A journey in developing an open-source application on Neo4j

Graph Summit Copenhagen, 7th March 2024

Mikkel Traun, Principal System Developer, Novo Nordisk A/S
Henrik Enquist, PhD, Lead Software Developer, Novo Nordisk A/S

What is the OpenStudyBuilder?...

A NEW APPROACH TO STUDY SPECIFICATION

- Compliance with external and internal standards
- Facilitates automation and content reuse
- Ensures a higher degree of end-to-end consistency

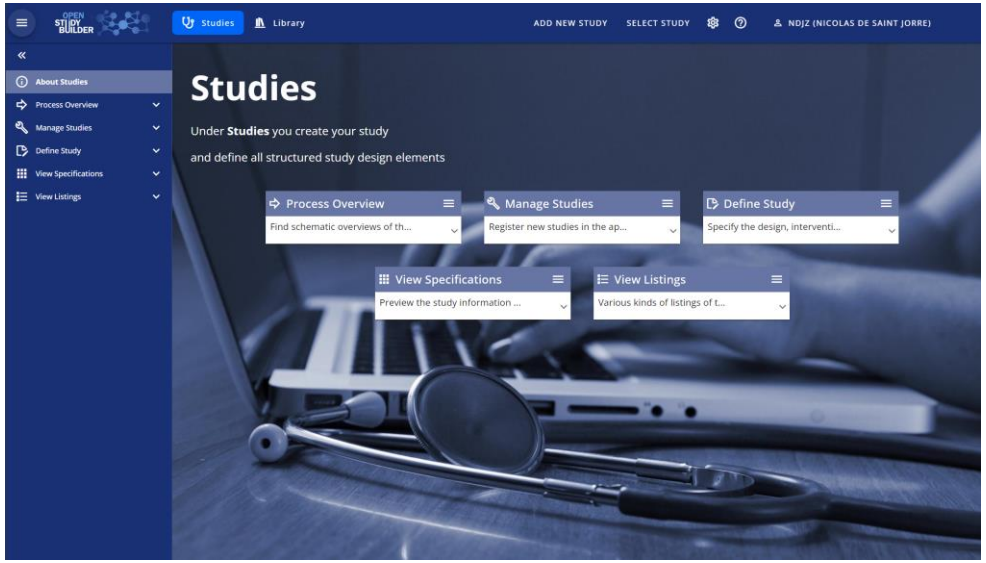
3 ELEMENTS OF OpenStudyBuilder

- **Clinical Metadata Repository (clinical MDR)**
  (central repository for all study specification data)
- **OpenStudyBuilder application / Web UI**
- **API layer**
  (allowing interoperability with other applications)
  (DDF API Adaptor – enabling DDF SDR Compatibility)

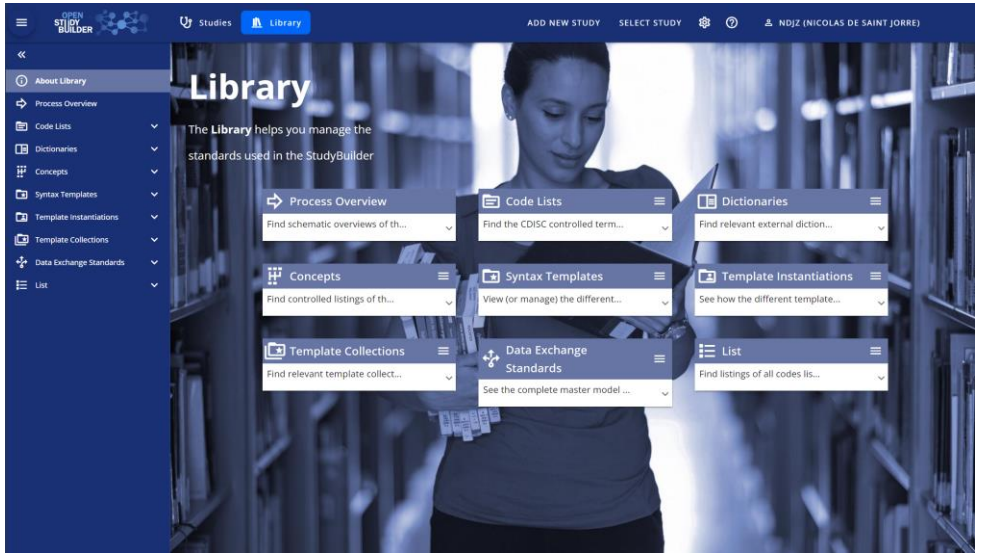
clinical **MDR**

OpenStudyBuilder Components

STUDIES	
TITLE	CRITERIA
REGISTRY IDENTIFERS	INTERVENTIONS
STRUCTURE	PURPOSE
POPULATION	ACTVITIES

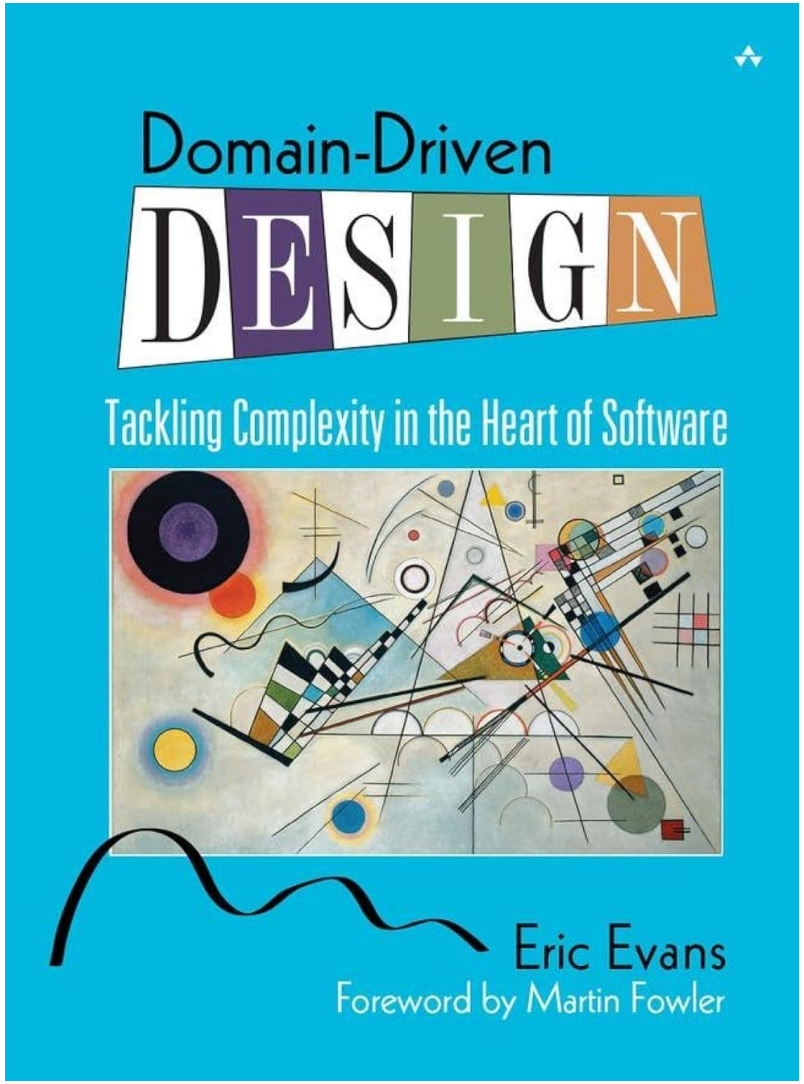


LIBRARY	
CONTROLLED TERMINOLOGY	MEDICAL DICTIONARIES (e.g., MedDRA)
CONCEPTS (ACTIVITIES, UNITS, CRFs, COMPOUNDS)	SYNTAX TEMPLATES
DATA EXCHANGE STANDARDS	



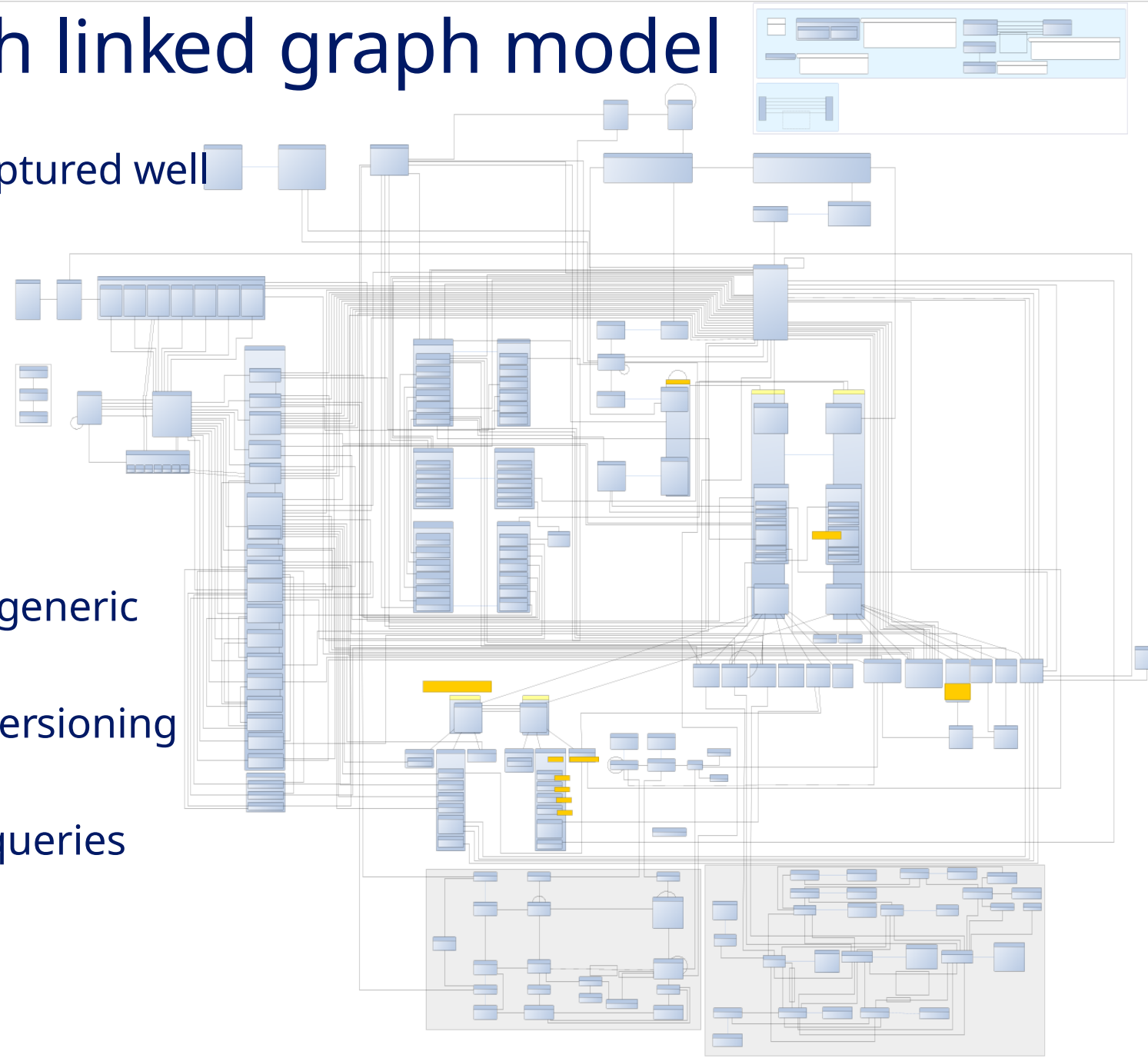
Applying Domain Driven Design principles

- Domain Driven Design (DDD) is a common design pattern in system development
  - Works well with the linked graph database design
  - Ending up with a very big data model – as the domain is complex – maybe OK
  - Can be complicated in the Python based API component
  - Challenges in using Neo4j OGM NeoModel with DDD principles in Python



Benefits working with linked graph model

- Big and complex data domain is captured well in the label property graph model
  - Given you understand the clinical data domain!
  - Support fine granular versioning
  - Support domain driven queries
  - Can deliver fast performance
- Other MDR solutions have applied generic relational data models
  - Having difficulties in managing versioning at a granular level
  - Very complex and long running queries



Technical details| MDR and StudyBuilder

Front-end

- Vue.js
  - Modern JavaScript web framework
  - Vuetify user interface styling library
  - Websites can be displayed on all operating systems
  - Views automatically adjust to underlying data
  - Pages are created with re-usable components, e.g. a table or a visualisation. Code once, reuse many times.
  - Wide usage:
    - Popular among developers
    - Google, Apple, Netflix
  - VuePress Documentation portal

Service layer

- Python driven API
  - FastAPI Framework
  - Highly readable code
  - More than 40% of developers use Python, according to Stack Overflow
  - Restful API
  - CDISC use python for their API as well
  - Automatically generate API documentation with inline code comments
  - Lightweight
  - Cloud hosting (Azure) provides elastic scaling – upscale and downscale according to immediate usage

Backend

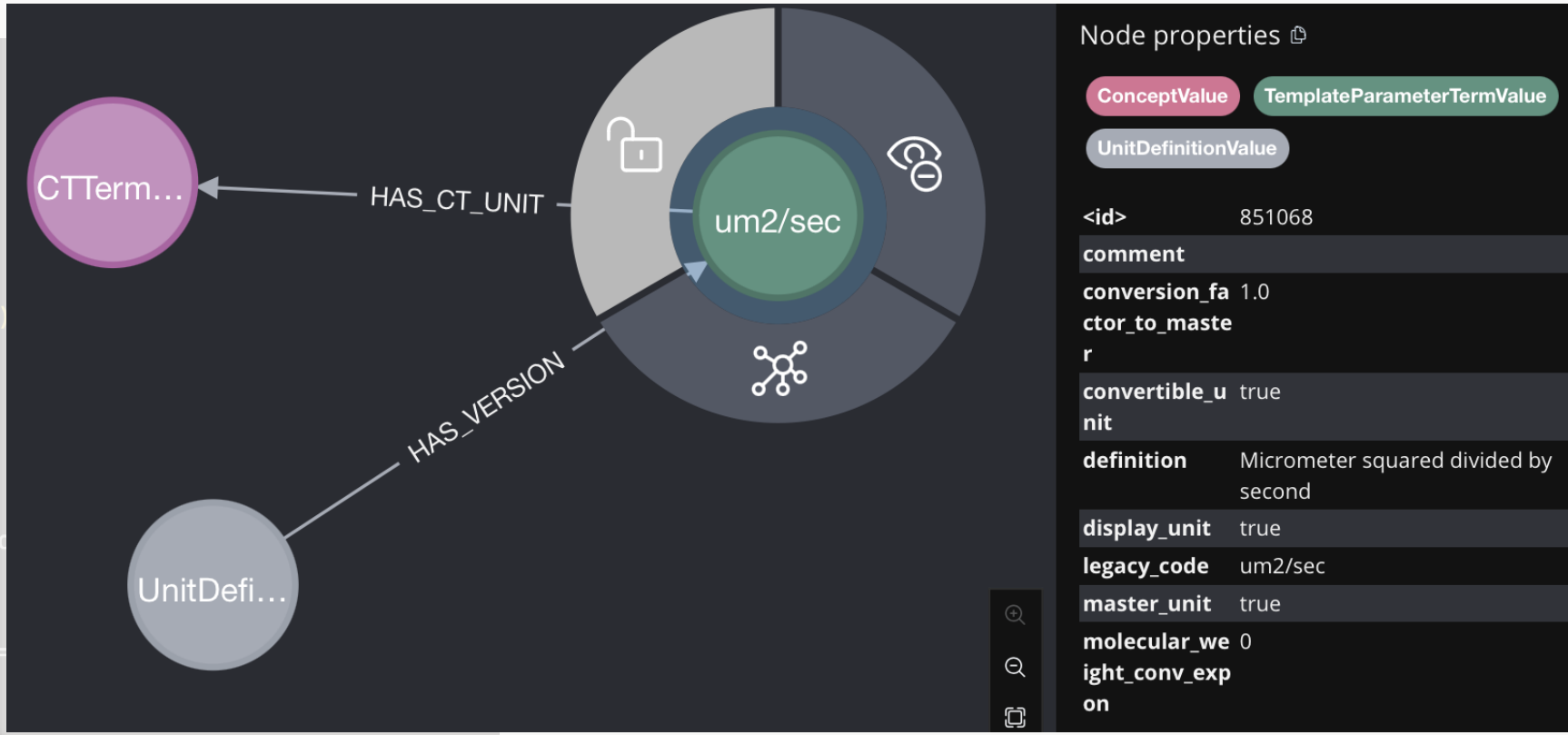
- Neo4j database
  - Modern label-property graph database
  - Data model is close to the domain
  - Hosted on any cloud or locally

Vue.js: https://vuejs.org/ *Vuetify*: https://vuetifyjs.com/en/ *VuePress*: https://vuepress.vuejs.org/ *FastAPI*: https://fastapi.tiangolo.com/ *Neo4j*: https://neo4j.com/neo4j-graph-database/

Challenges using neomodel versus plain Cypher - neomodel can be fast

```python
class UnitDefinitionValue(ConceptValue):
    legacy_code = StringProperty()
    convertible_unit = BooleanProperty()
    display_unit = BooleanProperty()
    master_unit = BooleanProperty()
    si_unit = BooleanProperty()
    us_conventional_unit = BooleanProperty()
    molecular_weight_conv_expon = IntegerProperty()
    conversion_factor_to_master = FloatProperty()
    order = IntegerProperty()
    comment = StringProperty()

    has_ct_unit = RelationshipTo(
        CTTermRoot, "HAS_CT_UNIT", cardinality=Zero
    )
    has_unit_subset = RelationshipTo(
        CTTermRoot, "HAS_UNIT_SUBSET", cardinality=
    )
    has_ct_dimension = RelationshipTo(
        CTTermRoot, "HAS_CT_DIMENSION", cardinality=ZeroOrOne, model=ClinicalMdrRel
    )
    has_ucum_term = RelationshipTo(
        UCUMTermRoot, "HAS_UCUM_TERM", cardinality=ZeroOrOne, model=ClinicalMdrRel
    )
```



https://github.com/neo4j-contrib/neomodel

Challenges using neomodel versus plain Cypher - neomodel can be slow

- Fetching many items with several properties → many queries, latencies add up
- A single “big” cypher query is much faster but is more work to write and maintain
- neomodel can do better

```
items = root_class.nodes.order_by(”-name")
for item in items:
    obj_a = item.has_obj_a.get_or_none()
    obj_b = item.has_obj_b.get_or_none()
    obj_c = item.has_obj_c.get_or_none()
    obj_d = item.has_obj_d.get_or_none()
```

```
items = root_class.nodes.all().fetch_relationships("obj_a", "obj_b", ...)
```

Challenges using neomodel versus plain Cypher

Make simple implementation with neomodel

If performance is satisfactory, done

If not, refactor to utilize neomodel better

- Or, if not possible, reimplement with Cypher

Keep track of performance as data amount increases

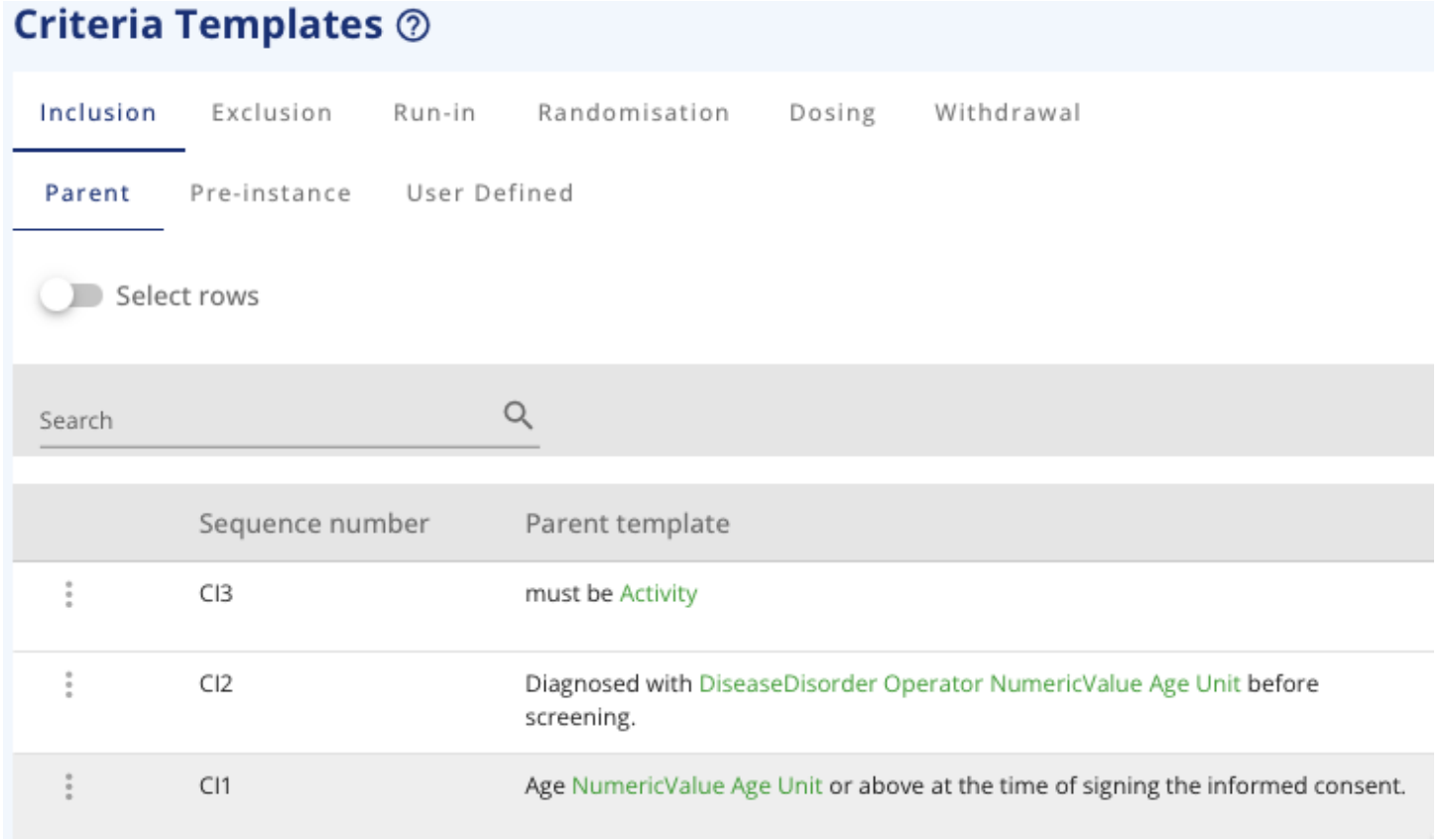
Challenges using neomodel versus plain Cypher

Dummy data:

- Simple
- Small
- Uses a subset of the data model

Production data:

- Complicated
- Big
- Uses (nearly) the full data model



- Need better dummy data!
- Generative AI can help to create richer dummy data

Challenges showing graph data in tables

- Vuetify Data table

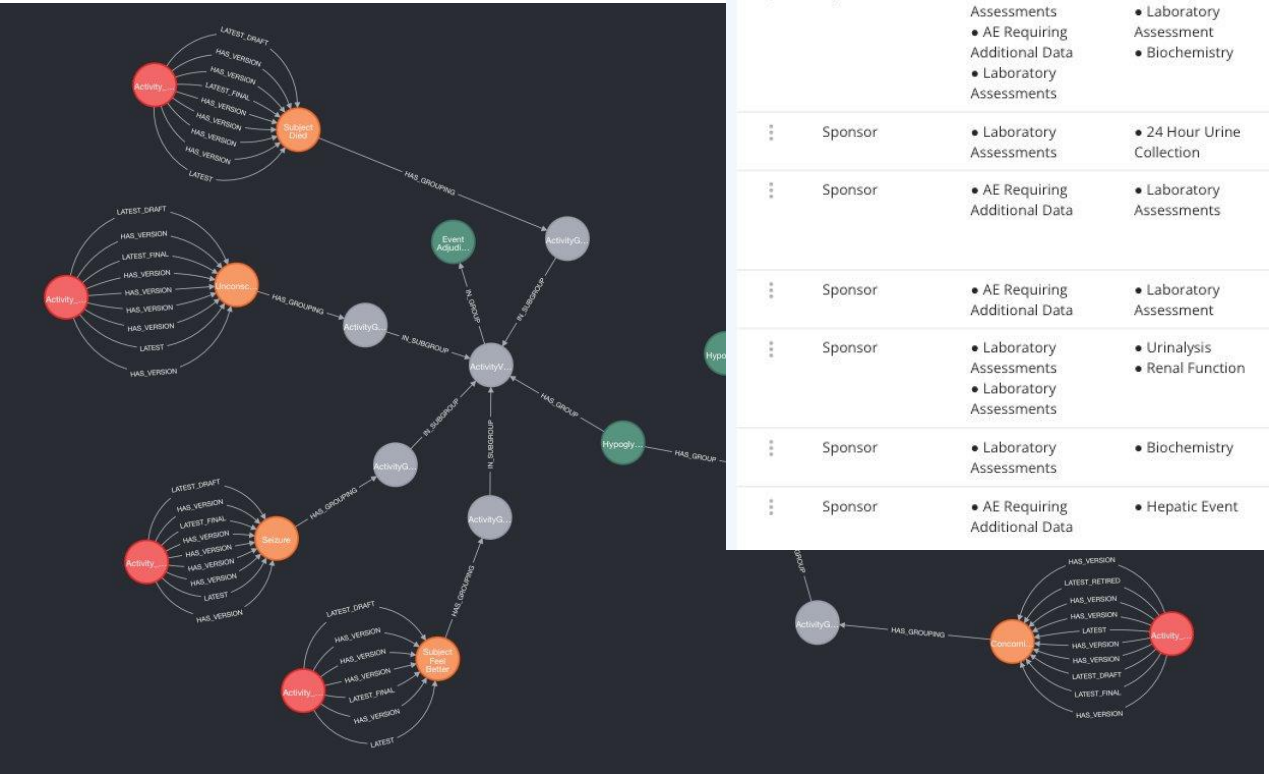
```
const desserts = [
  {
    name: 'Frozen Yogurt',
    calories: 159,
    fat: 6.0,
    carbs: 24,
    protein: 4.0,
    iron: '1',
  },
  {
    name: 'Jelly bean',
    calories: 375,
    fat: 0.0,
    carbs: 94,
    protein: 0.0,
    iron: '0',
  },
  {
    name: 'KitKat',
    calories: 518,
    fat: 26.0,
    carbs: 65,
    protein: 7,
    iron: '6',
  },
```

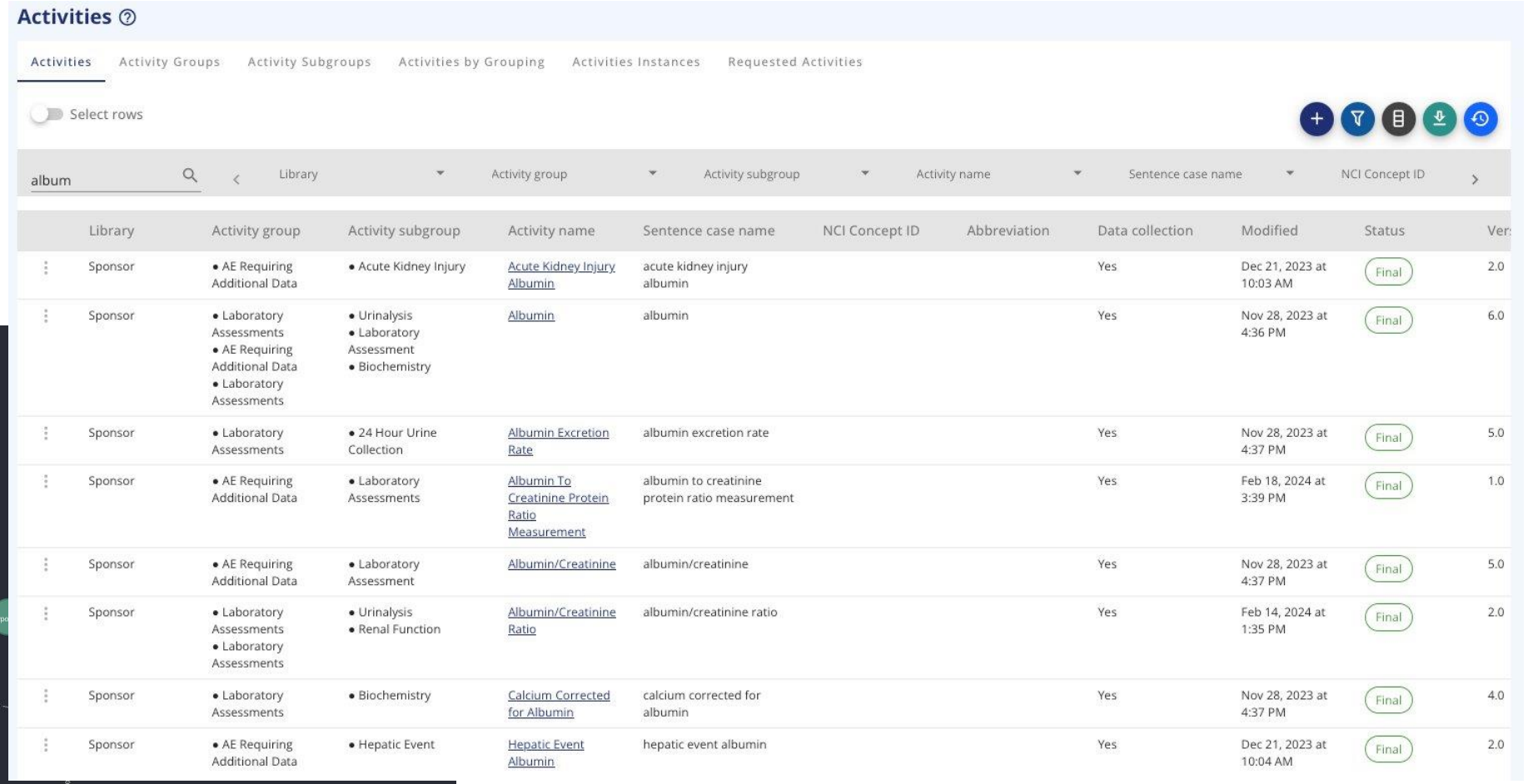
Dessert (100g serving)	Calories	Fat (g)	Carbs (g)	Protein (g)	Iron (%)
Frozen Yogurt	159	6	24	4	1
Jelly bean	375	0	94	0	0
KitKat	518	26	65	7	6
Eclair	262	16	23	6	7
Gingerbread	356	16	49	3.9	16

Items per page: 5 1-5 of 10

Challenges showing graph data in tables

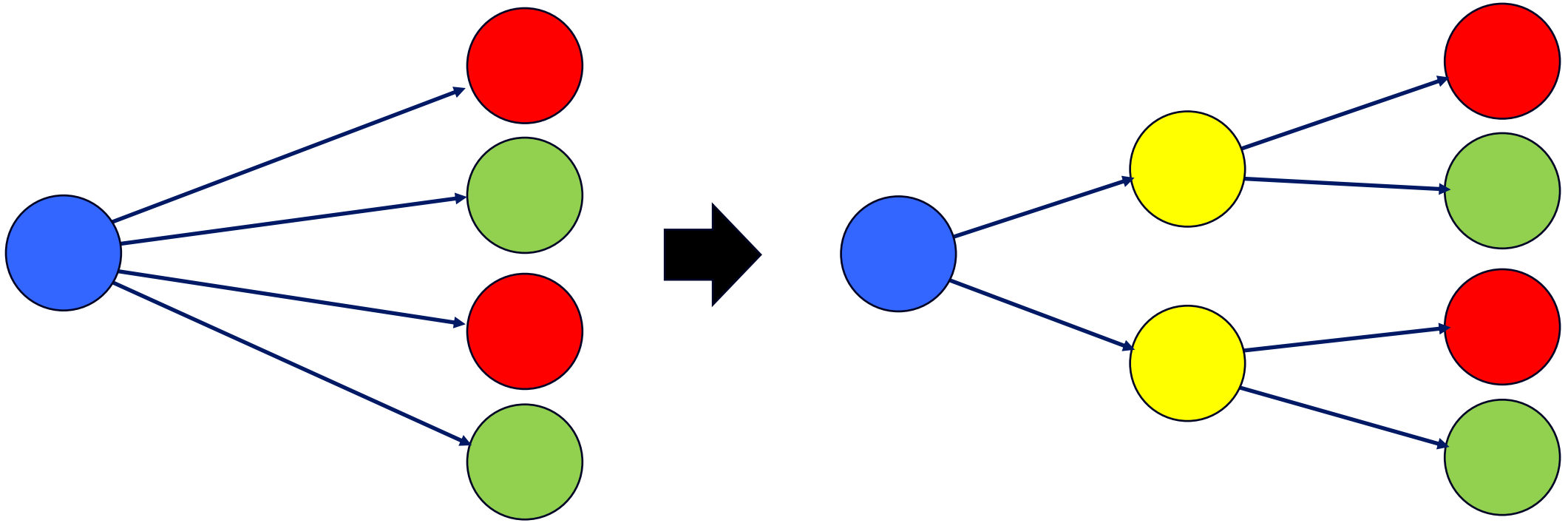
- StudyBuilder list of Activities





Changing data models and data migrations

- Model continuously evolves
- Example: Adding an intermediate node between two existing
- Migration needed!



Changing data models and data migrations

We need:

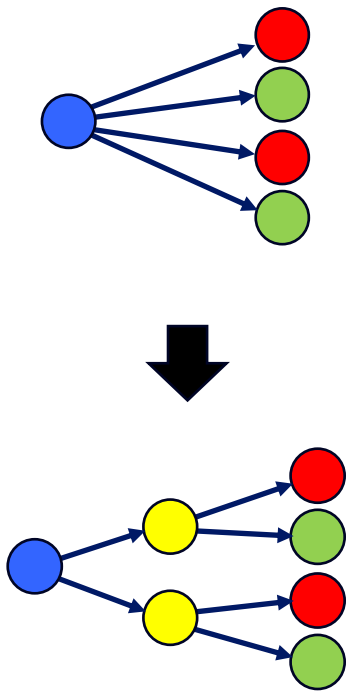
- Migration script (usually Cypher)
- Verification script (usually Cypher)
- Test data, preferably a sample extracted from production.

Test:

- Inject test data in a temporary DB
- Run migration
- Run verification
- Run migration again, assert that nothing changed

Challenge:

- Documenting the changes



Changing data models and data migrations - Change Data Capture

```python
@capture_changes()
def add_missing_end_dates(db_driver, log, run_label):
    """
    ## Add missing end date on HAS_VERSION relationships that are not the latest version.

    ### Change Description
    When a new version of an item is created the `HAS_VERSION`
    ....

    """

    desc = "Adding end dates for HAS_VERSION relationships that are not the latest"
    log.info(f"Run: {run_label}, {desc}")

    _, summary = run_cypher_query(db_driver,
        """
        MATCH (root)-[:HAS_VERSION]->()

                ....

                WHERE v.end_date IS NULL
                SET v.end_date = vp.start_date
            }
        """
    )
    counters = summary.counters
    return counters.contains_updates
```

@capture_changes() decorator

1) Extract docstring from wrapped function
2) Enable log enrichment
3) Query for the current change id
4) Run the wrapped function
5) Query for the changes
6) Dump the changes to a json file
7) Generate a summary and append to a markdown file

Changing data models and data migrations - Change Data Capture

Correction function: `add_missing_end_dates`

Function defined in `data_corrections/correction_005.py`

Description

> ### Add missing end date on HAS_VERSION relationships that are not the latest version.
>
> #### Change Description
>
> When a new version of an item is created the `HAS_VERSION` linking to the previous version must get an end date. There are a few old items where this has not worked. This correction fixes this by setting the missing end date to the start date of the following version.
>
> #### Nodes and relationships affected
>
> - Non-latest `HAS_VERSION` between `nnnRoot` and `nnnValue`, with missing `end_date` property.
> - Expected changes: 1 relationship property added

Recorded changes

- relationships:
  - updated:
    - count:
      - HAS_VERSION: 1
    - properties_added:
      - end_date: 1

Change details: add_missing_end_dates.test_correction.json

Neo4j Enterprise and open-source sharing

- Enterprise features are very useful.
- Who are the users of your open-source application?
  - Mainly other pharma's – they need full support so not an issue
  - Non-profit organizations – they can get a free license so not an issue
  - Smaller biotech's and CRO's – they can have an issue with costs!
- Can the functionality using Enterprise features be optional?
  - Will require a dedicated deployment, data level access control, will impact system validation and potential performance

Table 1. Community Edition vs Enterprise Edition key features

Feature	Community Edition	Enterprise Edition
Open source under GPLv3	✔	
Native Graph		
Bolt Protocol	✔	✔
Language drivers for C#, Go, Java, JavaScript & Python [1]	✔	✔
High-performance native API	✔	✔
APOC 450+ Core Procedures and Functions	✔	✔
Support for Neo4j Graph Data Science Community Edition [1]	✔	✔
Support for Neo4j Graph Data Science Enterprise Edition [1]		✔
Indexes and constraints		
Fast writes via native label indexes	✔	✔
Composite indexes	✔	✔
Full-text node & relationship indexes	✔	✔
Vector indexes Introduced in Neo4j 5.13	✔	✔
Online backup and restore		✔
Multiple databases (beyond the `system` and default databases)		✔
Autonomous clustering		✔
Composite databases		✔
Monitoring and management		
Endpoints and metrics for monitoring via Prometheus		✔
Neo4j Operations Manager		✔

Summary

- Neo4j database as store for enterprise application
- Big and complex data domain is captured well in the label property graph model
- Graph data model allows data to grow without getting more complicated
- Some challenges in how to present data to users

Thanks!
Questions?