex relationships can be complicated in SQL, and other NoSQL DBs.

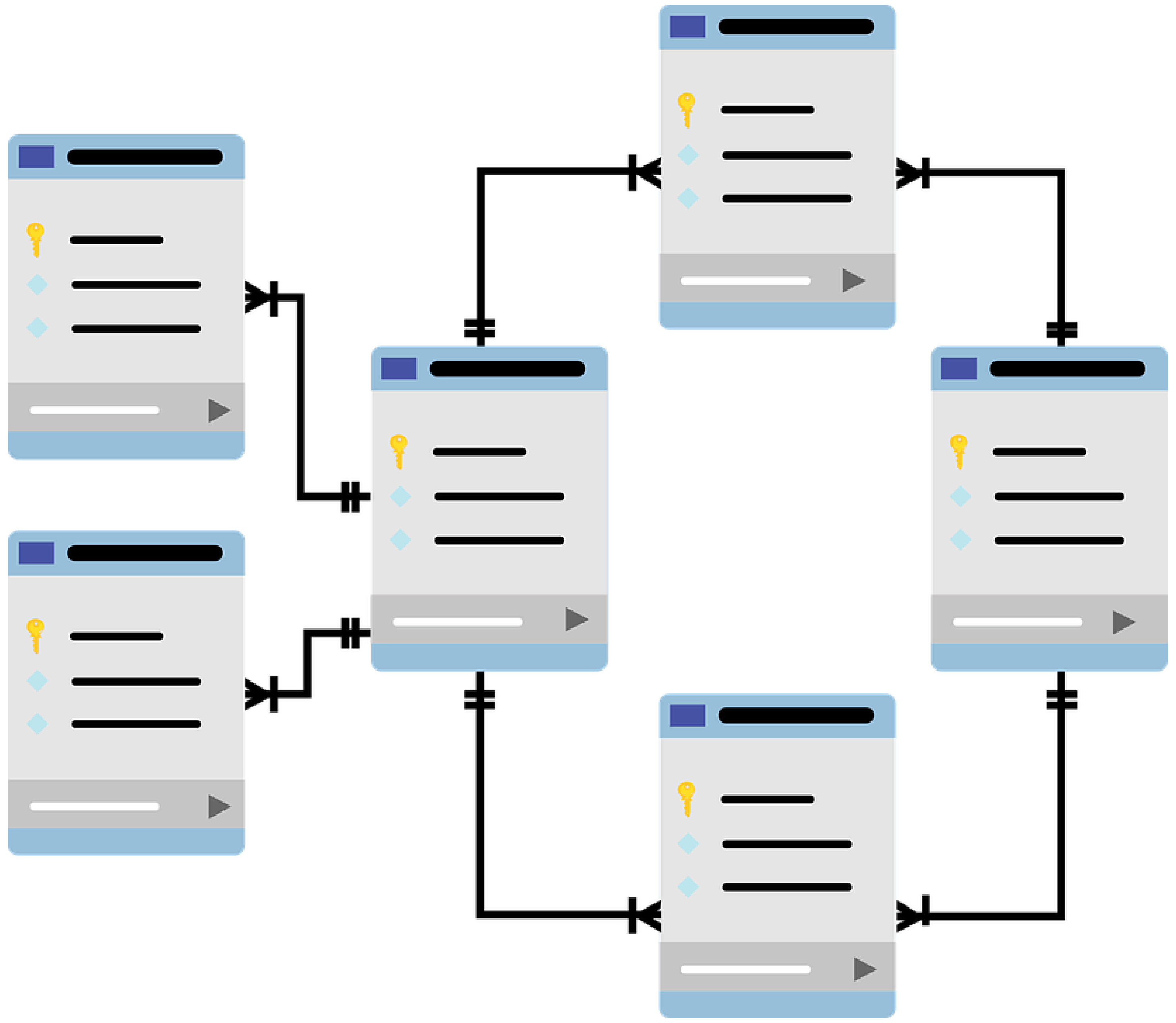
- Native query language of Neo4j is Cypher (equivalent to SQL, only with more graph):

Graph





RDBS



```
MATCH p=()-[r:FOLLOWS]->() RETURN p
```

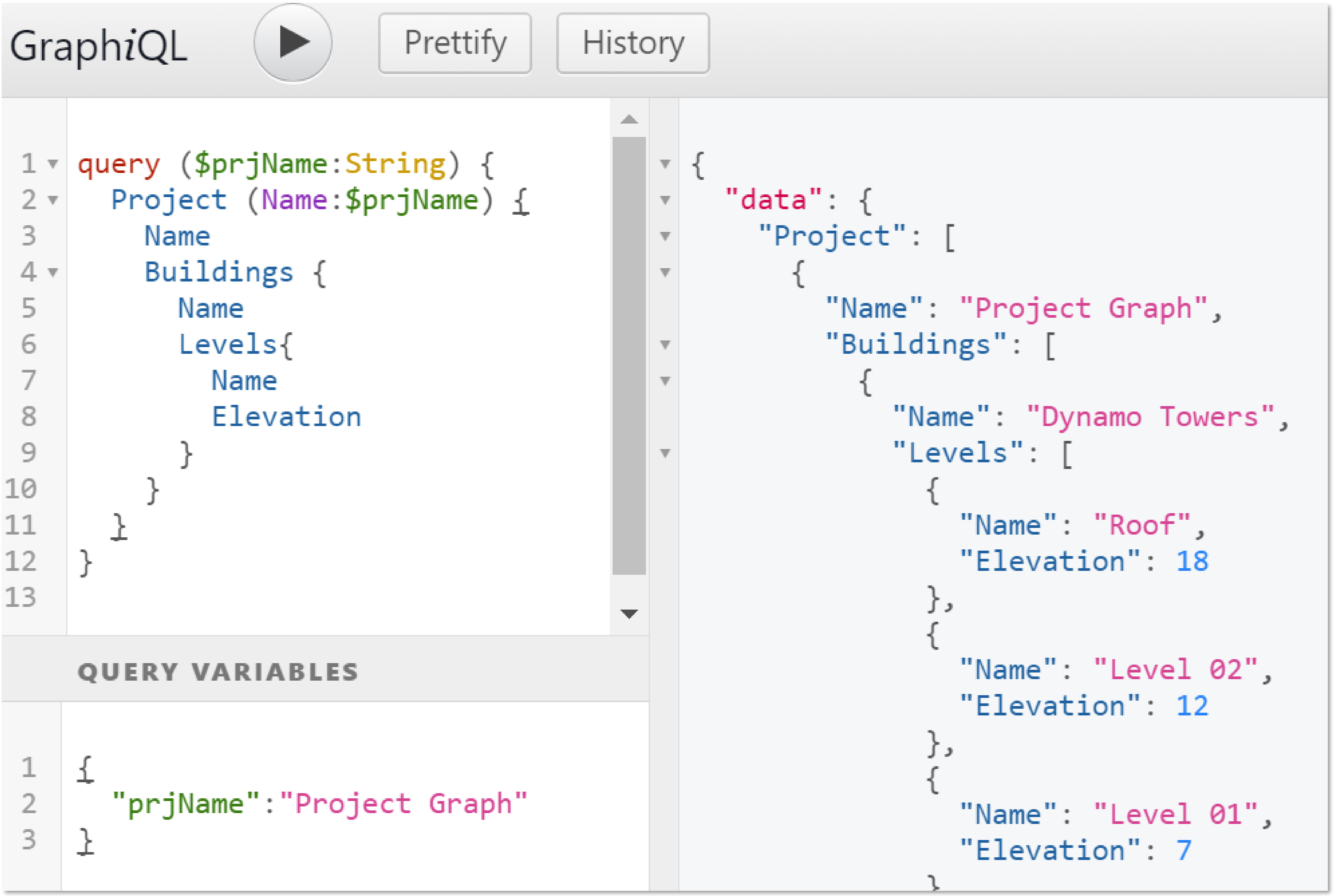
The relationships in data are also: data

What is GraphQL?

https://graphql.org/

https://youtu.be/4wyAcorzbO0

- REST replacement; the future of APIs?
- A single endpoint to rule them all!
- Only returns the data requested
- Built-in introspection API
- The only good thing to come out of Facebook?



```
query ($prjName:String) {
  Project (Name:$prjName) {
    Name
    Buildings {
      Name
      Levels{
        Name
        Elevation
      }
    }
  }
}
```

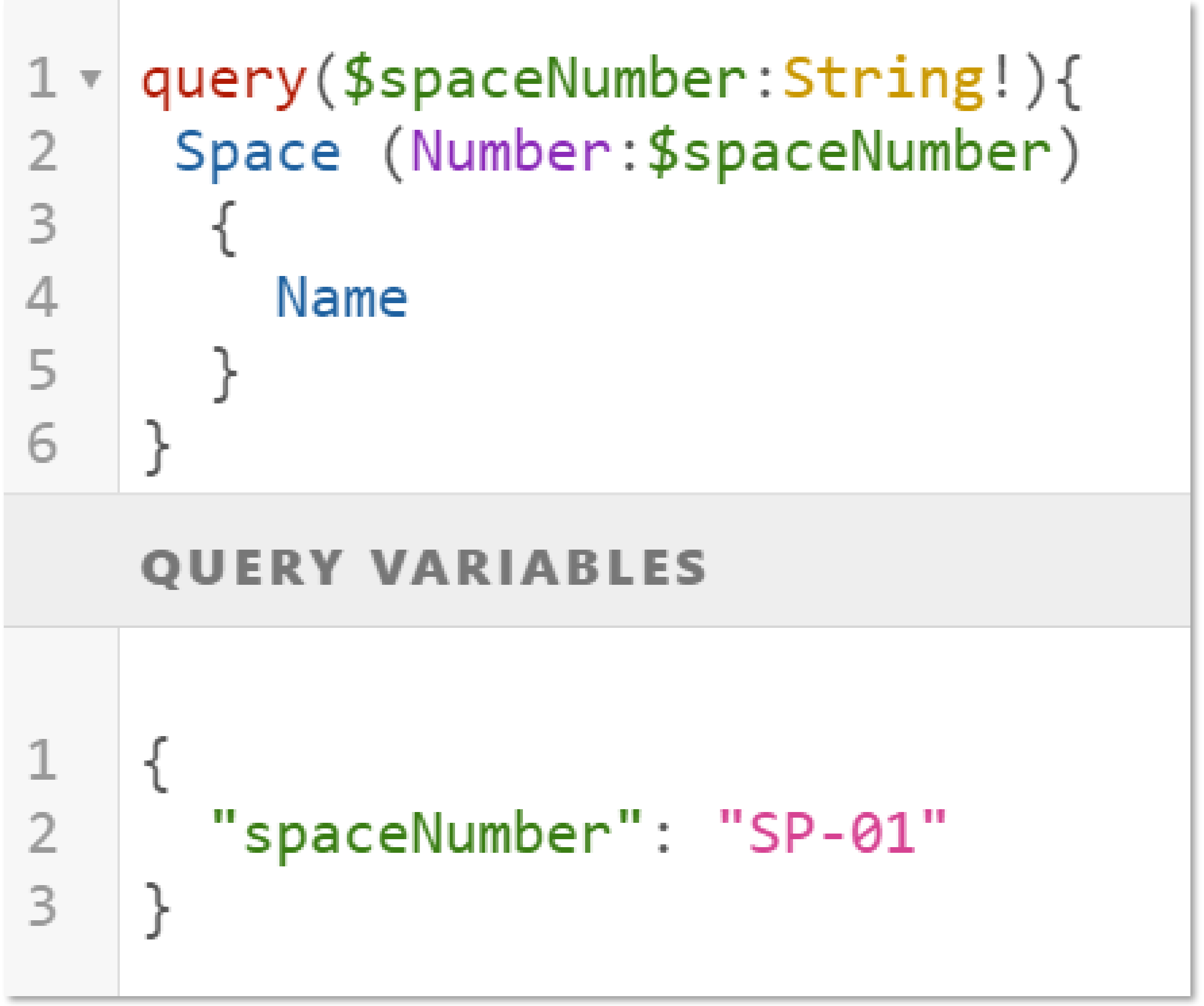
```
{
  "prjName":"Project Graph"
}
```

```
{
  "data": {
    "Project": [
      {
        "Name": "Project Graph",
        "Buildings": [
          {
            "Name": "Dynamo Towers",
            "Levels": [
              {
                "Name": "Roof",
                "Elevation": 18
              },
              {
                "Name": "Level 02",
                "Elevation": 12
              },
              {
                "Name": "Level 01",
                "Elevation": 7
              }
```

https://electronjs.org/apps/graphiql

Sending GraphQL requests

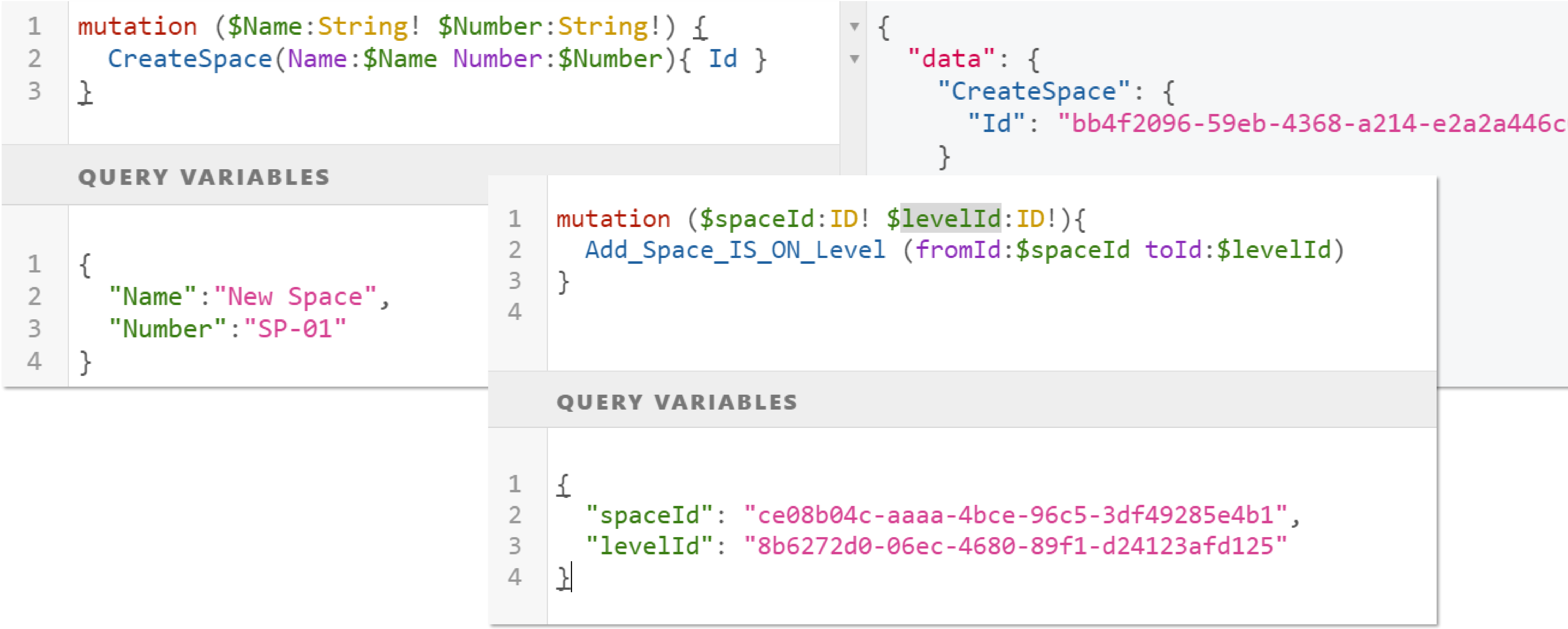
- Usually a HTTP GET or POST request with JSON body, see: https://graphql.org/learn/serving-over-http/
- Use Postman or GraphiQL to try it out
- Many GraphQL libraries are available for popular languages



```
query($spaceNumber:String!){
 Space (Number:$spaceNumber)
  {
    Name
  }
}
```

QUERY VARIABLES

```
{
  "spaceNumber": "SP-01"
}
```



```
mutation ($Name:String! $Number:String!) {
  CreateSpace(Name:$Name Number:$Number){ Id }
}
```

QUERY VARIABLES

```
{
  "Name":"New Space",
  "Number":"SP-01"
}
```

```
{
  "data": {
    "CreateSpace": {
      "Id": "bb4f2096-59eb-4368-a214-e2a2a446c
    }
```

```
mutation ($spaceId:ID! $levelId:ID!){
  Add_Space_IS_ON_Level (fromId:$spaceId toId:$levelId)
}
```

QUERY VARIABLES

```
{
  "spaceId": "ce08b04c-aaaa-4bce-96c5-3df49285e4b1",
  "levelId": "8b6272d0-06ec-4680-89f1-d24123afd125"
}
```

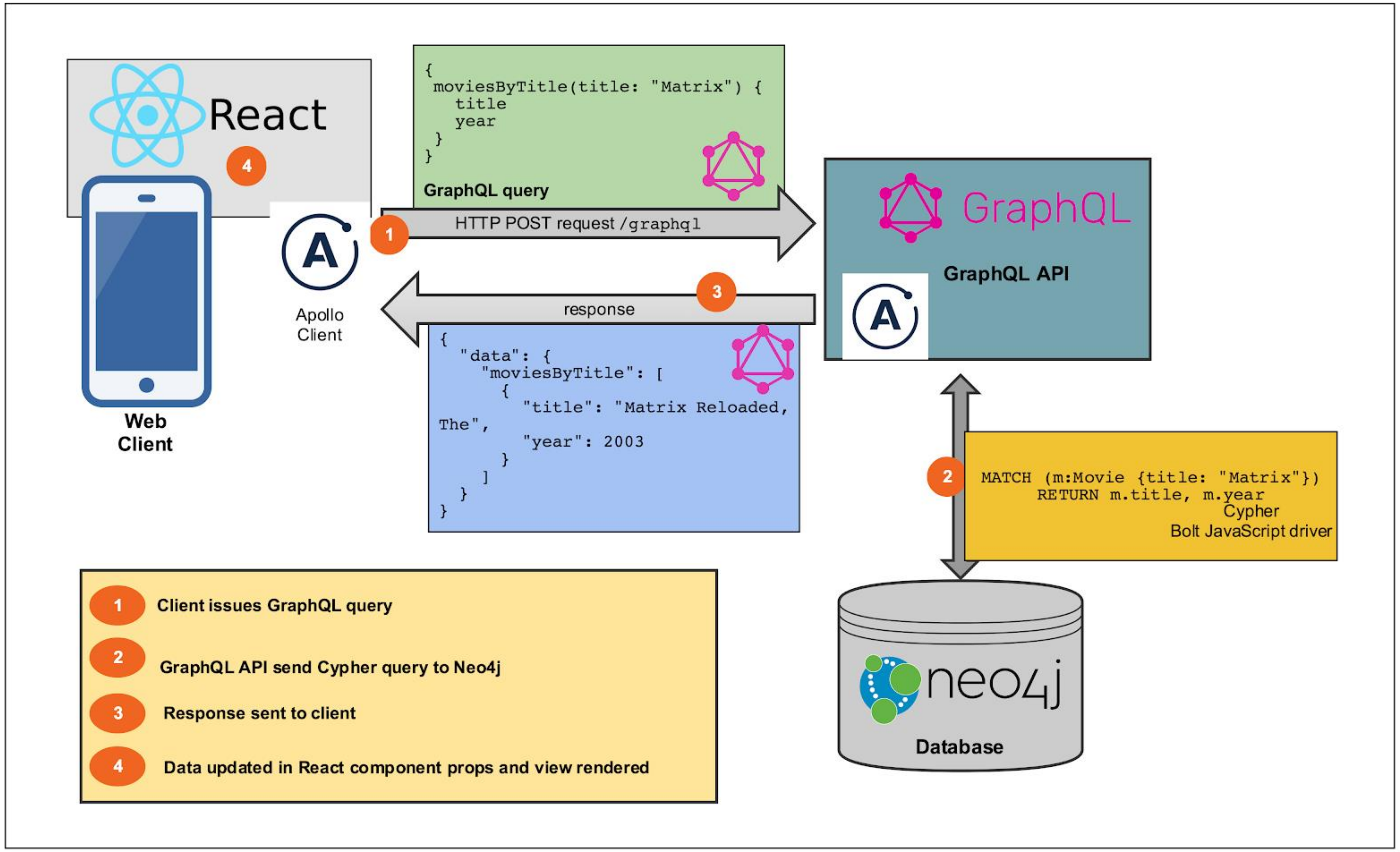
or a
lata
THEN
Graph Data
Recap
THEN
Building
Graph API
THEN
A journ
through
API

GRANDstack
https://grandstack.io

Building Graph API is based on:

GraphQL React Apollo Neo4j

- WEB Stack : Apollo running in Node.js and Express
- Create, Read, Update, Delete Auto generation from the GraphQL Schema
- Easy API : reduced need for boiler plate code (e.g. no MVC pattern).
- Seamless integration with Neo4j, and many other databases, authentication services, and middleware.



```
{
 moviesByTitle(title: "Matrix") {
   title
   year
 }
}
```

GraphQL query

```
{
  "data": {
    "moviesByTitle": [
      {
        "title": "Matrix Reloaded,
The",
        "year": 2003
      }
    ]
  }
}
```

```
MATCH (m:Movie {title: "Matrix"})
     RETURN m.title, m.year
```

Cypher
Bolt JavaScript driver

1 Client issues GraphQL query

2 GraphQL API send Cypher query to Neo4j

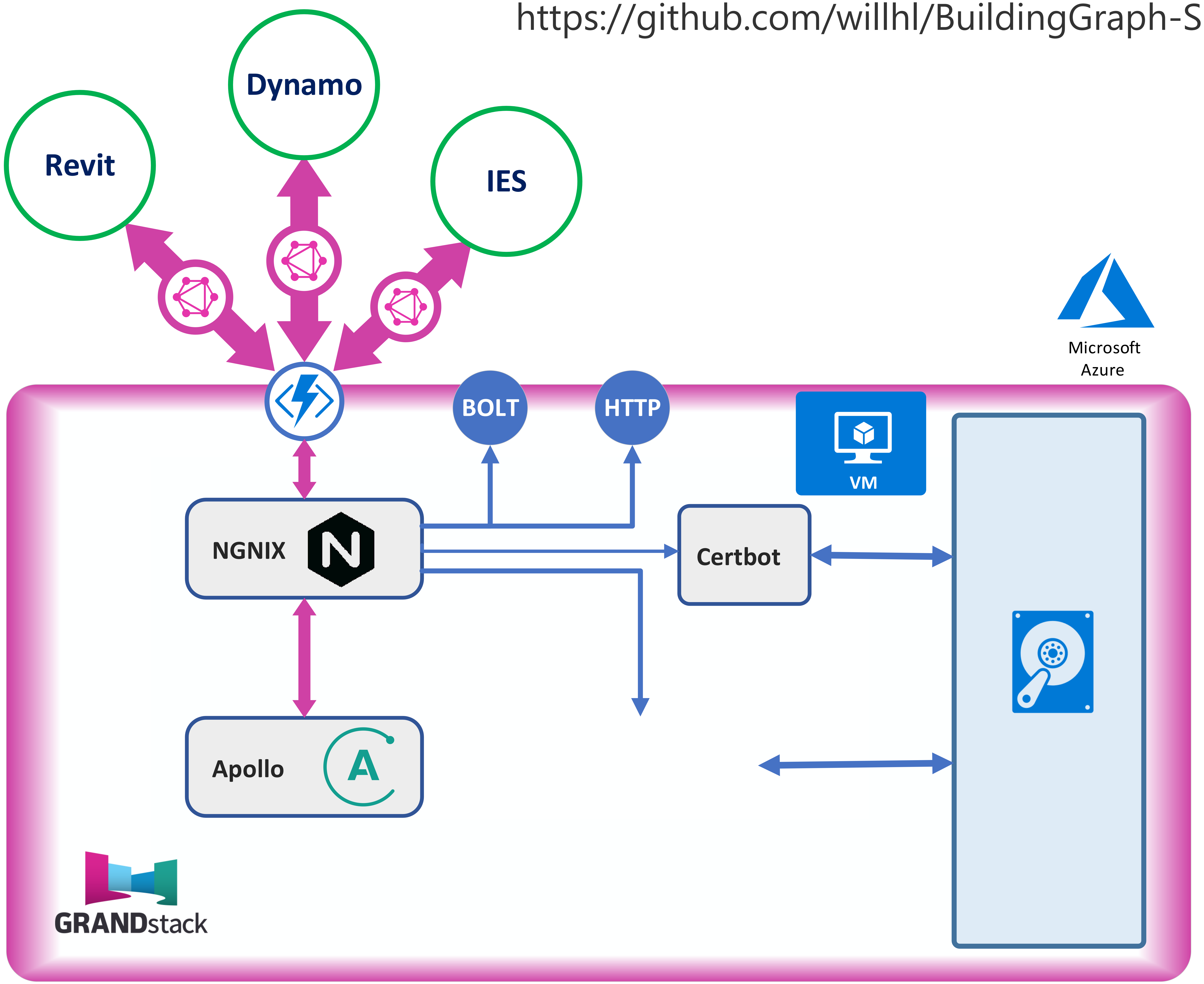
3 Response sent to client

4 Data updated in React component props and view rendered

Building Graph API Architecture

https://github.com/willhl/BuildingGraph-Server

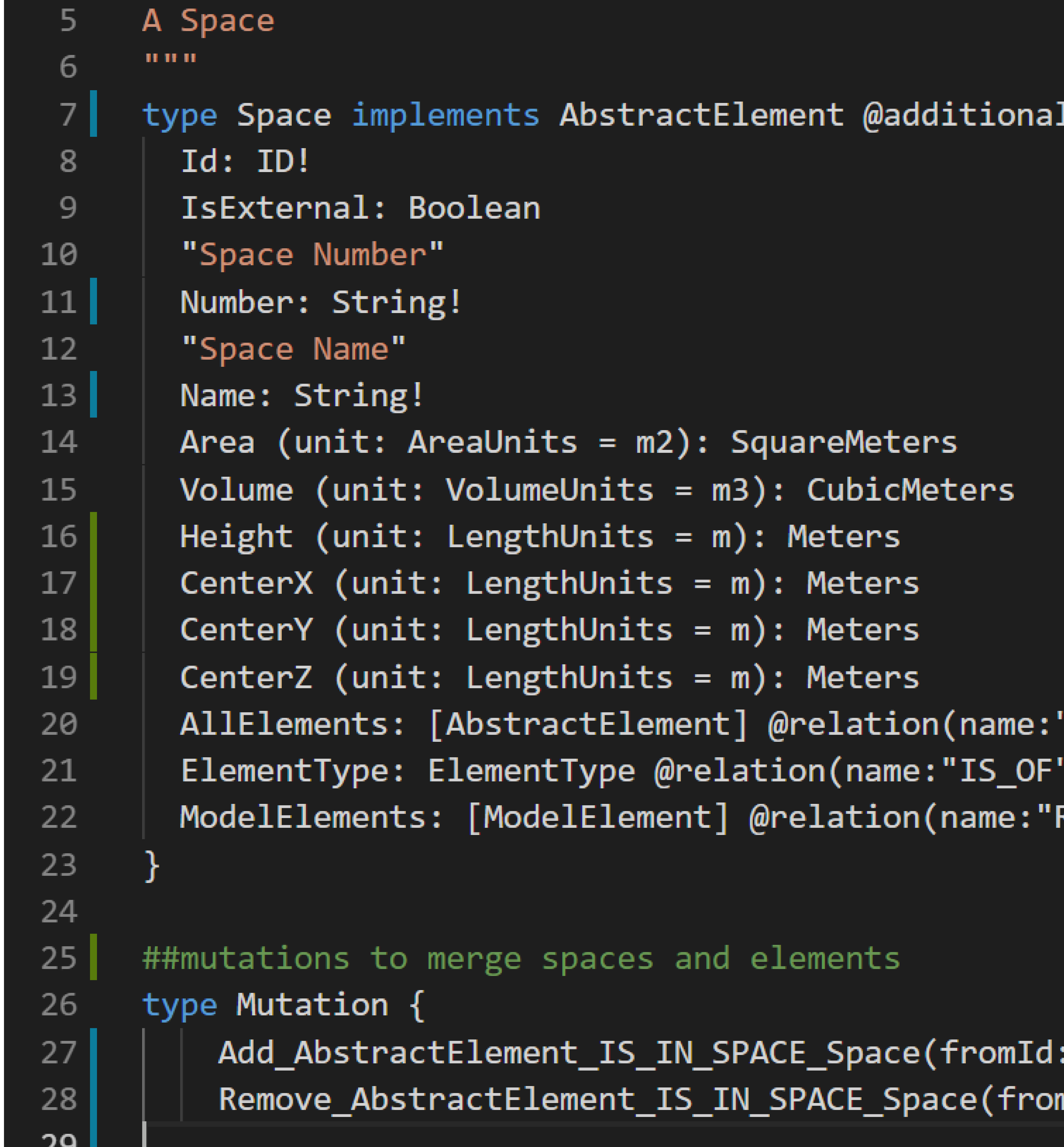
- Azure Ubuntu VM with docker installed
- Apollo + Neo4j running in containers
- Azure Function acts as secure endpoint
- Neo4j Data stored outside of container
- NGNIX used to route traffic to containers



It all starts from the GraphQL schema

https://graphql.org/learn/schema/

- Defined in .graphql files
- **Object types** – the entities or classes, such as Ducts, Spaces, or Models
- **Fields** – the property names and data types available on each type
- **Scalar Types –** The primitive (int, string, etc.) or custom data type
- **Arguments** – Additional data which can be passed to a field
- **Queries** – Predefined queries
- **Mutations** - Functions which change the data, such as create a Duct, Space or model entity

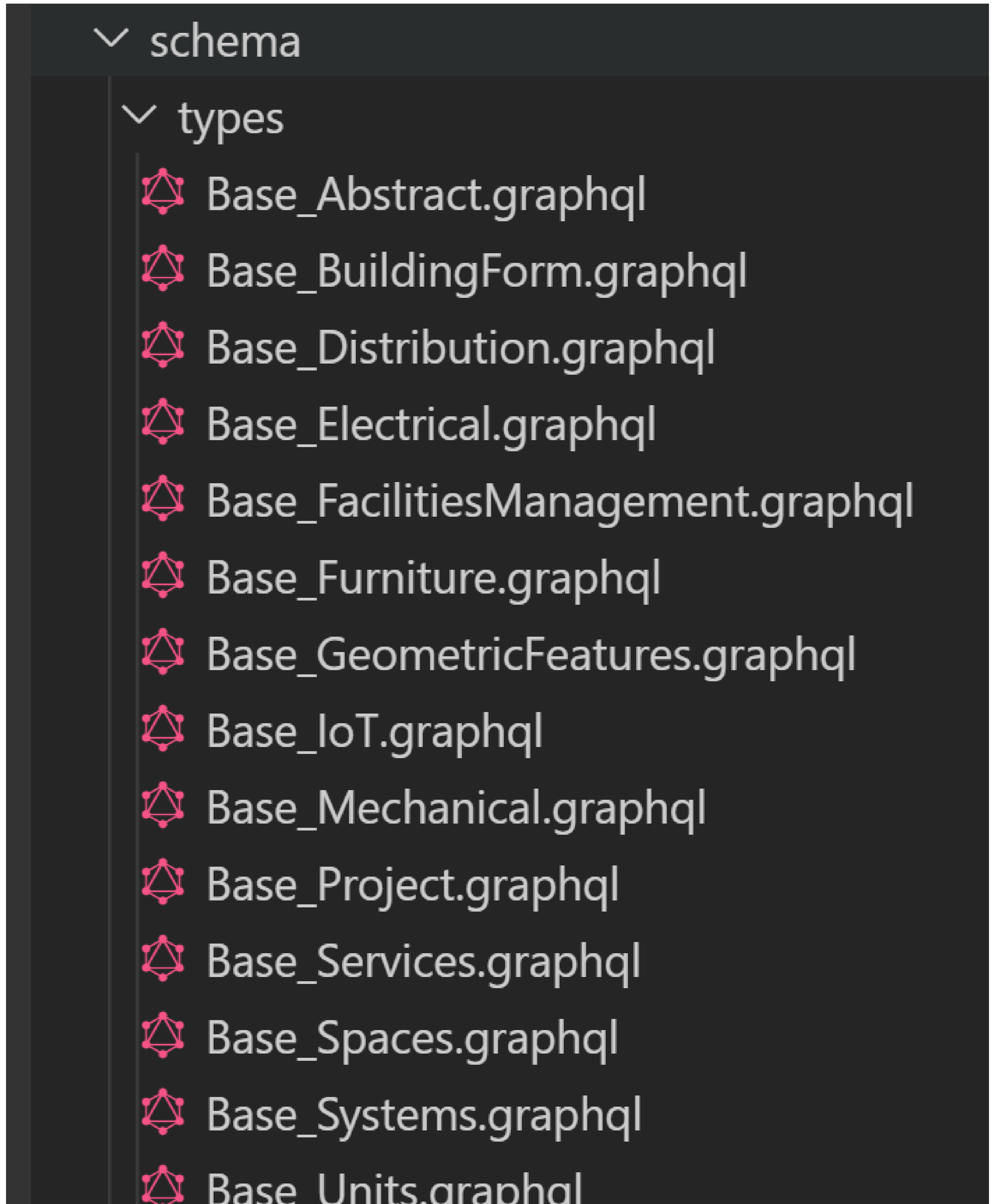


```
A Space
"""
type Space implements AbstractElement @additional
  Id: ID!
  IsExternal: Boolean
  "Space Number"
  Number: String!
  "Space Name"
  Name: String!
  Area (unit: AreaUnits = m2): SquareMeters
  Volume (unit: VolumeUnits = m3): CubicMeters
  Height (unit: LengthUnits = m): Meters
  CenterX (unit: LengthUnits = m): Meters
  CenterY (unit: LengthUnits = m): Meters
  CenterZ (unit: LengthUnits = m): Meters
  AllElements: [AbstractElement] @relation(name:'
  ElementType: ElementType @relation(name:"IS_OF'
  ModelElements: [ModelElement] @relation(name:"F
}

##mutations to merge spaces and elements
type Mutation {
    Add_AbstractElement_IS_IN_SPACE_Space(fromId:
    Remove_AbstractElement_IS_IN_SPACE_Space(from
```

Building Graph schema WIP https://github.com/willhl/BuildingGraph-Server

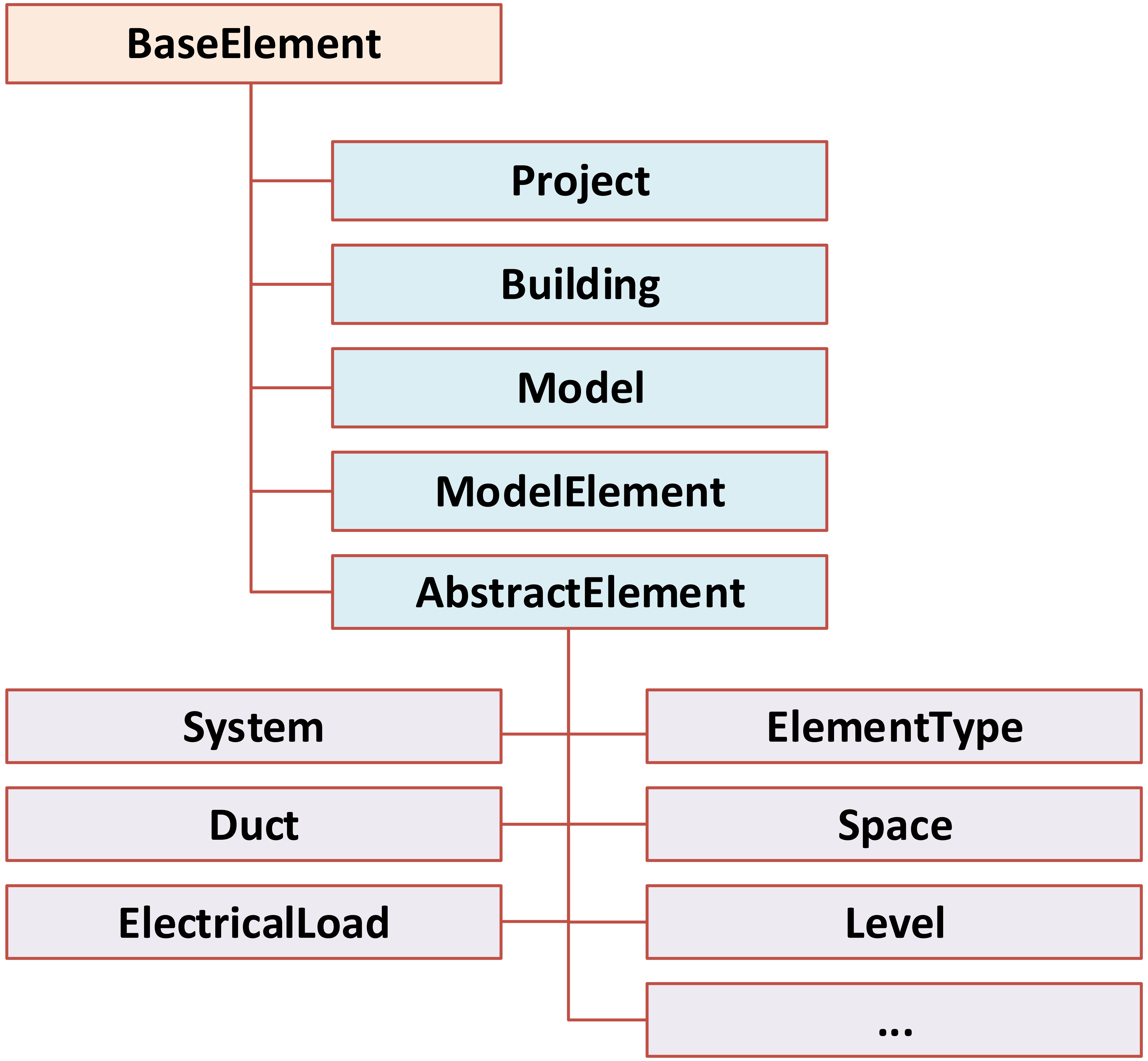
Collection of .graphql in the reop:

- Each file can augment types in other files
- Maintains separation of concerns



```
schema
  types
    Base_Abstract.graphql
    Base_BuildingForm.graphql
    Base_Distribution.graphql
    Base_Electrical.graphql
    Base_FacilitiesManagement.graphql
    Base_Furniture.graphql
    Base_GeometricFeatures.graphql
    Base_IoT.graphql
    Base_Mechanical.graphql
    Base_Project.graphql
    Base_Services.graphql
    Base_Spaces.graphql
    Base_Systems.graphql
    Base_Units.graphql
```

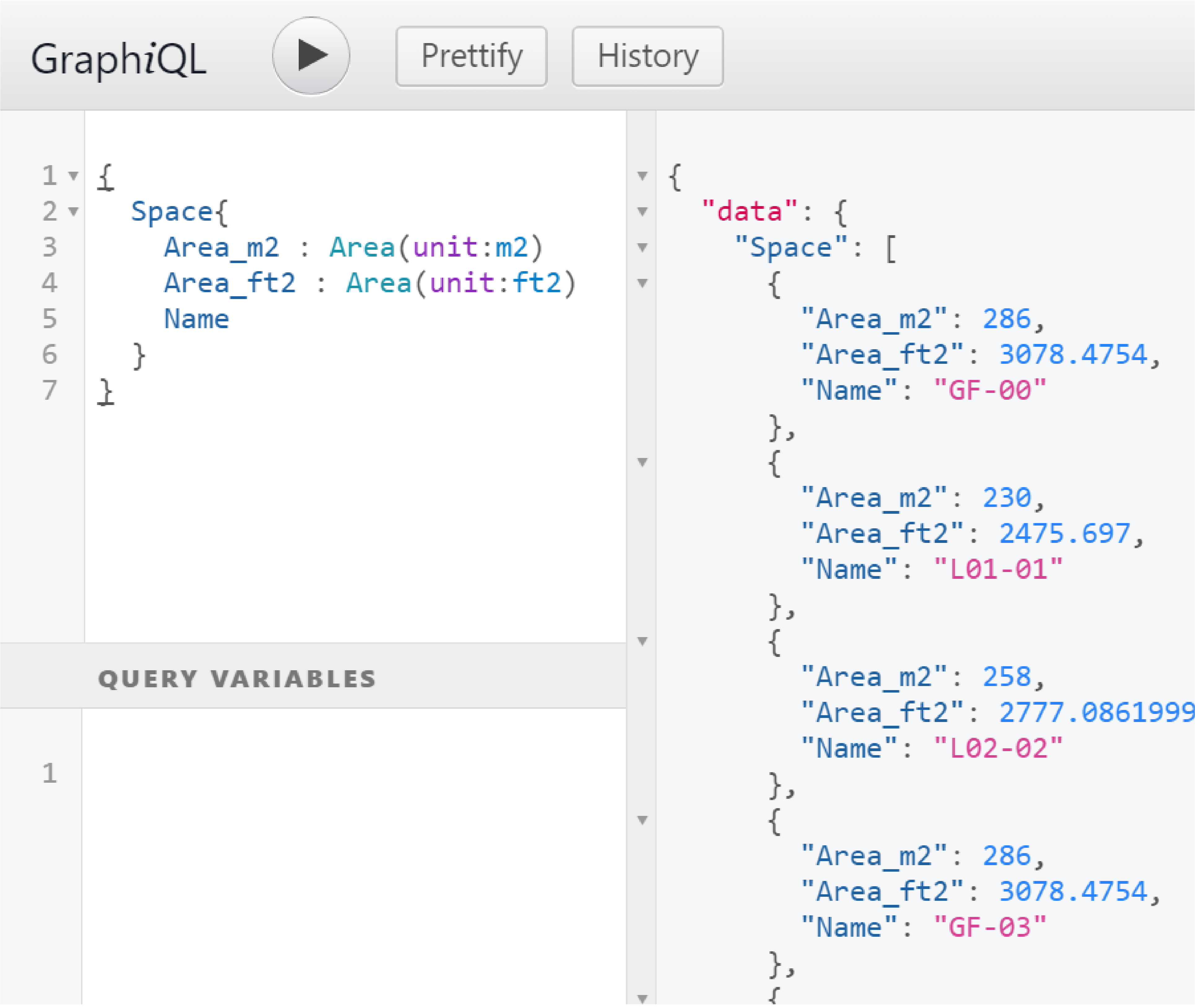
Type hierarchy



- BaseElement
  - Project
  - Building
  - Model
  - ModelElement
  - AbstractElement
    - System
    - Duct
    - ElectricalLoad
    - ElementType
    - Space
    - Level
    - ...

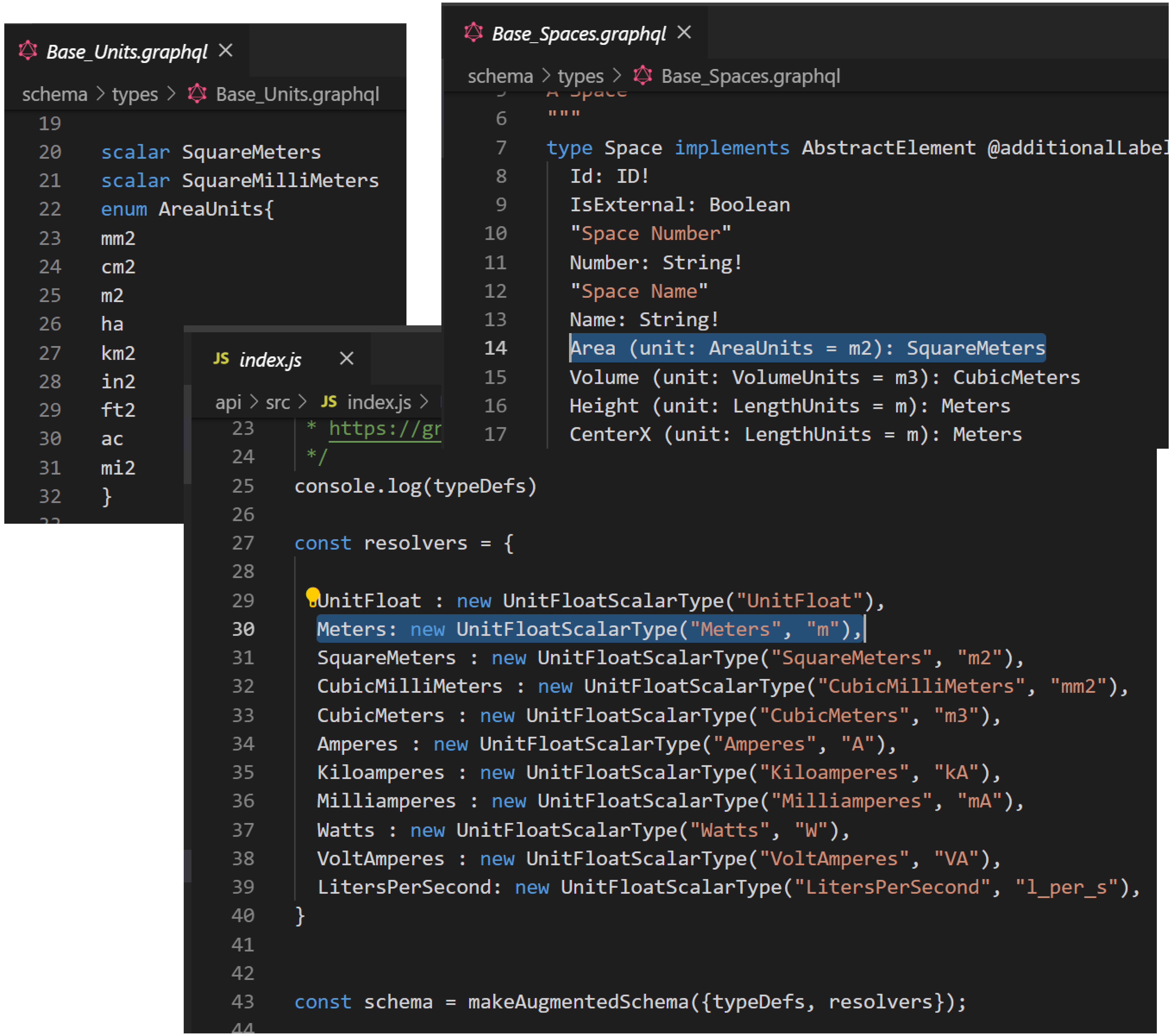
Support for specific data units

Apollo + GraphQL allows for custom scalars
A custom resolver takes care of unit conversion



```
{
  Space{
    Area_m2 : Area(unit:m2)
    Area_ft2 : Area(unit:ft2)
    Name
  }
}
```

```
{
  "data": {
    "Space": [
      {
        "Area_m2": 286,
        "Area_ft2": 3078.4754,
        "Name": "GF-00"
      },
      {
        "Area_m2": 230,
        "Area_ft2": 2475.697,
        "Name": "L01-01"
      },
      {
        "Area_m2": 258,
        "Area_ft2": 2777.0861999
        "Name": "L02-02"
      },
      {
        "Area_m2": 286,
        "Area_ft2": 3078.4754,
        "Name": "GF-03"
      },
      {
```


Base_Units.graphql
schema > types > Base_Units.graphql
Base_Spaces.graphql
schema > types > Base_Spaces.graphql
index.js
api > src > index.js

```
scalar SquareMeters
scalar SquareMilliMeters
enum AreaUnits{
mm2
cm2
m2
ha
km2
in2
ft2
ac
mi2
}
```

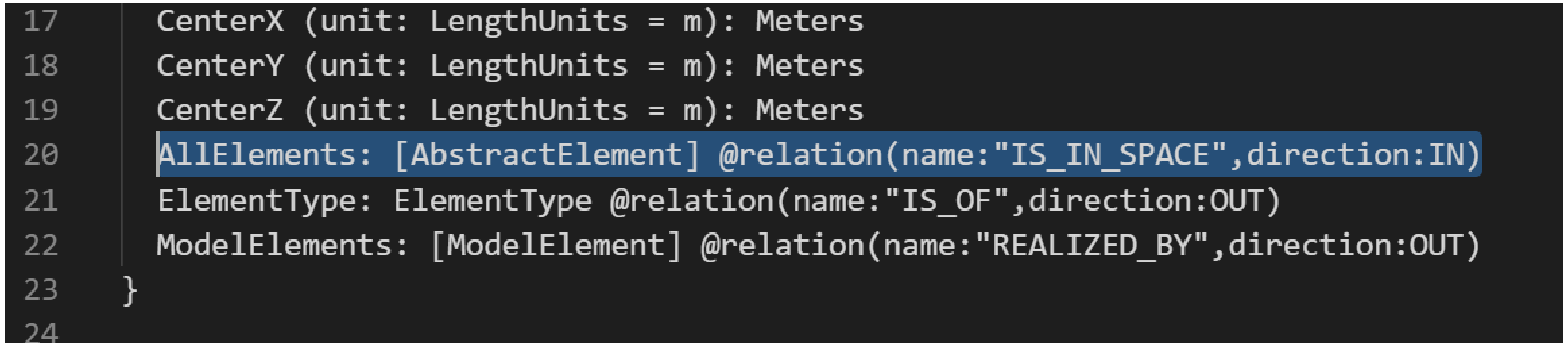
```
A Space
"""
type Space implements AbstractElement @additionalLabe
  Id: ID!
  IsExternal: Boolean
  "Space Number"
  Number: String!
  "Space Name"
  Name: String!
  Area (unit: AreaUnits = m2): SquareMeters
  Volume (unit: VolumeUnits = m3): CubicMeters
  Height (unit: LengthUnits = m): Meters
  CenterX (unit: LengthUnits = m): Meters
```

```
 * https://gr
 */
console.log(typeDefs)

const resolvers = {

  UnitFloat : new UnitFloatScalarType("UnitFloat"),
  Meters: new UnitFloatScalarType("Meters", "m"),
  SquareMeters : new UnitFloatScalarType("SquareMeters", "m2"),
  CubicMilliMeters : new UnitFloatScalarType("CubicMilliMeters", "mm2"),
  CubicMeters : new UnitFloatScalarType("CubicMeters", "m3"),
  Amperes : new UnitFloatScalarType("Amperes", "A"),
  Kiloamperes : new UnitFloatScalarType("Kiloamperes", "kA"),
  Milliamperes : new UnitFloatScalarType("Milliamperes", "mA"),
  Watts : new UnitFloatScalarType("Watts", "W"),
  VoltAmperes : new UnitFloatScalarType("VoltAmperes", "VA"),
  LitersPerSecond: new UnitFloatScalarType("LitersPerSecond", "l_per_s"),
}


const schema = makeAugmentedSchema({typeDefs, resolvers});
```

Neo4j Specific GraphQL Schema Directives

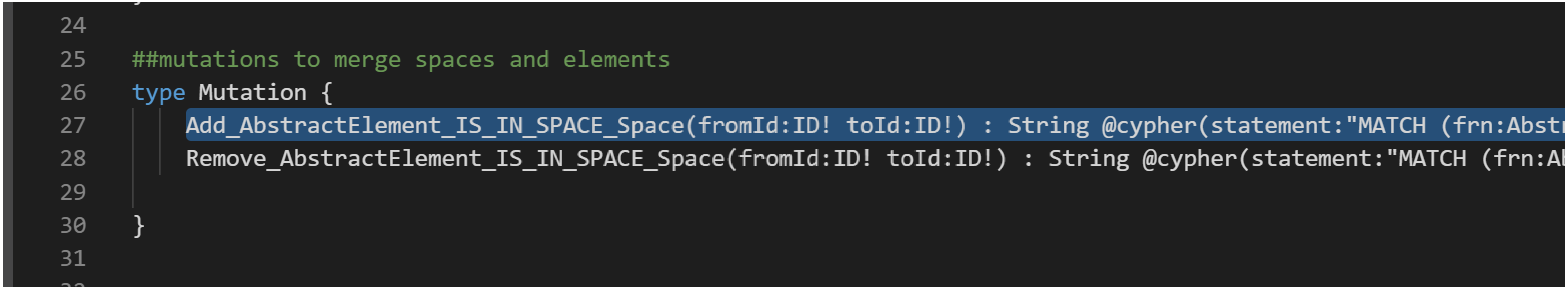
https://grandstack.io/docs/neo4j-graphql-js.html

- @Relation – Specifies which child elements, or parent elements, of a type are supported



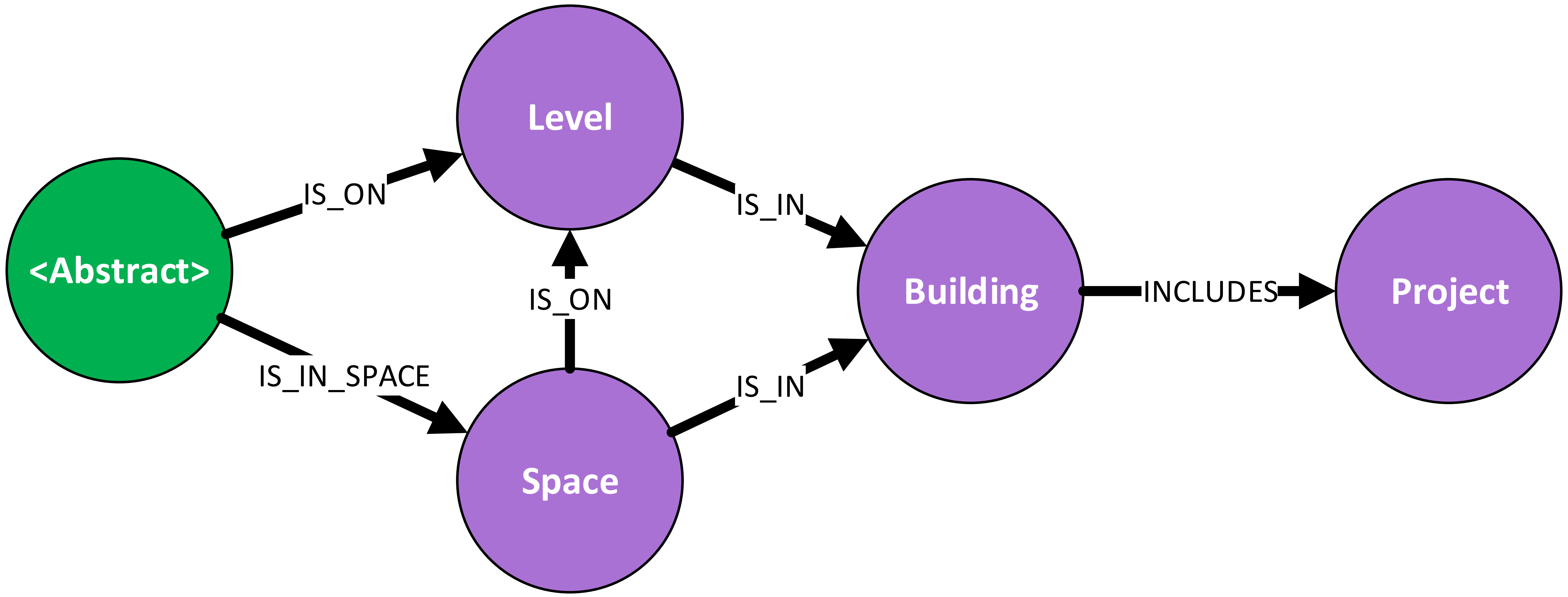
```
    CenterX (unit: LengthUnits = m): Meters
    CenterY (unit: LengthUnits = m): Meters
    CenterZ (unit: LengthUnits = m): Meters
    AllElements: [AbstractElement] @relation(name:"IS_IN_SPACE",direction:IN)
    ElementType: ElementType @relation(name:"IS_OF",direction:OUT)
    ModelElements: [ModelElement] @relation(name:"REALIZED_BY",direction:OUT)
}
```

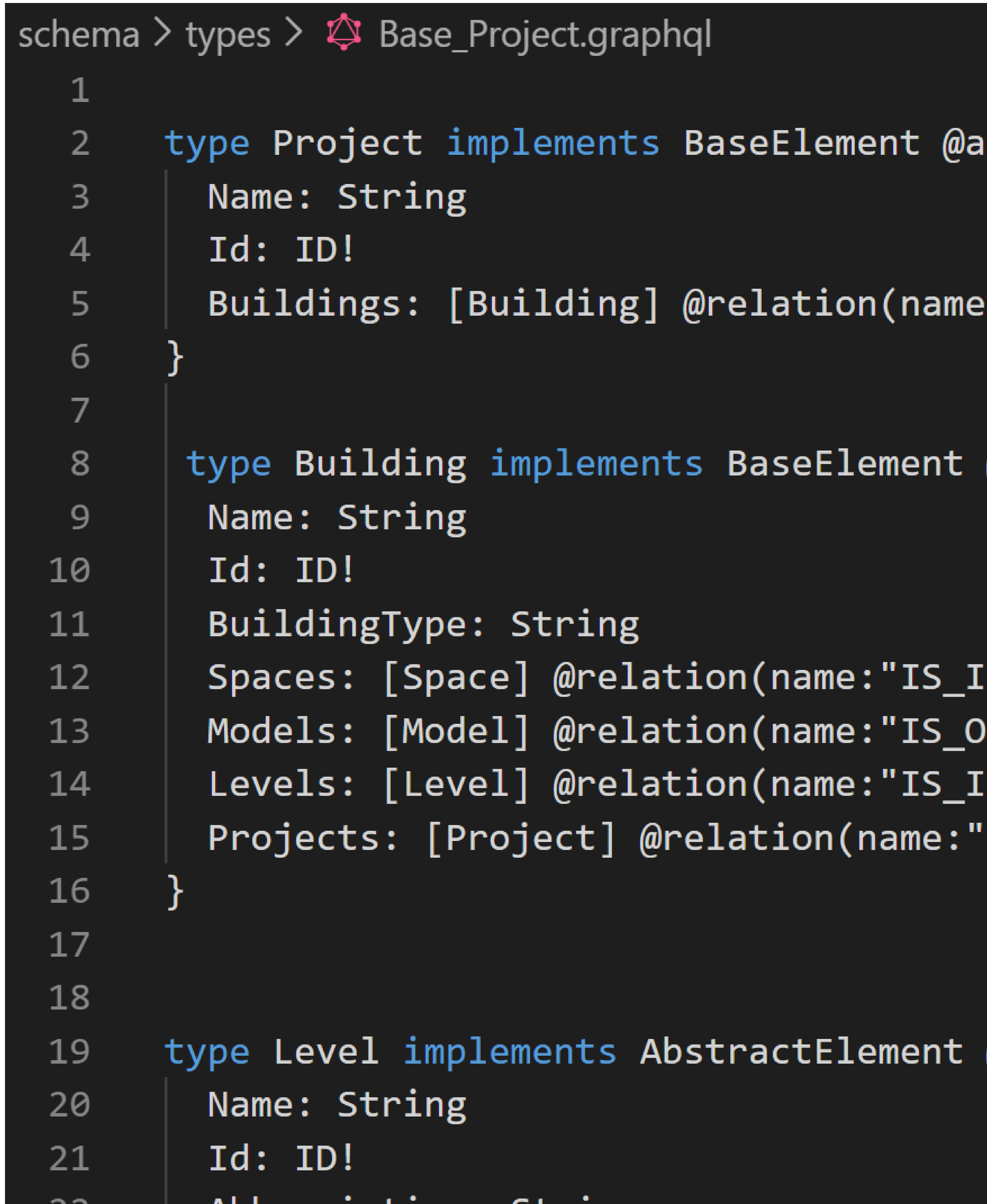
- @Cypher – Specifies a Cypher statement for a query, mutation or field



```

##mutations to merge spaces and elements
type Mutation {
    Add_AbstractElement_IS_IN_SPACE_Space(fromId:ID! toId:ID!) : String @cypher(statement:"MATCH (frn:Abst
    Remove_AbstractElement_IS_IN_SPACE_Space(fromId:ID! toId:ID!) : String @cypher(statement:"MATCH (frn:Al

}
```

Schema: Project, Buildings, Levels and Spaces

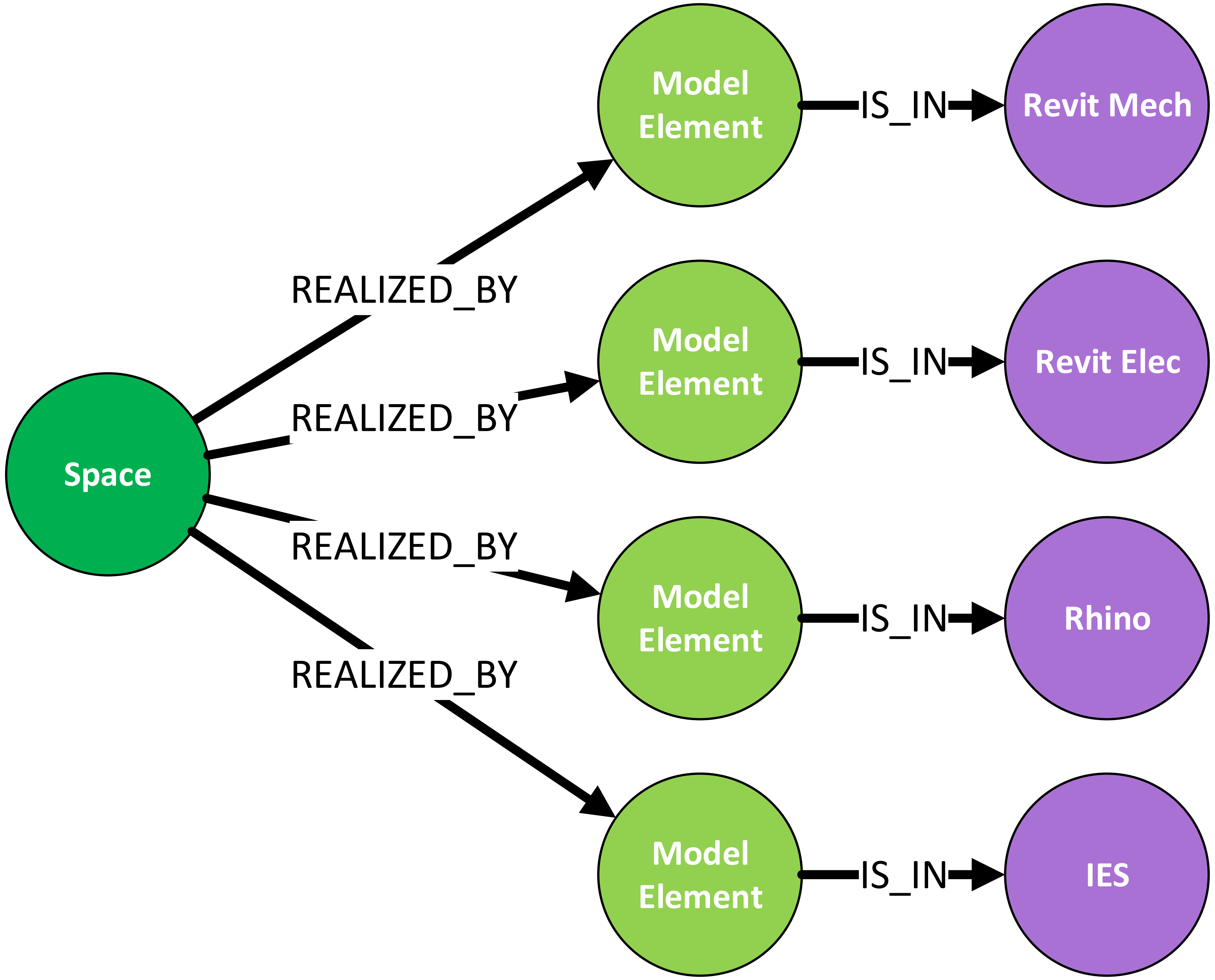




```

type Project implements BaseElement @a
 Name: String
 Id: ID!
 Buildings: [Building] @relation(name
}

 type Building implements BaseElement 
 Name: String
 Id: ID!
 BuildingType: String
 Spaces: [Space] @relation(name:"IS_I
 Models: [Model] @relation(name:"IS_O
 Levels: [Level] @relation(name:"IS_I
 Projects: [Project] @relation(name:"
}


type Level implements AbstractElement 
 Name: String
 Id: ID!
```

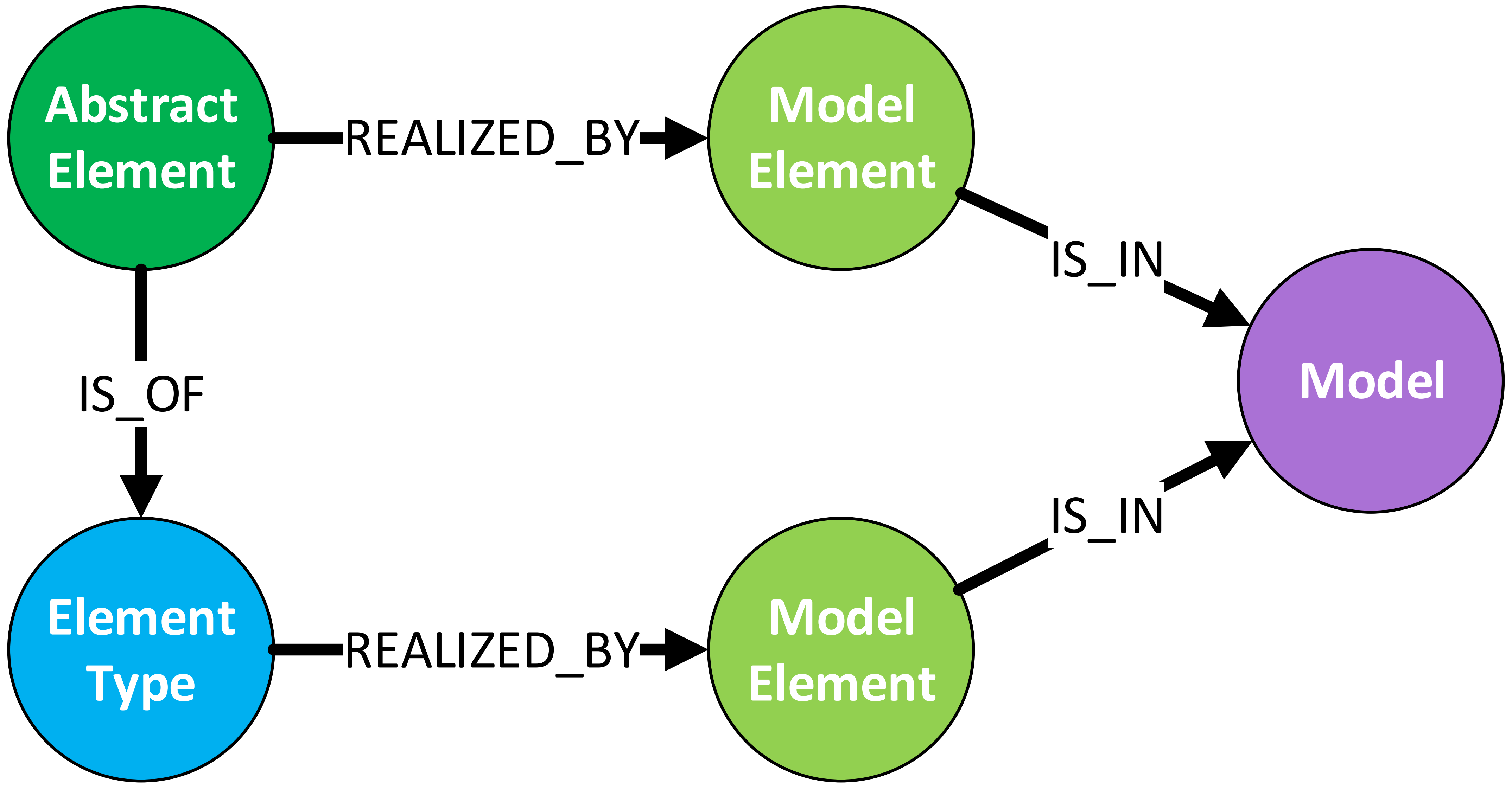
Schema: Abstract Vs Model Elements

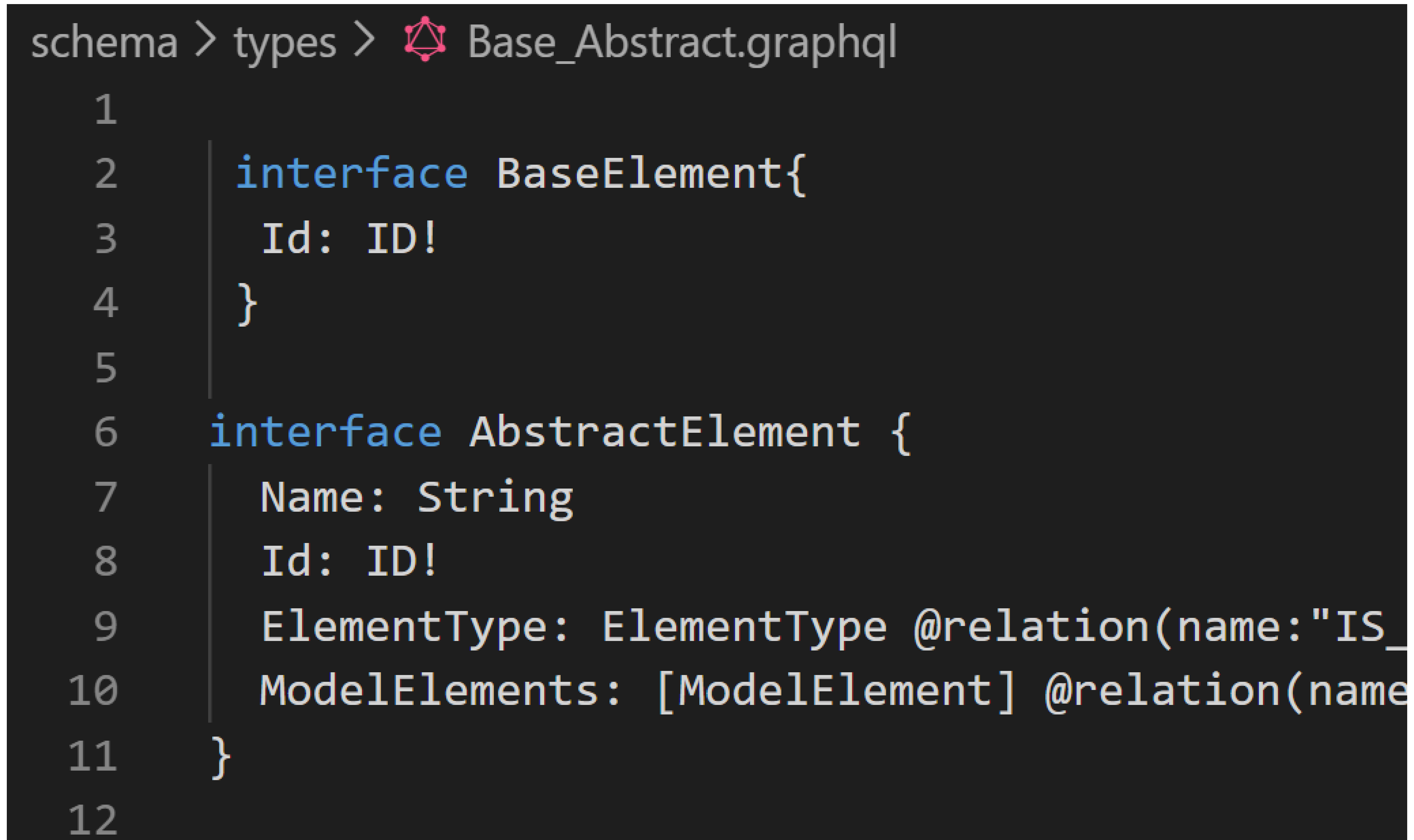
- In this case, the existence of a space is related to many other models
- Applicable to any other abstract element, such as Levels, Zones, Ducts, Outlets, etc.
- Updates on the abstract node can be brought straight into other models
- Models affected by changes can be easily identified



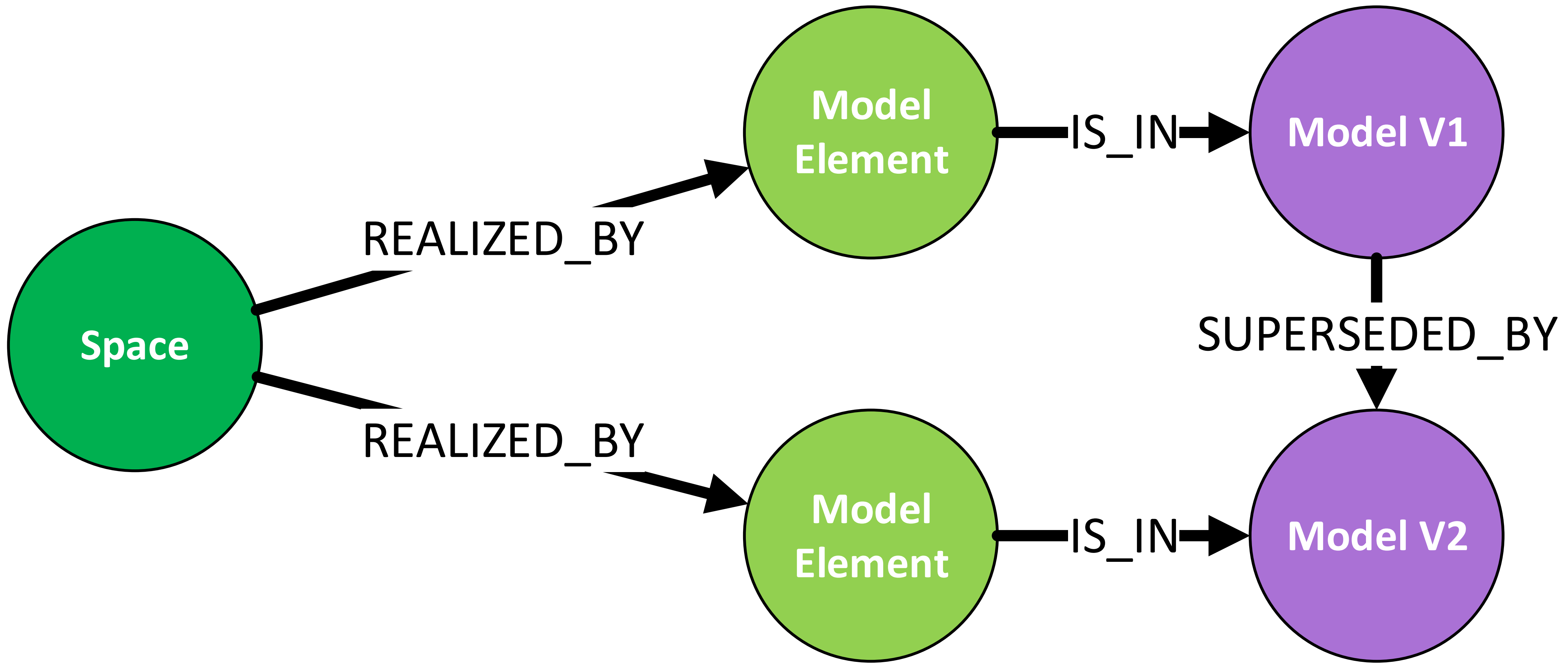
Schema: Abstract Vs Model Elements continued..

- Abstract Elements can exist before a 3D model is developed
- Elements can exist in multiple models
- Changes can be tracked across model versions



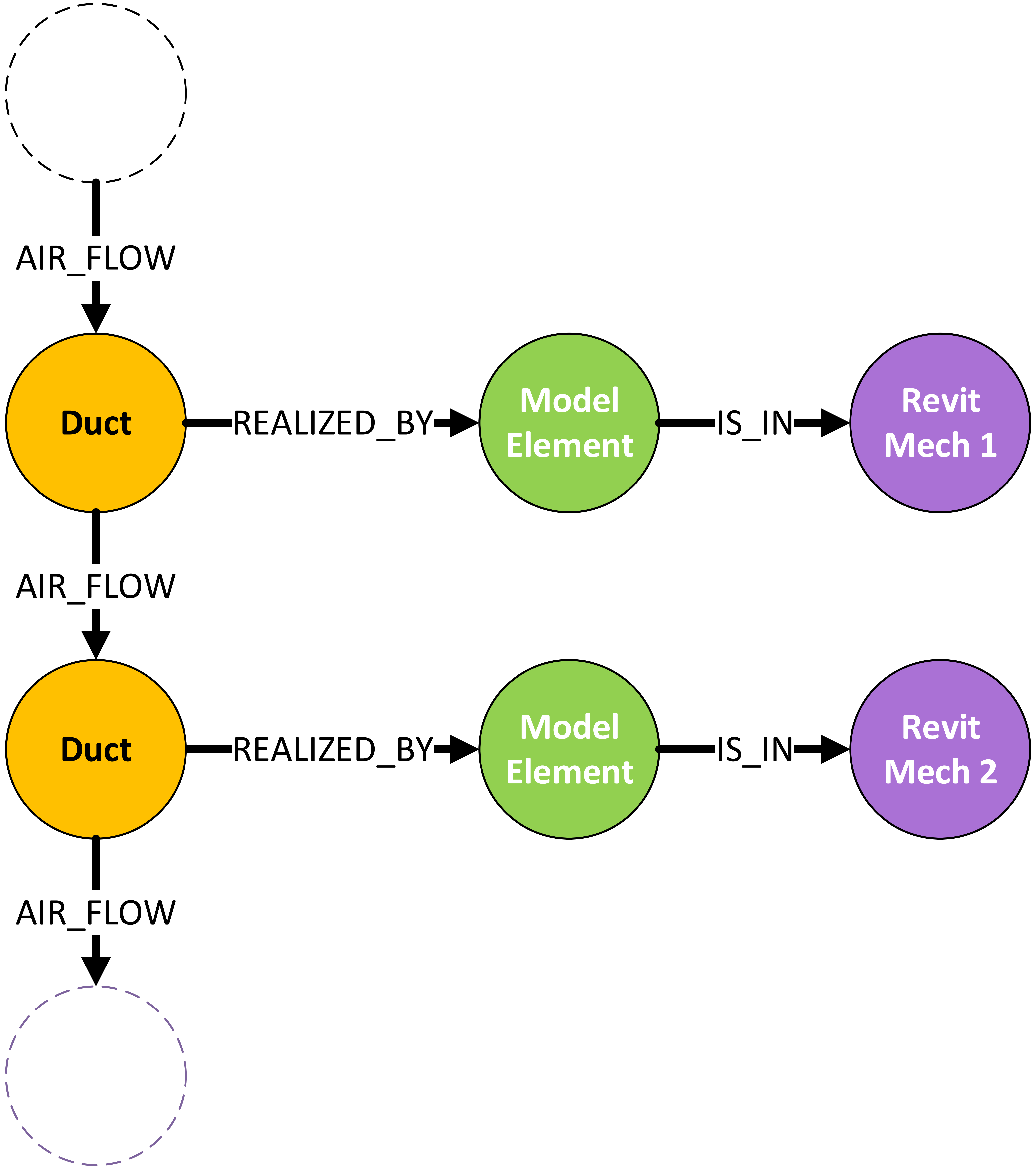


```

 interface BaseElement{
  Id: ID!
 }

interface AbstractElement {
  Name: String
  Id: ID!
  ElementType: ElementType @relation(name:"IS_
  ModelElements: [ModelElement] @relation(name
}

```

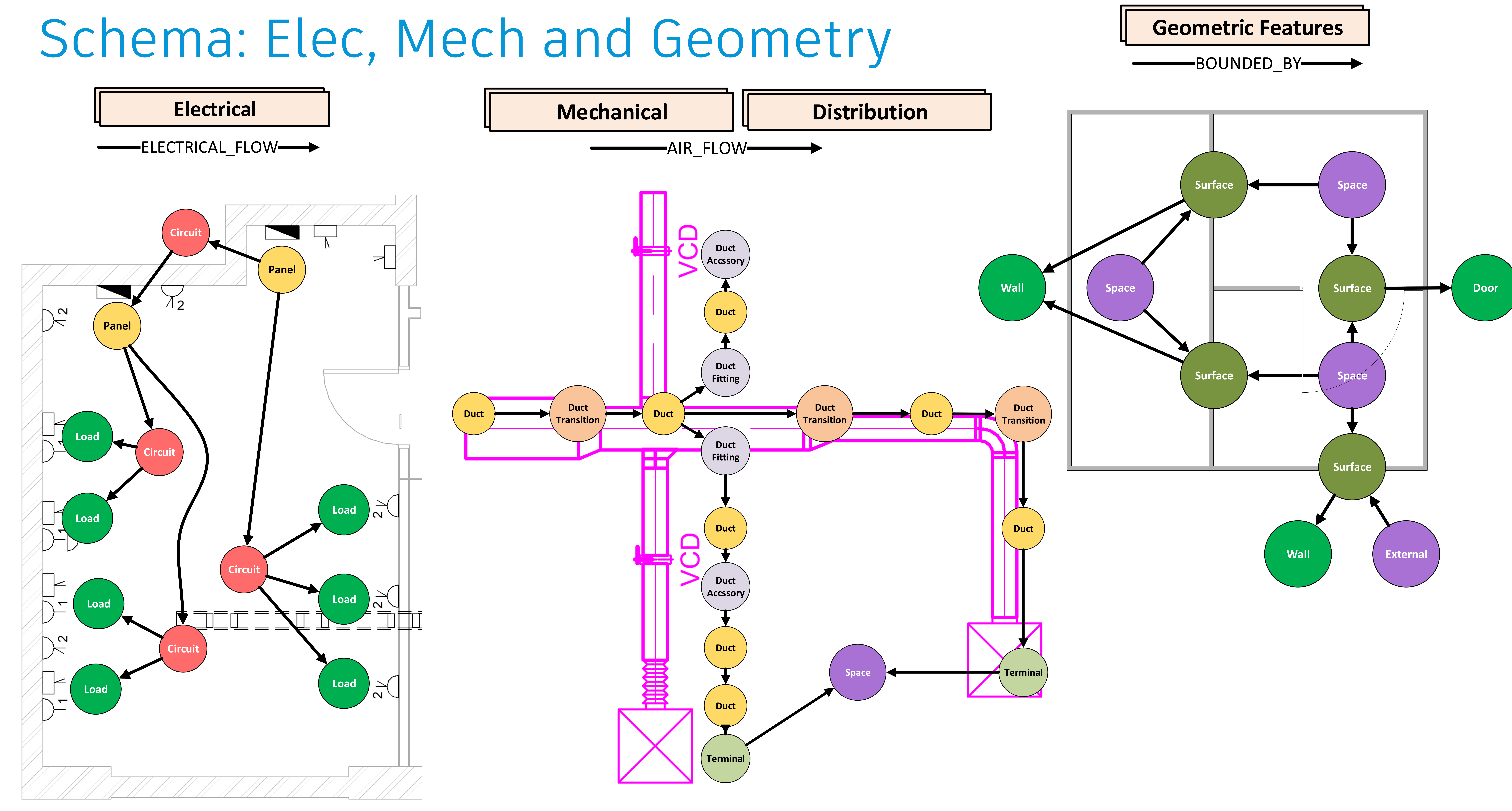


Schema: Abstract Elements… yet more

- Design information is not constrained by what is possible within the models
- Any relationships, such as Air flow, can cross between models
- Other relationships are possible, such as putting circuits or wires on cable trays

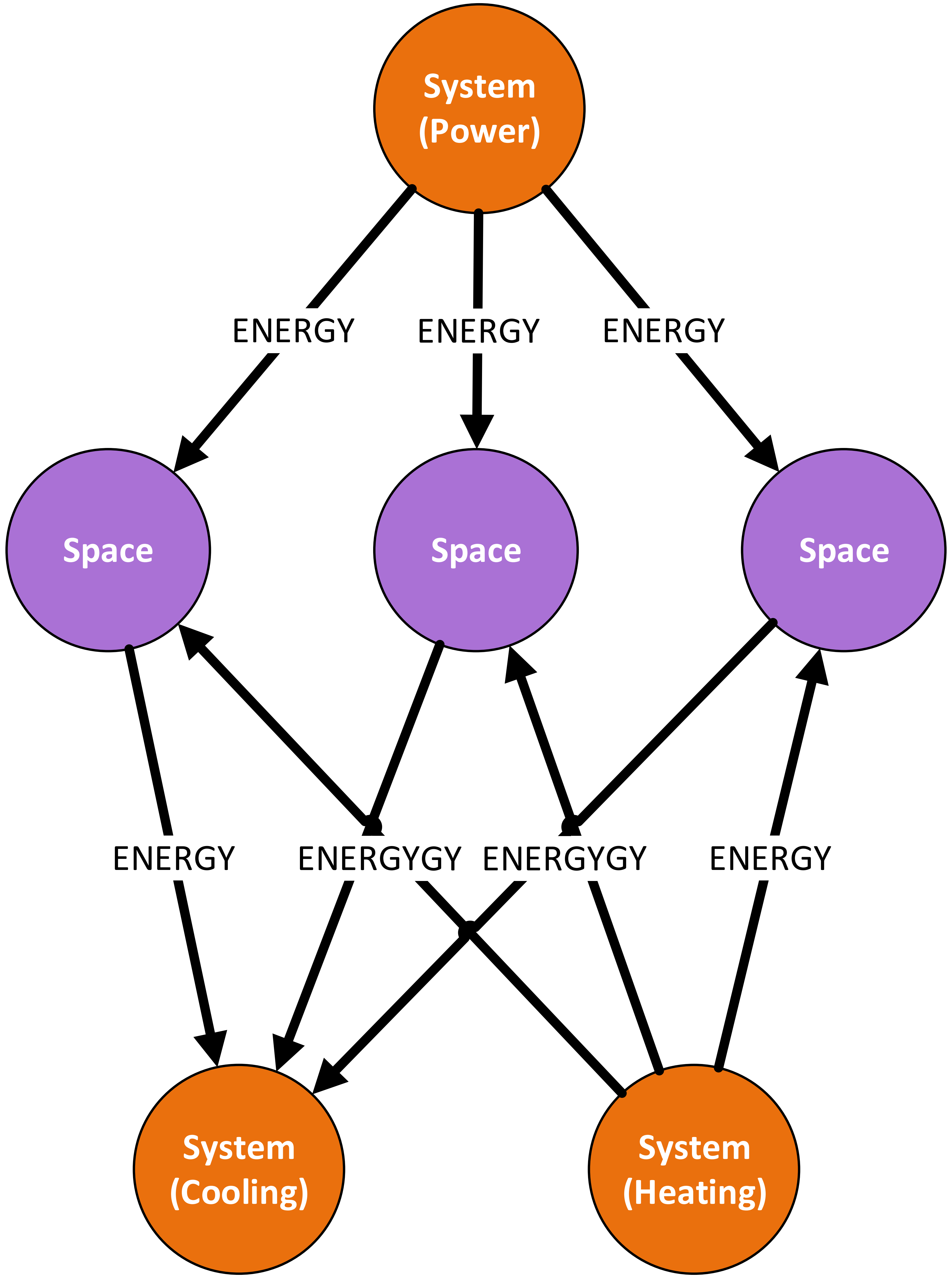


Schema: Elec, Mech and Geometry



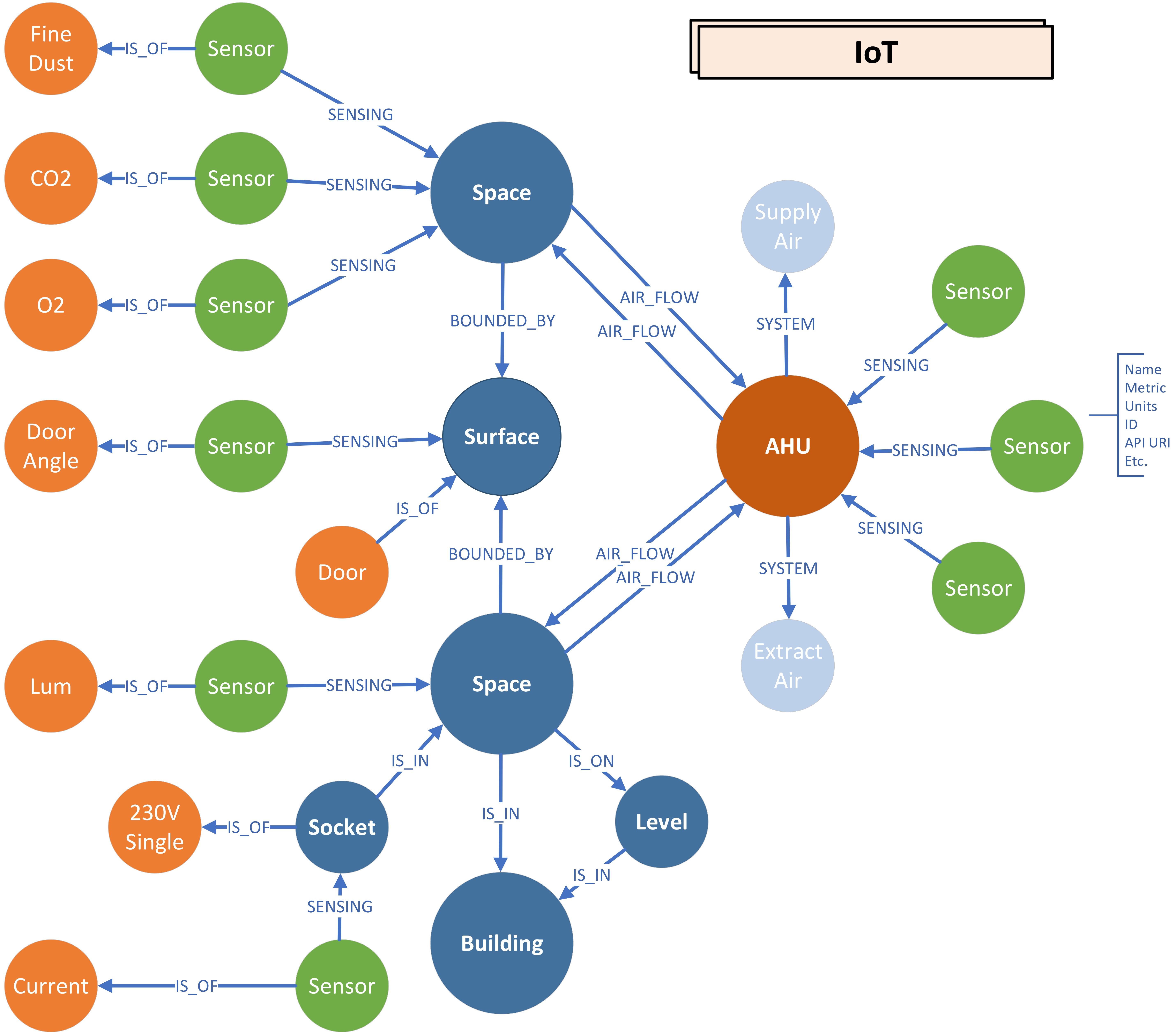
Schema: High level systems

- Represents higher level system relationships
- No need to have a 3D model
- Can be used by other services to perform calculations



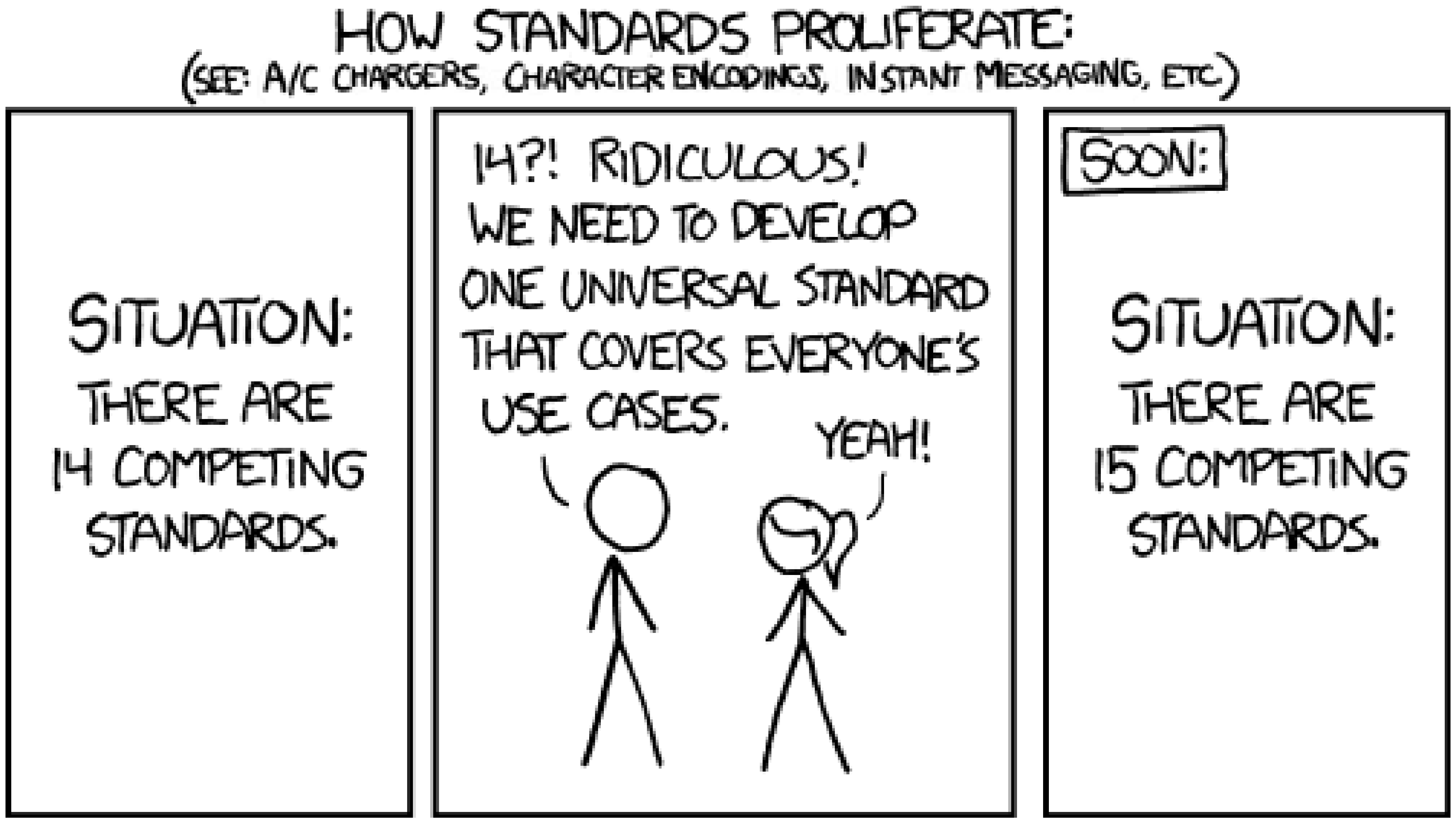
Schema: IoT Sensors

- Doesn’t store the data from each sensor
- Each sensor has its ID or URI pointing to where to get its data
- Full context data is available to derive features for machine learning or analytics.



But wait… we already have IFC, gbXML, BIM360 and others, isn't this just another standard?

- It's primarily a communication schema
- Application agnostic; All you need is a HTTP Client
- Can store abstract elements and non-building elements, before a 3D model is developed
- Can adopt types and parameter naming from any other standard
- Existing apps can still use any standard they choose


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

https://xkcd.com/license.html

HOARE LEA

ph Data Recap

THEN

Building Graph API

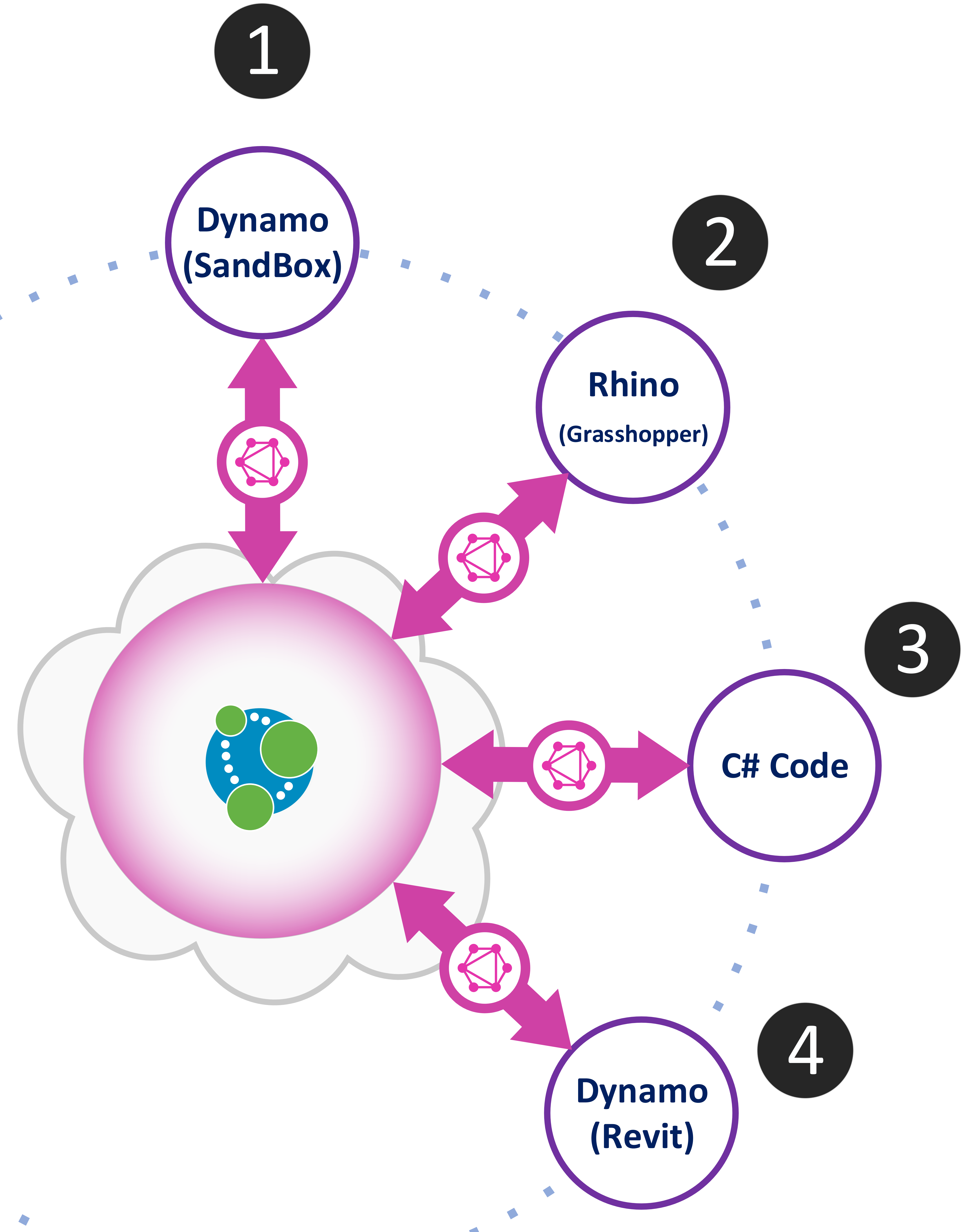
THEN

A journey through the API

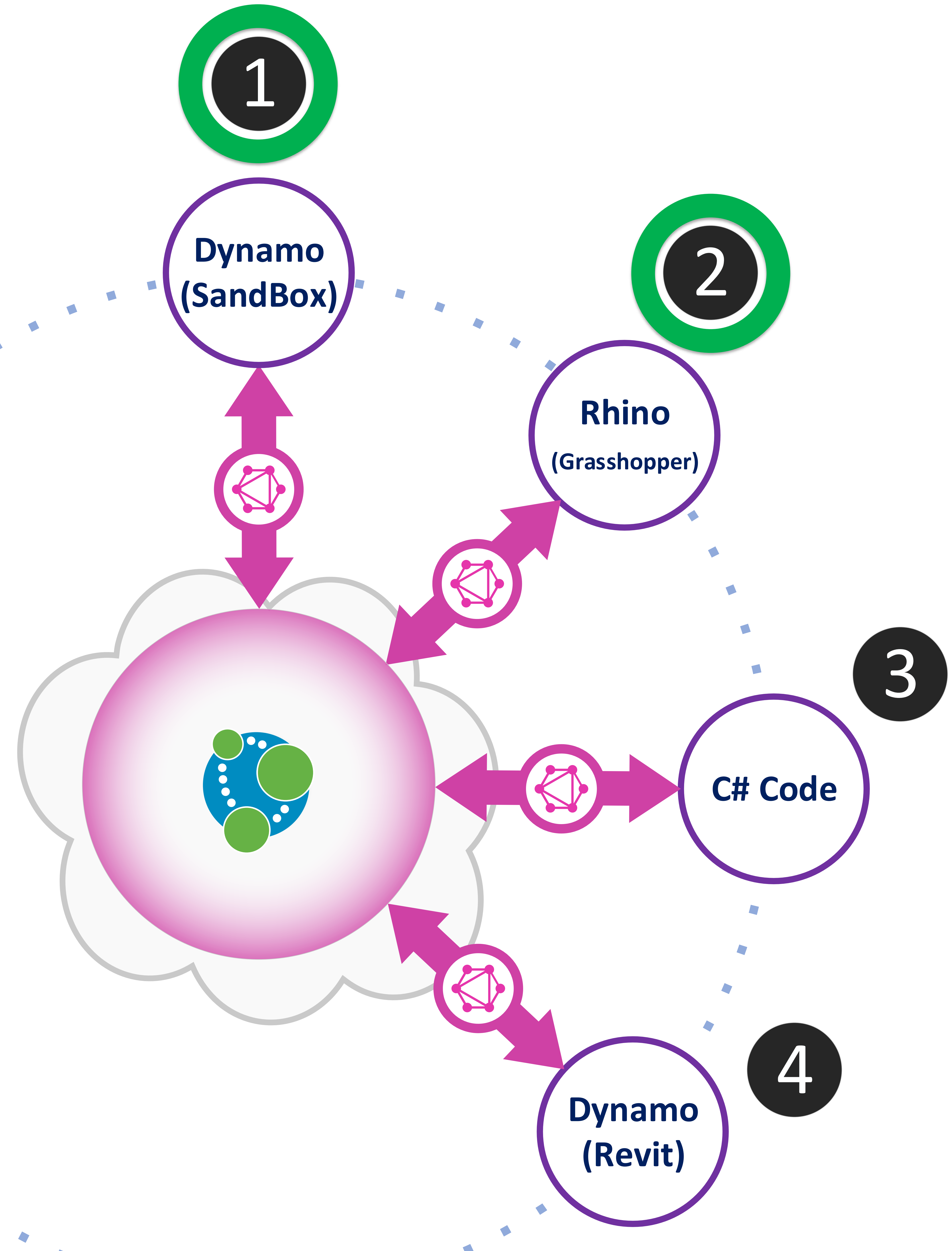
THEN

Gett star

A journey through the API

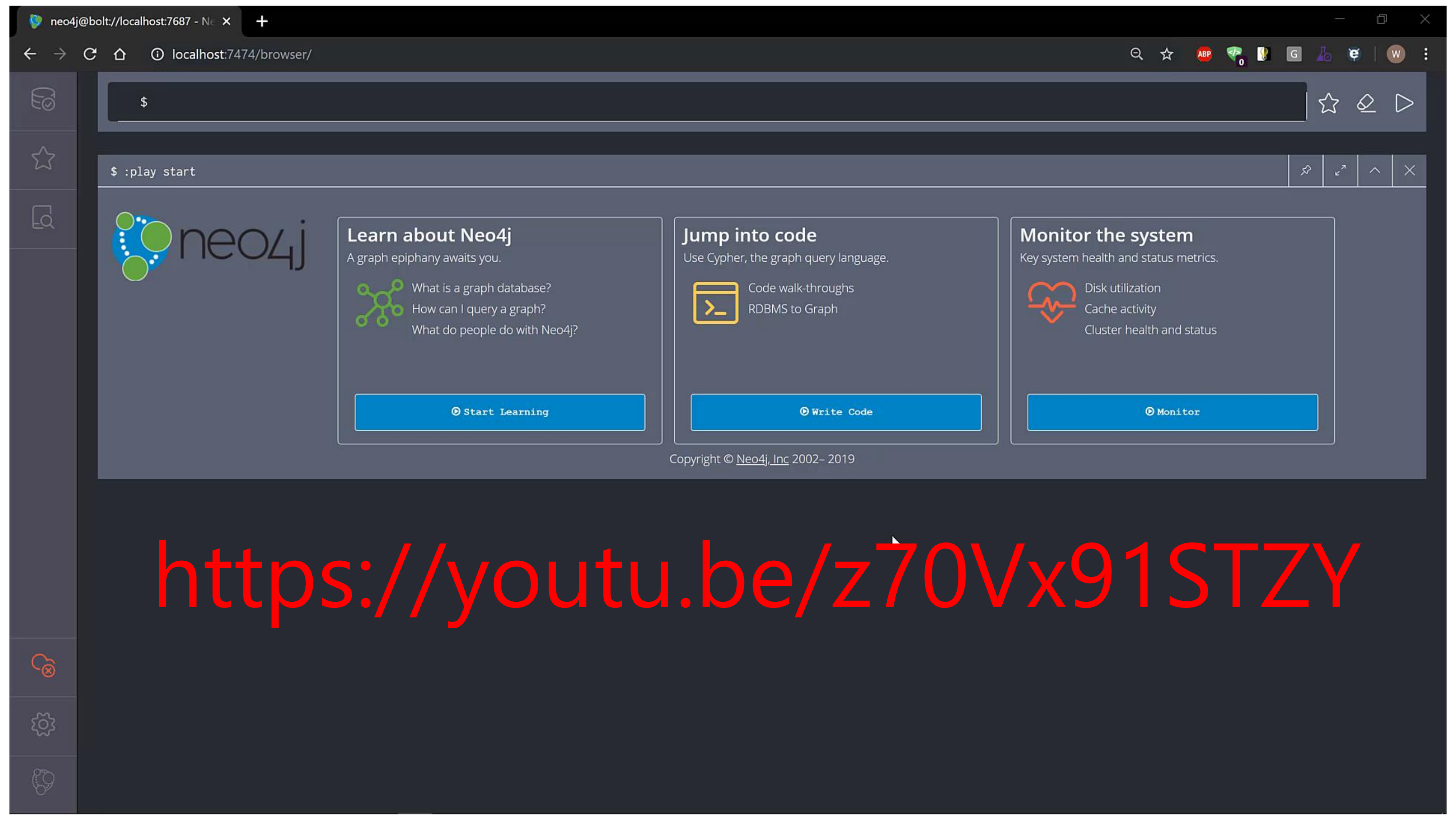


1. Create a project, building and levels
2. Create spaces and add to building, calculate volumes, area and other basic calcs
3. Add data to space for circuits, electrical outlets and DB Panels
4. Update spaces in Revit ~~and add terminals, sockets and panels~~



A journey through the API

1. Create a project, building and levels



```
$ :play start
```

Learn about Neo4j
A graph epiphany awaits you.
What is a graph database?
How can I query a graph?
What do people do with Neo4j?

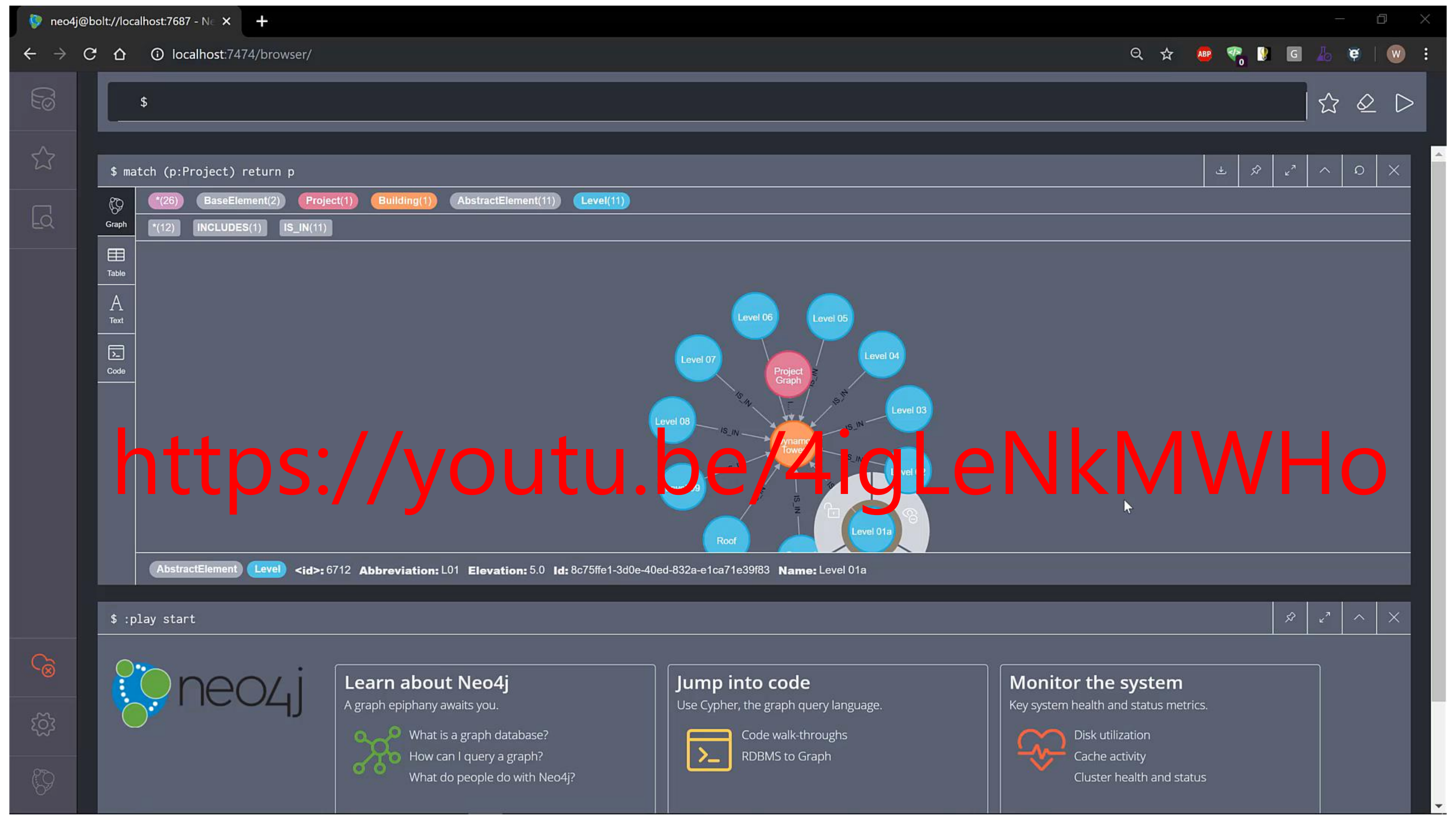
Jump into code
Use Cypher, the graph query language.
Code walk-throughs
RDBMS to Graph

Monitor the system
Key system health and status metrics.
Disk utilization
Cache activity
Cluster health and status

Copyright © Neo4j, Inc 2002– 2019

https://youtu.be/z70Vx91STZY

2. Create spaces and add to building, calculate volumes, area and other basic calcs



```
$ match (p:Project) return p
```

AbstractElement Level <id>: 6712 Abbreviation: L01 Elevation: 5.0 Id: 8c75ffe1-3d0e-40ed-832a-e1ca71e39f83 Name: Level 01a

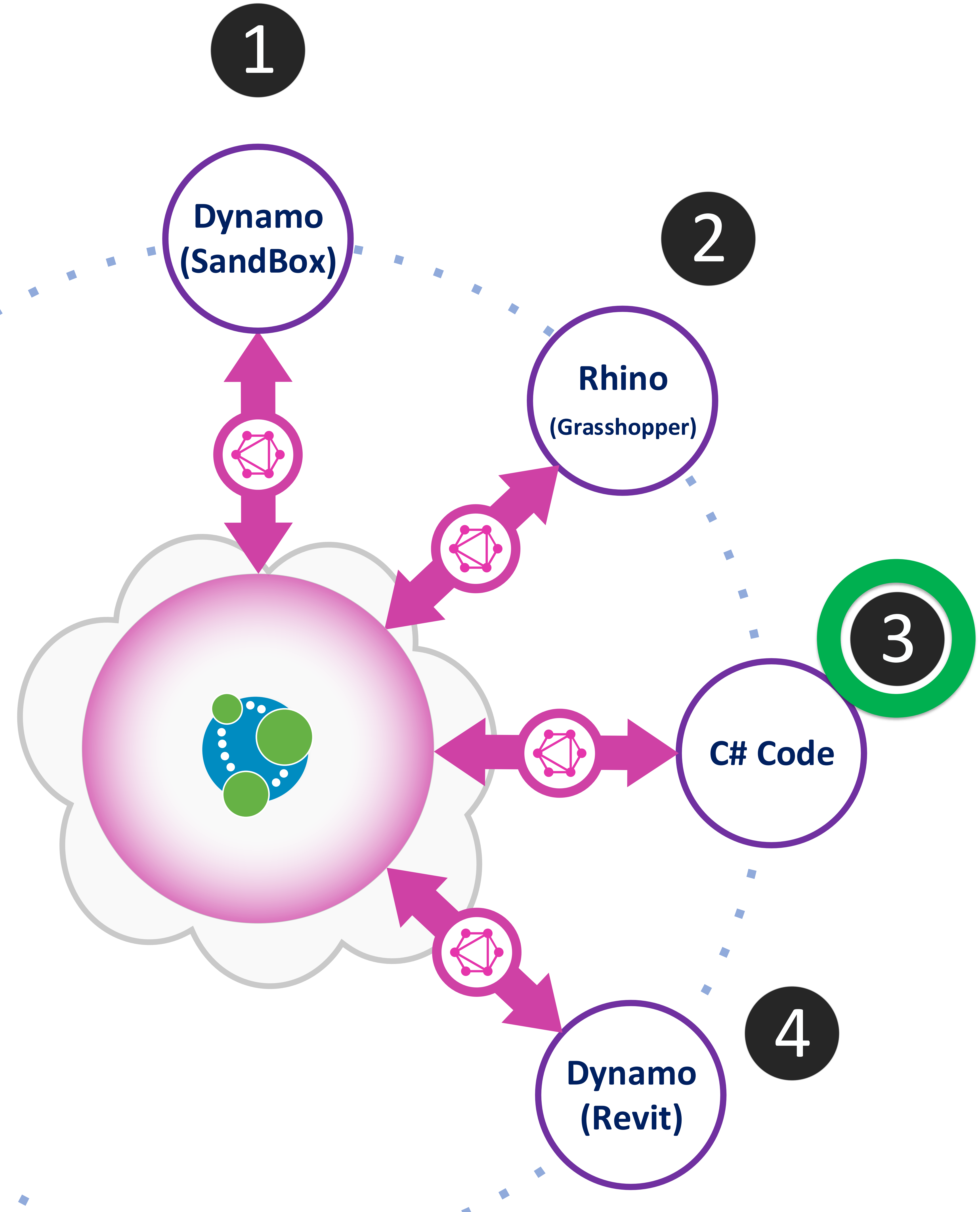
```
$ :play start
```

Learn about Neo4j
A graph epiphany awaits you.
What is a graph database?
How can I query a graph?
What do people do with Neo4j?

Jump into code
Use Cypher, the graph query language.
Code walk-throughs
RDBMS to Graph

Monitor the system
Key system health and status metrics.
Disk utilization
Cache activity
Cluster health and status

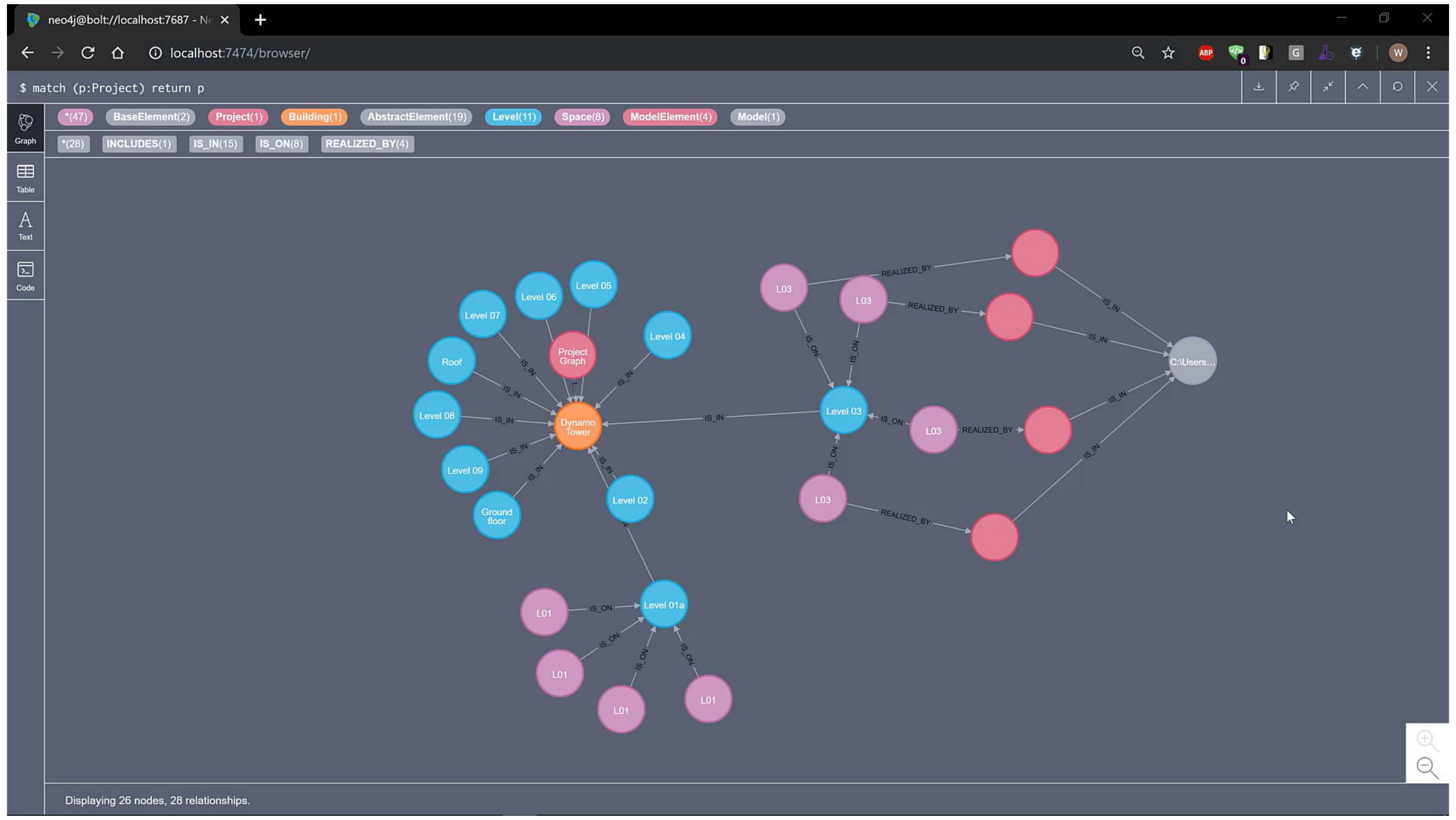
https://youtu.be/4igLeNkMWHo



A journey through the API

3. Add data to space for circuits, electrical outlets and DB Panels

https://youtu.be/PyqtlbQc6U4



neo4j@bolt://localhost:7687 - N

localhost:7474/browser/

```
$ match (p:Project) return p
```

*(47) BaseElement(2) Project(1) Building(1) AbstractElement(19) Level(11) Space(8) ModelElement(4) Model(1)

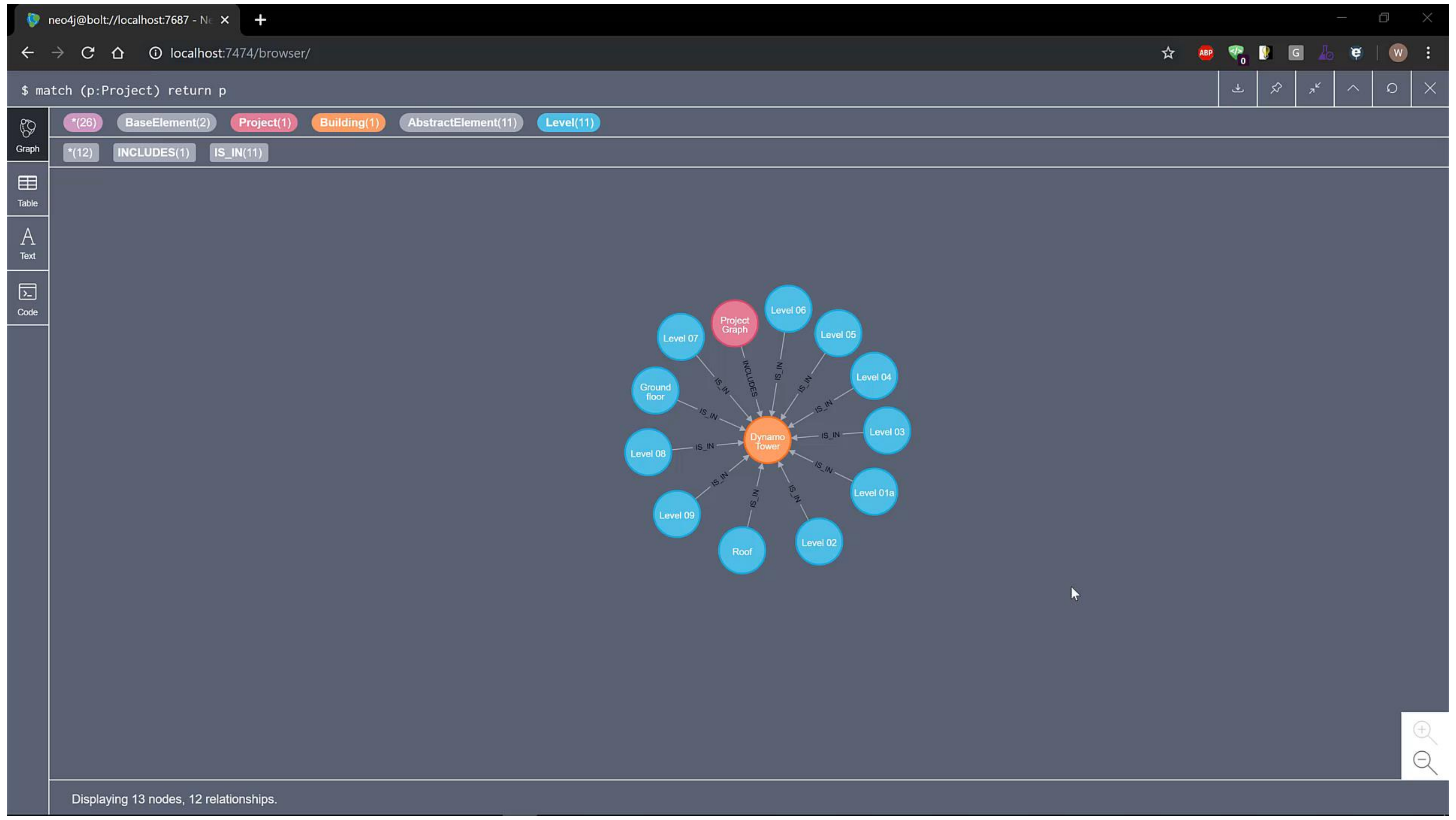
*(28) INCLUDES(1) IS_IN(15) IS_ON(8) REALIZED_BY(4)

Displaying 26 nodes, 28 relationships.

A journey through the API

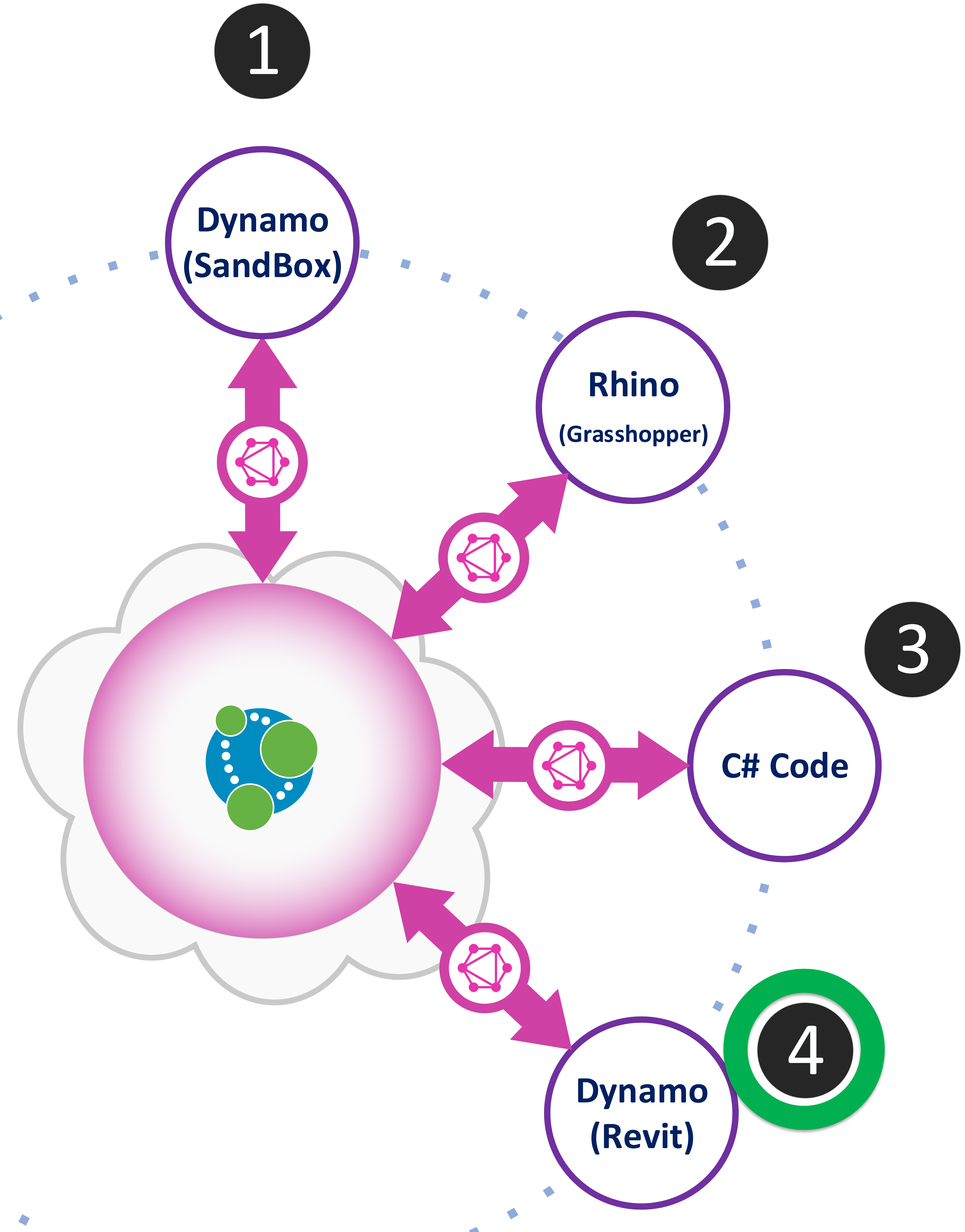
4. Update spaces in Revit

https://youtu.be/2bya1c1Djak



```
$ match (p:Project) return p
```

Displaying 13 nodes, 12 relationships.



Where might you use this??

- Coordinate space data between simulation software and Revit, and show it on treatment plan views in Revit
- Update level names or other info across all models
- Connect multiple buildings together for a more holistic design
- Model setup; select levels and/or other elements and bring them in to a new model, together with any required linked models.
- Surface up data to PowerBI for a more intuitive interface than schedules
- Many more…
- Paving the way for engineers skilled in Dynamo, Python and other languages.

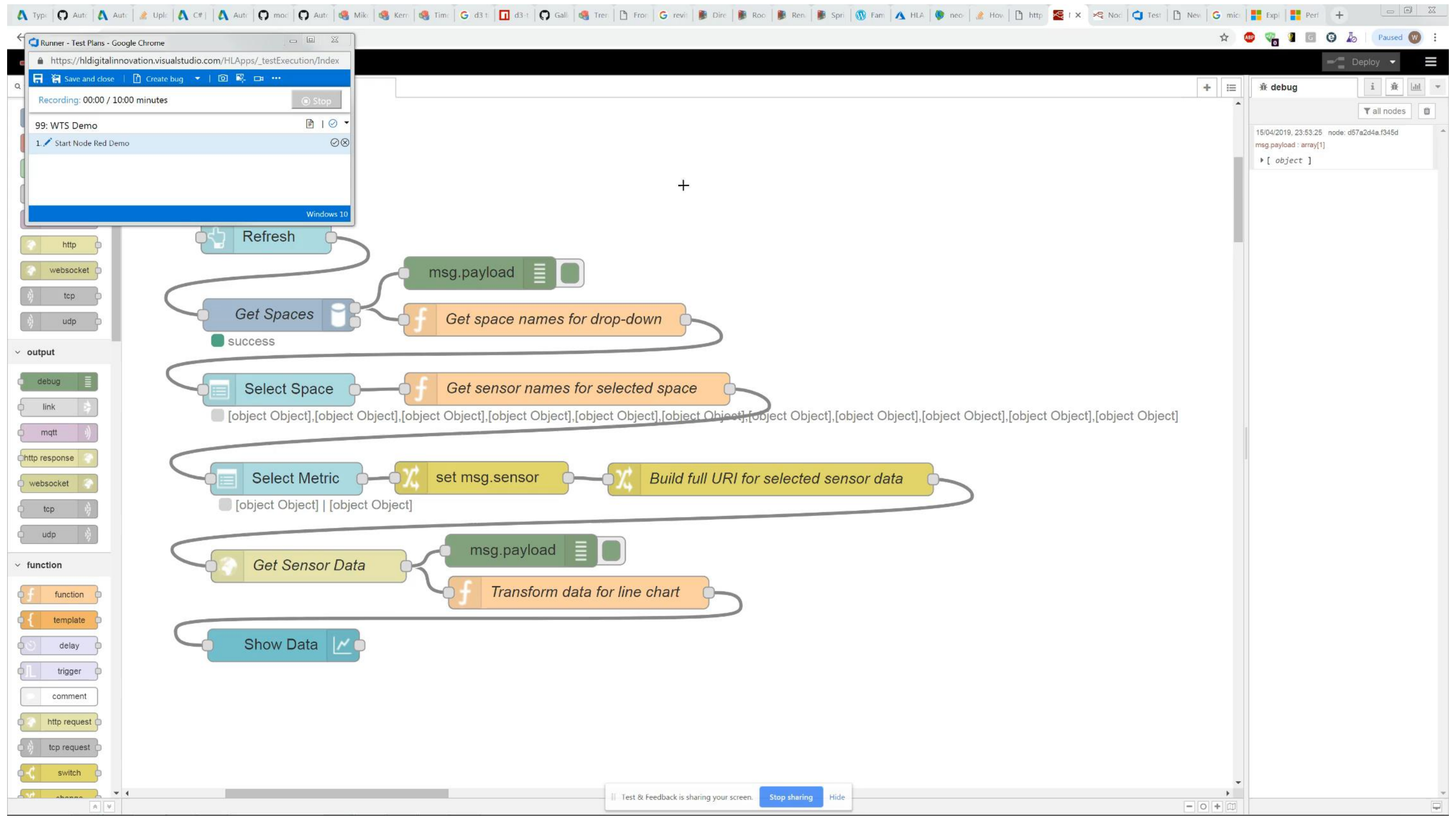
Other Building Graph example integrations

Operational data : Integration with Forge

https://youtu.be/hsbBHZTMWtM

Operational data : Integration with NodeRED

https://youtu.be/VXadWvZcSZ0



Runner - Test Plans - Google Chrome

https://hldigitalinnovation.visualstudio.com/HLApps/_testExecution/Index

Recording: 00:00 / 10:00 minutes

99: WTS Demo

1. Start Node Red Demo

```
15/04/2019, 23:53:25  node: d57a2d4a.f345d
msg.payload : array[1]
▸[ object ]
```

Save and close
Create bug
Stop
Windows 10
Deploy
debug
all nodes
http
websocket
tcp
udp
output
debug
link
mqtt
http response
websocket
tcp
udp
function
function
template
delay
trigger
comment
http request
tcp request
switch
Refresh
msg.payload
Get Spaces
success
Get space names for drop-down
Select Space
Get sensor names for selected space
[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object]
Select Metric
set msg.sensor
Build full URI for selected sensor data
[object Object] | [object Object]
Get Sensor Data
msg.payload
Transform data for line chart
Show Data
Stop sharing
Hide

Getting started

THEN

THEN

A journey
hrough the
API

Getting Started

Knowledge required to create a Building graph server:

- GIT
- Docker, including Docker Compose
- Optional: Node.js and Express
- Optional: Azure/Amazon/Google/other cloud

Knowledge required to build integrations:

- HTTP POST/GET
- GraphQL and JSON
- Dynamo/Grasshopper/NodeRED
- Python/C#/JavaScript/Or any other language

To get started, use these commands in your favorite terminal:

```
$ git clone https://github.com/willhl/BuildingGraph-Server.git
$ cd BuildingGraph-Server
$ docker-compose up
```

- Downloads all the code and files required
- Compiles the building graph server
- Brings up a local building graph server container instance
- Brings up a local Neo4j Database container instance.
- Mounts essential directories outside of the container

What's in the repositories:

https://github.com/willhl/BuildingGraph-Client-Examples
https://github.com/willhl/BuildingGraph-Client-Revit
https://github.com/willhl/BuildingGraph-Server

Building Graph Server (JavaScript)
/BuildingGraph-Server

- GRANDstack implementation + schema files

Building Graph Client (C#)
/BuildingGraph-Client-Revit

- Handles GraphQL HTTP requests and introspection
- Automatic generating of mutation requests
- Client mapping framework to translates parameter names to schema parameter names

Dynamo ZeroTouch and Grasshopper nodes (C#)

- Wrappers around Building Graph Client

Revit Integration (C#)

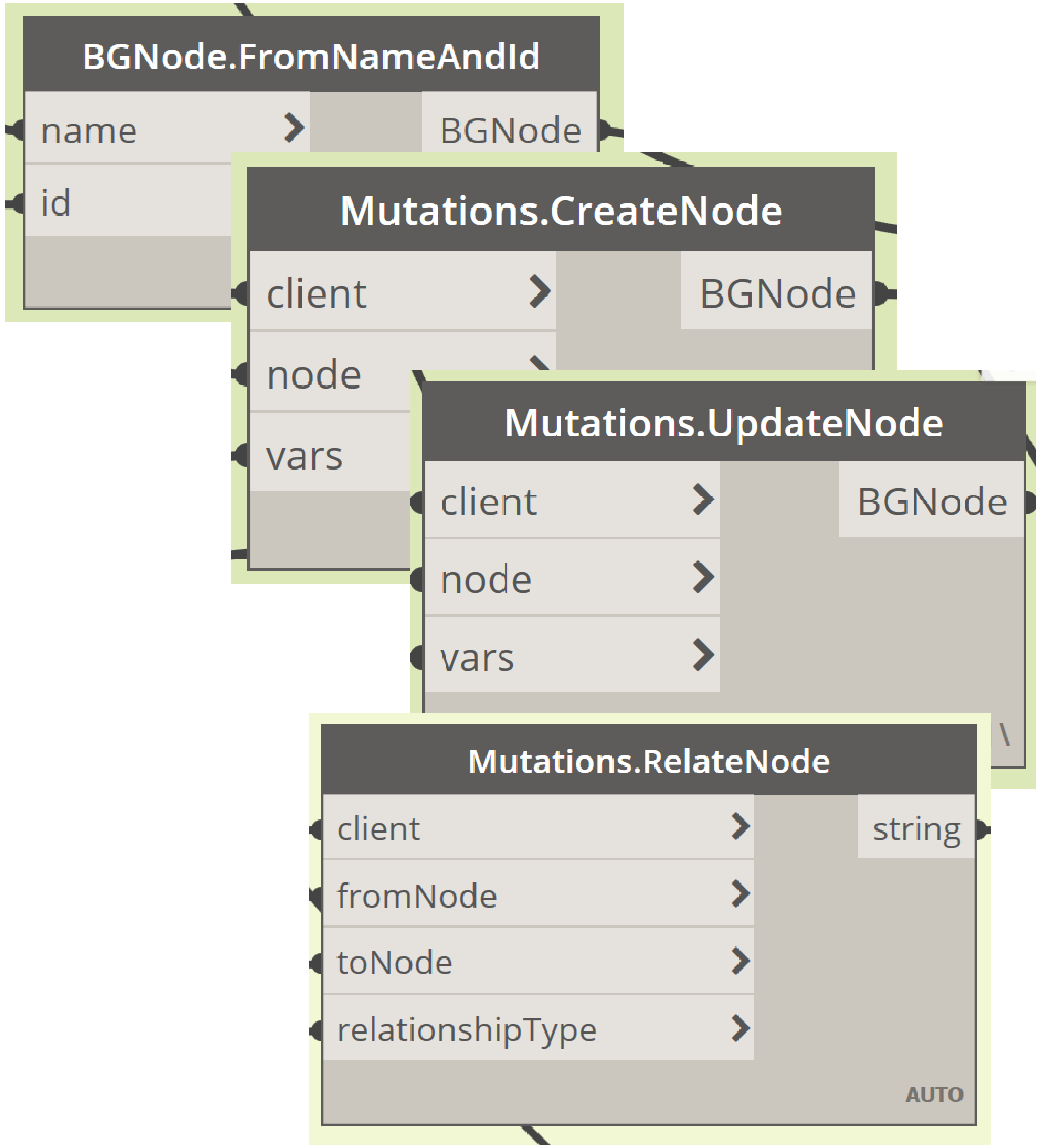
- Publishes full Revit models to Neo4j. BOLT only, GraphQL WIP
- Writes change requests to Revit model from Neo4j
- Unit and parameter translation from Revit to GraphQL.

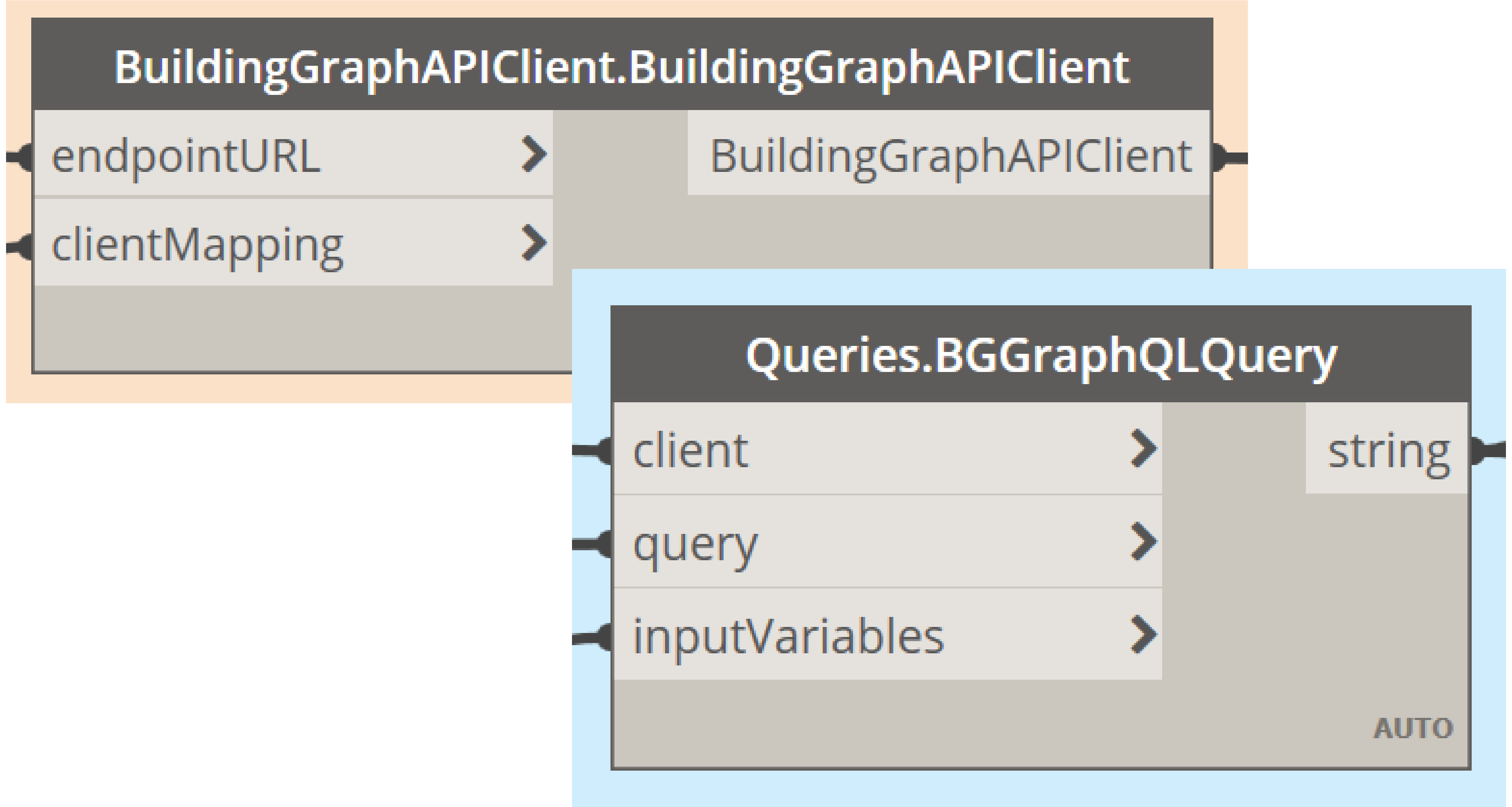
API Journey scripts:
/BuildingGraph-Client-Examples

Dynamo and Grasshopper:

Dynamo ZeroTouch and Grasshopper nodes (C#)

- Wrappers around Building Graph Client
- Add to Dynamo via Import
- Other packages required: JsonData





Caveats and Limitations

Multiple projects

- As of Neo4j 3.5: One docker container instance per project, routing via NGNIX or equivalent
- Coming soon Neo4j 4: Multiple databases in a single instance and cross database queries
- Just released: Neo4j Aura DBaaS https://neo4j.com/aura/

Authentication and Authorisation

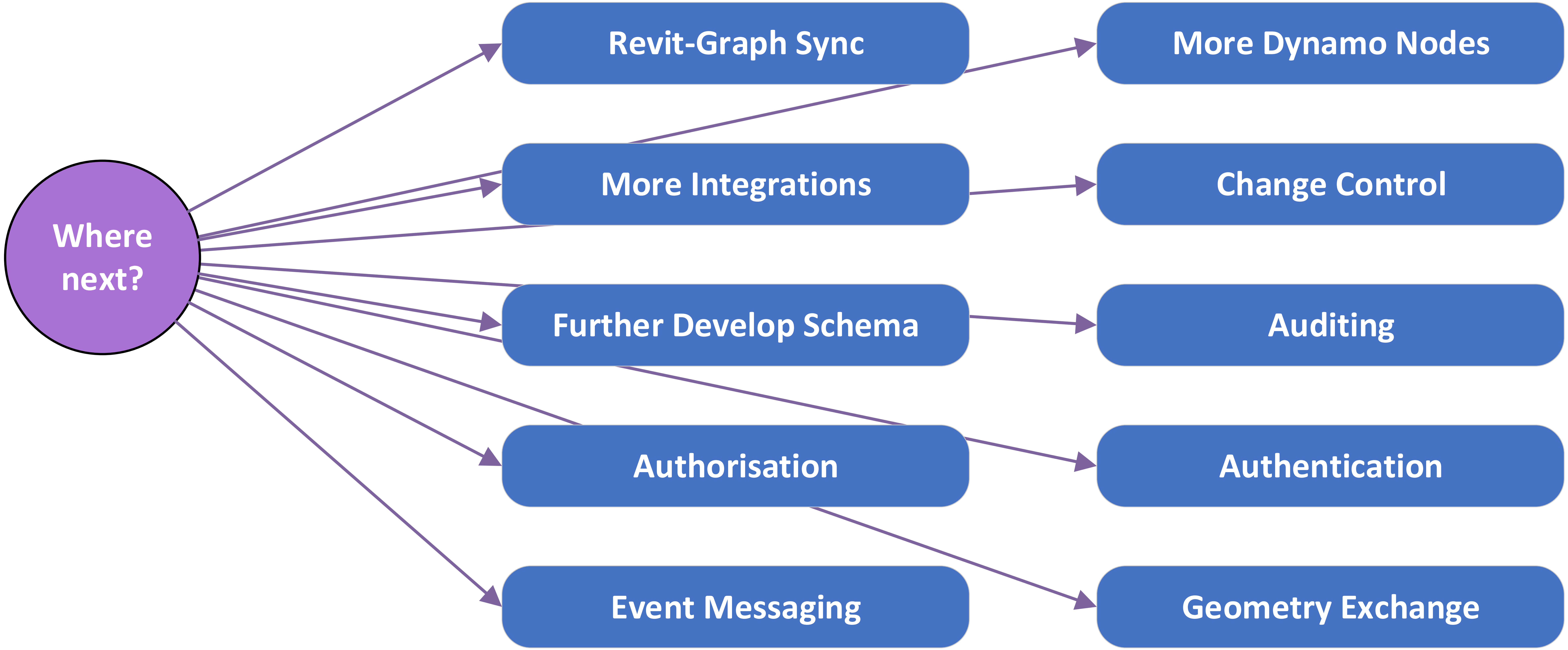
- Must use secured http endpoints when exposed to the internet, by proxy, VPN, etc.
- ToDo: Add to Passport.js middleware to Building Graph Server https://jkettmann.com/authentication-and-authorization-with-graphql-and-passport/
- GRANDstack can support fine grained trust levels. Defined by directives in the GraphQL schema https://grandstack.io/docs/neo4j-graphql-js-middleware-authorization.html

Neo4j Database Backup

- Community edition: Possible with offline backups (with data mounted outside of container)
- Enterprise edition: Possible with online backups

Where next

The Building Graph API is only just getting started, still lots to do:



Conclusion

This class was aimed at presenting the case, the final solution may be a little way off..
But I think the Building Graph API has huge potential:

- Feel free to build your own integrations
- Develop the schema for your own use
- Contributions welcome

Inevitably though, the Building Graph schema does need to be standardized for this to work across separate organizations… so where to go from here??

Hopefully, at least, this presentation has conveyed the virtues of using GraphQL and Neo4j.

✉ WillReynolds@HoareLea.com
https://twitter.com/d2liYmxl
https://www.instagram.com/d2liymxl/

https://github.com/willhl/BuildingGraph-Client-Examples
https://github.com/willhl/BuildingGraph-Client-Revit
https://github.com/willhl/BuildingGraph-Server

THEN

THEN

Questions

AUTODESK®

Make anything™