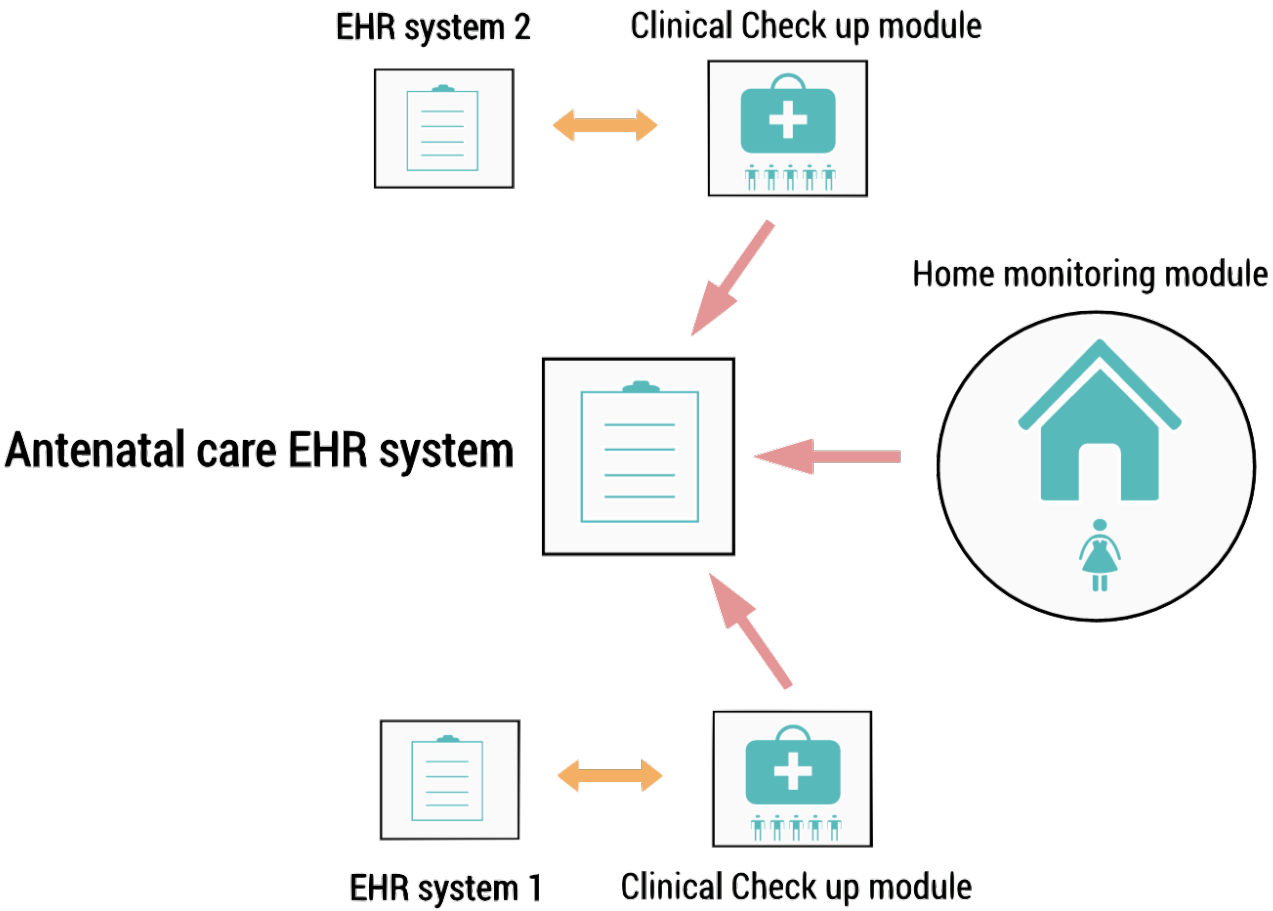
Implementation of an openEHR repository using a Graph Database

medical informatics

El Helou Samar

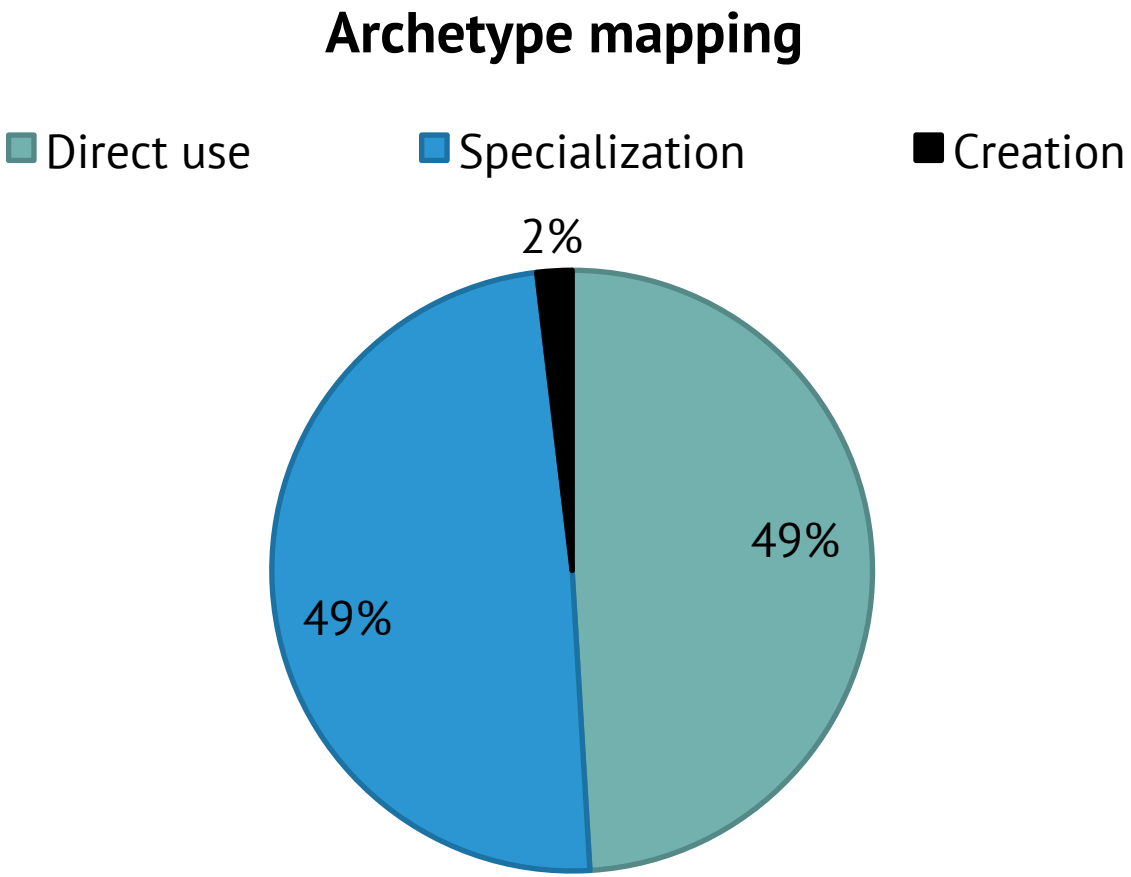
openEHR developer's workshop - HEC16
Munich

KYOTO UNIVERSITY

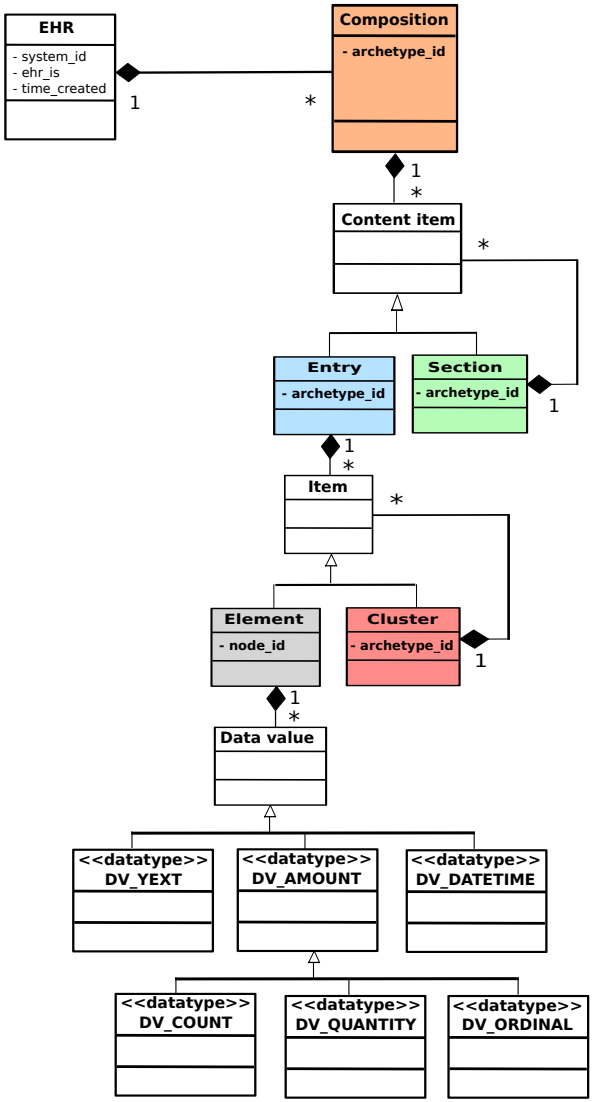

EHR system 2
Clinical Check up module
Home monitoring module
Antenatal care EHR system
EHR system 1
Clinical Check up module

Clinical Encounters Screenings	First Visit	week 8 -12	week 16-20-24-26-30-32-34	week 28	week 36	week 37-28-39-40	Additional Tests
Initial History and Physical	✔						
Family Medical History	✔						
Lifestyle	✔						
Screening for Depression	✔						
Pregnancy Test	✔						
Pelvic Exam	✔						
Pap Smear Test	✔						
Blood Tests*	✔			✔	✔		
Gonorrhea and Chlamydia cultures	✔						
Estimated Date of Birth	✔						
Height	✔						
Weight	✔	✔	✔	✔	✔	✔	
Edema Check	✔	✔	✔	✔	✔	✔	
Blood Pressure	✔	✔	✔	✔	✔	✔	
Urine Tests	✔	✔	✔	✔	✔	✔	
Ultrasound Scan	✔	✔	✔	✔	✔	✔	
Length of Fundus Uteri and abdominal circumference			✔	✔	✔	✔	
Fetal Heartbeat Check		✔	✔	✔	✔	✔	
NST- Non Stress Test					✔	✔	
Group B Streptococci					✔		
Tuberculosis							✔
Maternal Serum Screen							✔
Chorionic Villus sampling							✔
Cell Free Fetal DNA							✔
Nuchal translucency Ultrasound							✔
Amniocentesis							✔

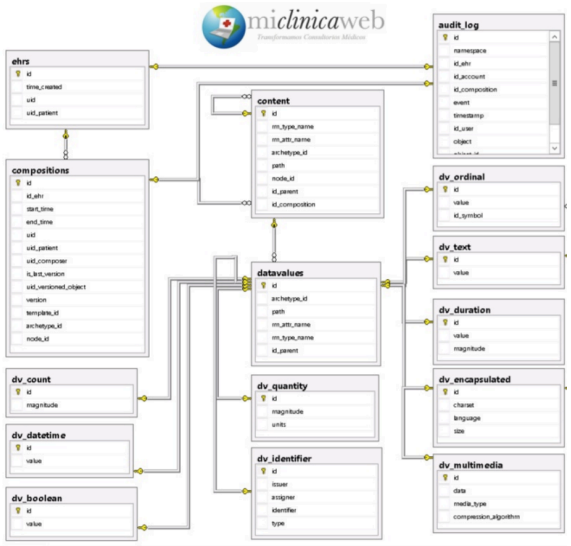
Archetypes allowed a 98% external reuse level for the antenatal care EHR



A graph database for the repository implementation



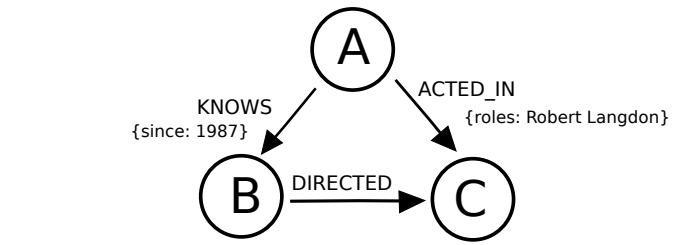
openEHR Reference Model structure



RDB implementation [1]

Schema hard to model

Queries can be complex



Labeled property graph model

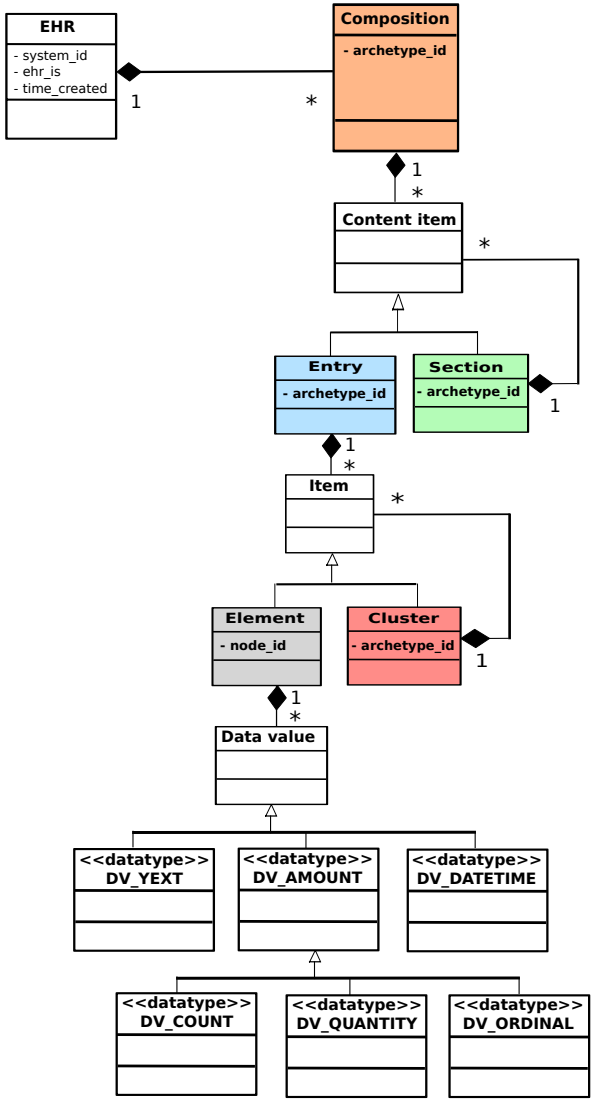
Simple model

Direct Mapping

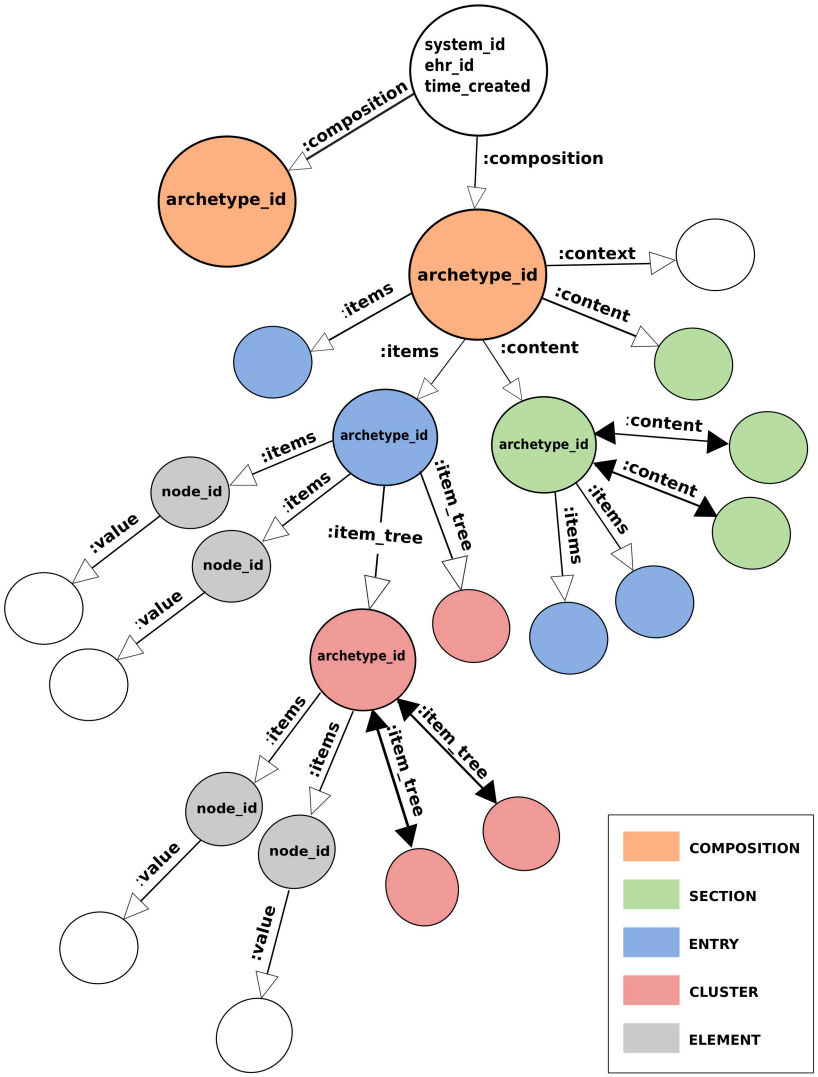
neo4j

[1] Tutorial - Design and Implementation of Clinical Databases with openEHR Pablo Pazos Gutiérrez, Koray Atalag, Luis Marco- Ruiz, Erik Sundvall, Sérgio Miranda Freire

Direct mapping



openEHR Reference Model structure



OSX 10.11.3

Neo4j Community edition version 3.0.1

Cypher query language

Rails version 4.2.5.1

Heroku + GrapheneDB

openEHR	Path	/ content[openEHR-EHR-SECTION.adhoc.v1] / items[openEHR-EHR-OBSERVATION.body_weight.v1] / data[at0002] / events[at0003] / state[at0008] / items[at0009]

```
CREATE (a:SECTION {archetype_id:"openEHR-EHR-SECTION.adhoc.v1"})-[:items]->
(b:OBSERVATION{archetype_id:"openEHR-EHR-OBSERVATION.body_weight.v1"})-[:data]->
(c:HISTORY{archetype_node_id:"at0002"})-[:events]->(d:POINT_EVENT{archetype_node_id:"at0003"})-[:state]->
(e:ITEM_TREE{archetype_node_id:"at0008"})-[:items]->(f:ELEMENT{archetype_node_id:"at0009"})
```

Graph

Semantic correspondence between the **path**, the **query** and the resultant **graph**

Demo application

Symptom/Sign name

Morning Sickness

Severity rating

6

Episodicity

Ongoing

Impact on life

Slow research progress

Severity Category

Mild

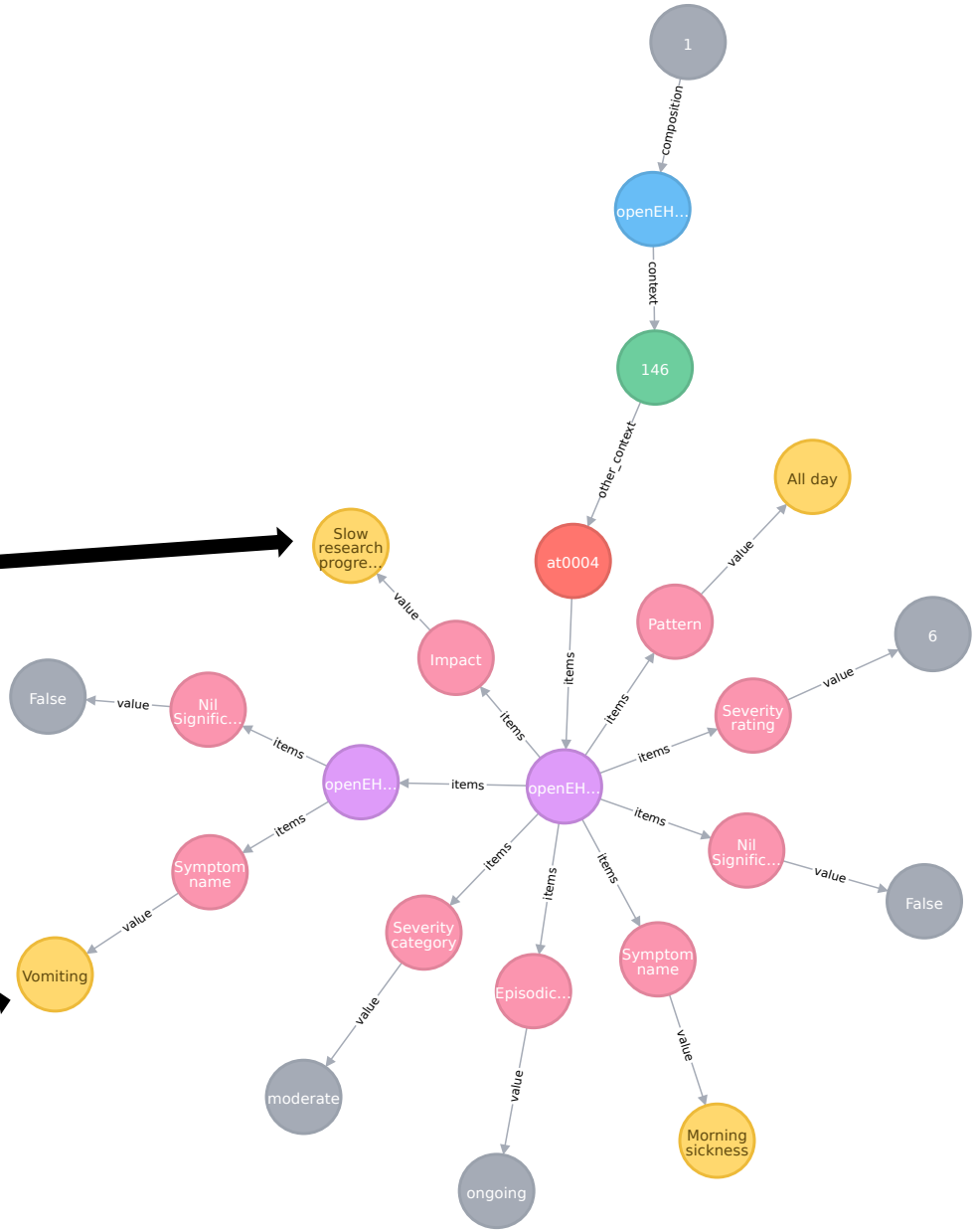
Pattern

Every morning

Associated symptom

Vomiting

Report



Using a graph DB resulted in less LLOC and less complexity for querying archetypes

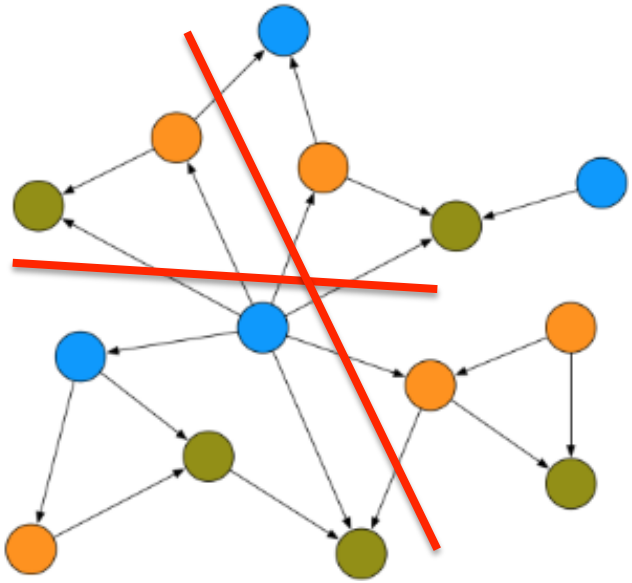
Criteria	RDB	GDB
Actions	• Design the schema • Create the tables • Input data by adding values into the tables	• Input data by creating nodes and relationships
Logical steps	• 5 Create table statements • 9 insert value statements	• 6 Create branches statements
LLOC for first time storing	14	6
Complexity of first time storing	19	2.7
LLOC for consequent storing	9	6
Complexity of consequent storing	12.2	2.7
LLOC for retrieving node lead value	3	2
Complexity of retrieving node leaf value	5.37	4.03

SQLite version 3.8.10.2, SQL queries **Neo4j Community edition version 3.0.1, Cypher queries**

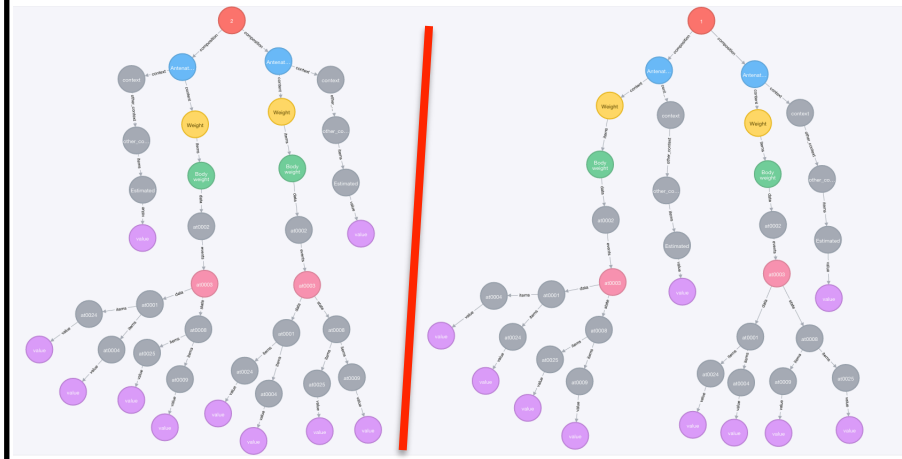
Neo4j limitation: 34 billion nodes ⟹ Need for partitioning

Problem: how to partition a graph in a way that minimizes the need to go back and forth between the partitions?

Big problem



Not a big problem in this case



Evaluate the performance of Neo4j for large openEHR data sets

Implement the demographic model

AQL ----> Cypher

Connecting the EHR graph to a SNOMED CT graph ?