LinkML Tutorial
ISMB 2024

Sierra Moxon, Patrick Kalita, and Kevin Schaper

July 8, 2024 (11am - 3pm PT / 2pm - 6pm ET)

These slides: bit.ly/LinkML-2024

Getting Started Together!

Shared google drive for this workshop: bit.ly/LinkML-2024-Drive

These slides: bit.ly/LinkML-2024

Learning Objectives:

- Learn how to create a new LinkML Schema Project.
- Learn the basics of LinkML model development by understanding the components of a LinkML model.
- Understand how to validate schema syntax and data conformity to a LinkML Schema.
- Understand how to generate JSONSchema and Pydantic models from a LinkML Schema.
- Understand how to automatically deploy documentation.

Logistics

Please feel free to ask any questions you may have during the tutorial.

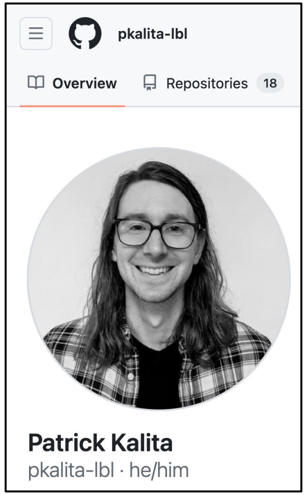
- Juno Q/A, chat
- google doc for questions and answers

Our Presenters:

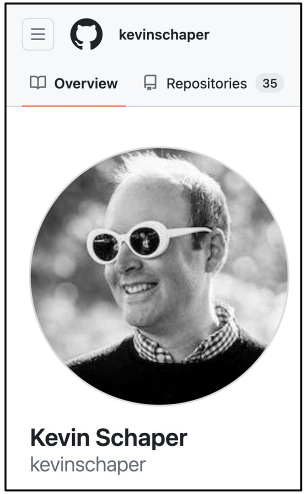
Sierra Moxon
Patrick Kalita
Kevin Schaper

Mark Miller will be monitoring the google doc for offline questions.

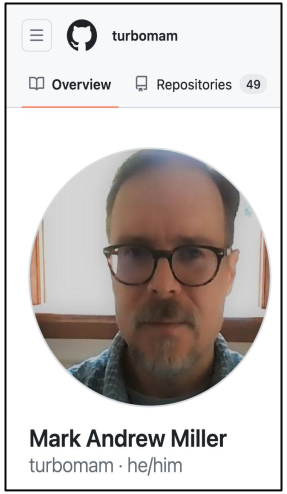
Sierra Moxon
sierra-moxon · she/her
orcid: 0000-0002-8719-7760



Patrick Kalita
pkalita-lbl · he/him



Kevin Schaper
kevinschaper



Mark Andrew Miller
turbomam · he/him

github.com/sierra-moxon github.com/pkalita-lbl github.com/kevinschaper github.com/turbomam

Thank you!!

Thank you to all of our open source contributors and to the NIH for funding so much of the work in this field!

NIH RM1HG010860

Code Of Conduct

LinkML Code of Conduct:

- Use welcoming and inclusive language
- Be respectful of differing viewpoints and experiences
- Gracefully accept constructive criticism
- Focus on what is best for the community
- Show empathy towards other community members

This Code of Conduct is adapted from the Contributor Covenant, version 1.4: https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

Tutorial schedule

Time (ET)	Topic	Presenter
14:00	**Introduction**	Sierra Moxon
14:20	**Section 0: Set up a LinkML repository**	Patrick Kalita
14:50	**Section 1: Authoring a LinkML model** A. Model components B. Classes and slots C. Hierarchies	Sierra Moxon
15:10	**BREAK**	
15:25	**Section 2: Authoring a LinkML model (cont.)** C. Mappings, enumerations	Sierra Moxon
15:40	**Section 3: Schema best practices, including linting**	Patrick Kalita
15:55	**Section 4: Generating code from your model** A. Generating Documentation B. Pydantic, JSONSchema	Kevin Schaper
16:35	**BREAK**	
16:45	**Section 5: Validating Data**	Patrick Kalita
17:35	**Wrap up/Questions**	Sierra Moxon

Software prerequisites for tutorial

- A GitHub account
- Python 3.9 and higher
- pipx

These slides: bit.ly/LinkML-2024

Rest stops along the way

- **We hope you follow along with us as we build up our project!**
- If you end up lost, each section of the tutorial has a corresponding tag in the linkml-tutorial-2024 repository (github.com/linkml/linkml-tutorial-2024)
- You can clone the linkml-tutorial-2024 repository and checkout the appropriate tag.

Step 0 - basic project creation

Step 1 - modeling

Step 2 - linting

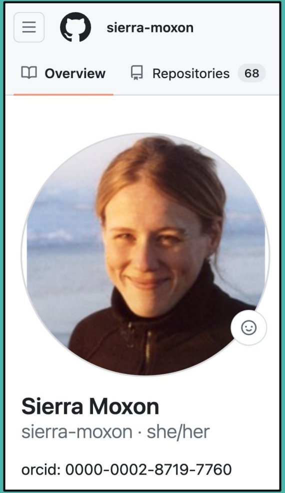
Step 3 - documentation

Step 4 - code generation

Step 5 - validation

```
> git checkout step_0_basic_project_creation
```

Introduction: Why LinkML?


sierra-moxon

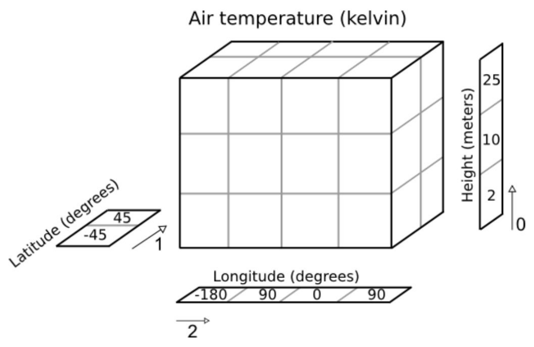
Overview Repositories 68

Sierra Moxon
sierra-moxon · she/her

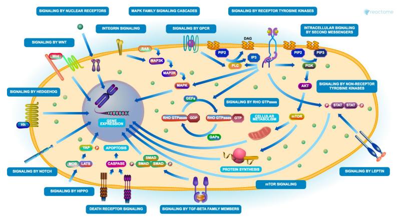
orcid: 0000-0002-8719-7760

Sierra Moxon
https://github.com/sierra-moxon

Biological data is complex…



- clearly labeled attributes
- whole numbers
- all integers
- harmonized units
- everyone is recording the same attributes
- easy to compare and reuse



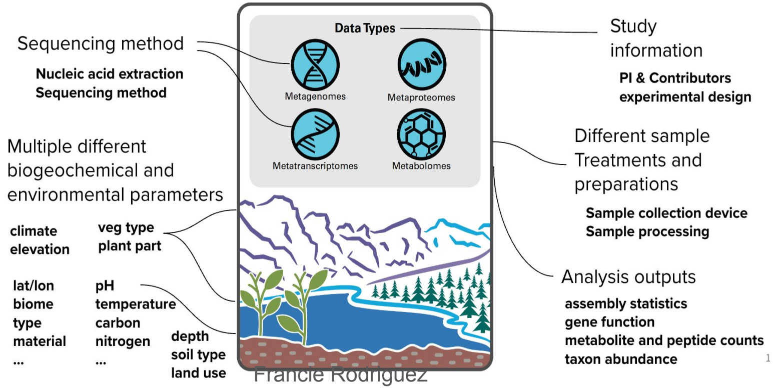
GENEONTOLOGY Unifying Biology

https://www.mdpi.com/jour

ALLIANCE of GENOME RESOURCES

NIH NCATS TRANSLATOR Funded by NIH's National Center for Advancing Translational Sciences

bio link



nmdc National Microbiome Data Collaborative

- Complex, relational, contextual knowledge
- >100k of named entities and terms
- Most knowledge exists in unstructured form (literature, figures, lab notebooks, spreadsheets)

linkML

Invalid and incomplete metadata

Unstandardized variables

Wide sparse tables

(100s of variables)

ID	dep	lat	lon	env	Ca	Mn	P	Zn	pH	C	K	meth	...	...
S1	5 cm			soil							220.8			
S2	2ft			sand							208.9			
S3	100			forest							169.3			
S4	0-20			island							148.1			
S5	3.149			gut							289.8			
S6	n/a			oil							300.3			
S7	1,5,8			root							153.7			

depth
N40.1164_W88.2543
25 santimeters
0 – 20 cm
3.149
30-60cm replicate6
Surface soil from deep water
Metamorph4 (19dpf) biological replicate 3

No global IDs

Free text categorical information

No units

Sparse data

Data Integration, Collection, and Distribution

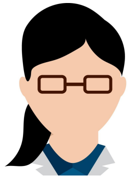
- **Start with shared standards (ontologies, etc)**
  - **reuse and contribute to existing efforts when possible!**
- Make implicit models explicit
  - use an open, community driven approach
  - meet tool developers, subject matter experts, and organizations where they are
  - make documentation easy

linkML

Example: biosample datasets

Lake Albert Sample Dataset

depth	species
22 cm	x
23 cm	x, y, z

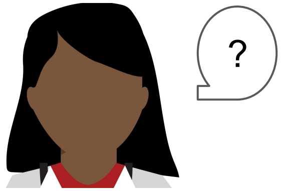


Pacific Ocean Sample Dataset

depth	species
22 cm	p
22 cm	g

Crater Lake Sample Dataset

depth	species
22 inches	x
15 feet	x, p



Can I compare bacterial compositions of bodies of water?
Can I compare the bacterial compositions of samples taken from the epipelagic zones?
Can I compare bacterial compositions from salt water samples of epipelagic zones?

Example: biosample datasets

Lake Albert Sample Dataset

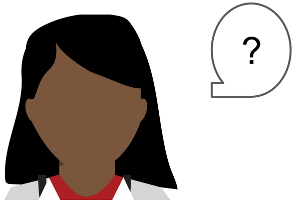
depth	species	type
22 cm	x	lake
23 cm	x, y, z	lake

Pacific Ocean Sample Dataset

depth	species	type
22 cm	p	ocean
22 cm	p	ocean

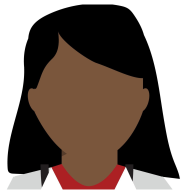
Crater Lake Sample Dataset

depth	species	type
22 inches	x	lake
15 feet	x, p	lake

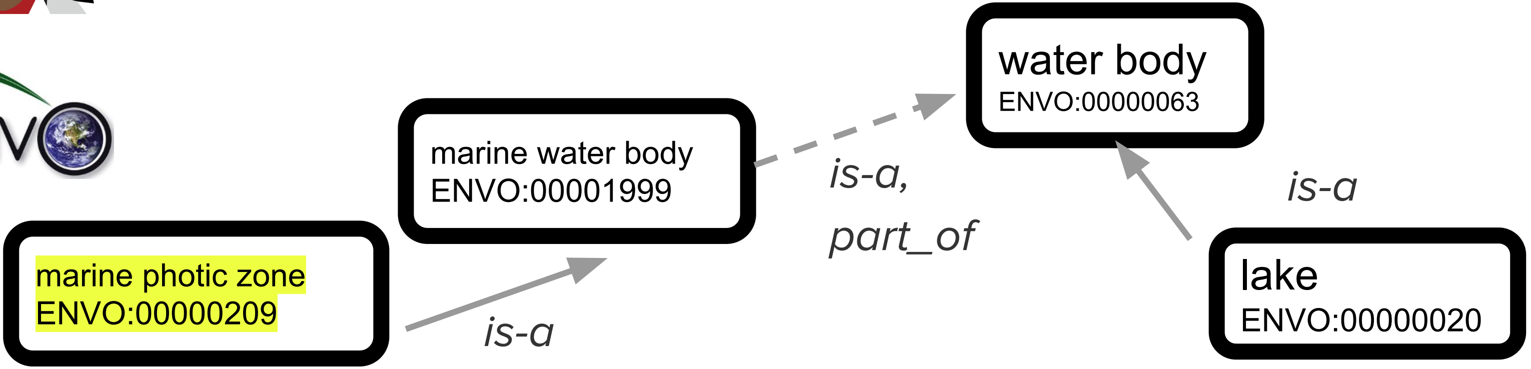


Can I compare bacterial compositions of bodies of water?
Can I compare the bacterial compositions of samples taken from the epipelagic zones?
Can I compare bacterial compositions from salt water samples of epipelagic zones?

Common vocabularies are key



?
Can I compare bacterial compositions of bodies of water? ✓
Can I compare the bacterial compositions of samples taken from the epipelagic zones ? ✓
Can I compare bacterial compositions from salt water samples?



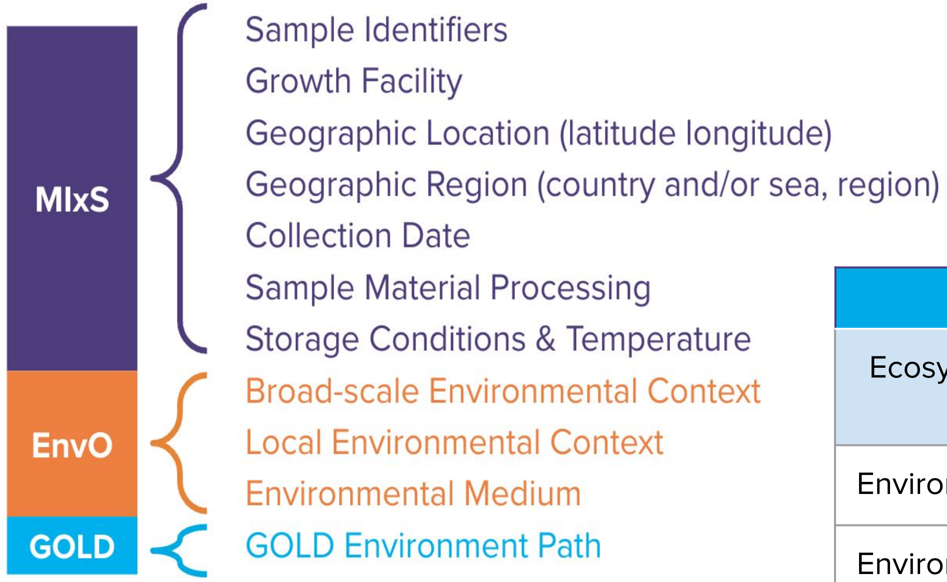
Pacific Ocean Sample Dataset

id	depth	species	type
PO1	22 cm	p	ENVO:00000209
PO2	22 cm	p	ENVO:00000209

Crater Lake Sample Dataset

id	depth	species	type
CL1	22 inches	x	ENVO:00000020
CL2	15 feet	x, p	ENVO:00000020

Ontologies combined in annotations



Sample Identifiers
Growth Facility
Geographic Location (latitude longitude)
Geographic Region (country and/or sea, region)
Collection Date
Sample Material Processing
Storage Conditions & Temperature
Broad-scale Environmental Context
Local Environmental Context
Environmental Medium
GOLD Environment Path

MIxS / EnvO

Broad-scale environment	Local-scale environment	Environmental Medium
Freshwater lake biome	Lake Shore	Sediment

GOLD Ecosystem classification

Ecosystem	Ecosystem Category	Ecosystem Type	Specific Ecosystem	Ecosystem Tree
Environment	Aquatic	Freshwater	Lake	Sediment
Environment	Aquatic	Freshwater	Lake	Algal bloom

Francie Rodriguez

nmdc National Microbiome Data Collaborative

Are ontologies enough?

- Start with ontologies
  - reuse and contribute to existing efforts when possible!
- **Make implicit models explicit**
  - use an open, community driven approach
  - meet tool developers, subject matter experts, and organizations where they are
  - make documentation easy

id	depth	species	type
CL1	22 inches	x	ENVO:00000020
CL2	15 feet	x, p	ENVO:00000020

depth
N40.1164_W88.2543
25 santimeters
0 – 20 cm
3.149
30-60cm replicate6
Surface soil from deep water
Metamorph4 (19dpf) biological replicate 3

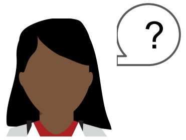
Models hiding in plain sight

Pacific Ocean Sample Dataset

depth	species	type
22 cm	p	ENVO:00001999
22 cm	g	ENVO:00001999

Crater Lake Sample Dataset

depth	species	type
22 inches	x	ENVO:00000020
15 feet	x, p	ENVO:00000020



These are “standards” (and “models”), but they are not computable without a human.

Existing frameworks not designed for interop

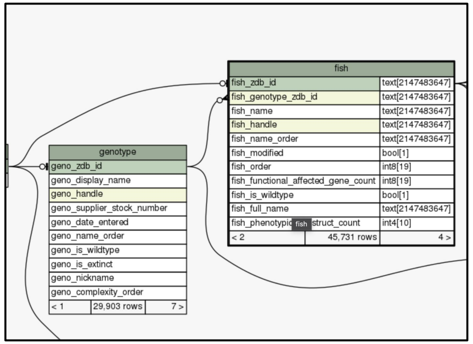
Pacific Ocean Sample

```
CREATE TABLE biosample (
  acc varchar primary
key,
  depth float,
  lat float,
  long float,
  environment varchar
  …
)
```

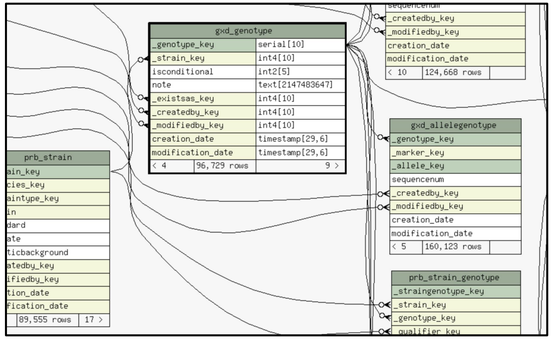
Crater Lake Sample Dataset

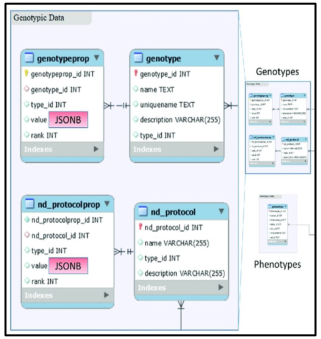
```
CREATE TABLE lake_sample (
  id varchar primary key,
  depth foreign key,
  location foreign key,
  environment foreign key
  …
)
```

https://www.mdpi.com/journal/dna



genotype
geno_zdb_id
geno_display_name
geno_handle
geno_supplier_stock_number
geno_date_entered
geno_name_order
geno_is_wildtype
geno_is_extinct
geno_nickname
geno_complexity_order
< 1 | 29,903 rows | 7 >
fish
fish_zdb_id text[2147483647]
fish_genotype_zdb_id text[2147483647]
fish_name text[2147483647]
fish_handle text[2147483647]
fish_name_order text[2147483647]
fish_modified bool[1]
fish_order int8[19]
fish_functional_affected_gene_count int8[19]
fish_is_wildtype bool[1]
fish_full_name text[2147483647]
fish_phenotypic struct_count int4[10]
fish
< 2 | 45,731 rows | 4 >

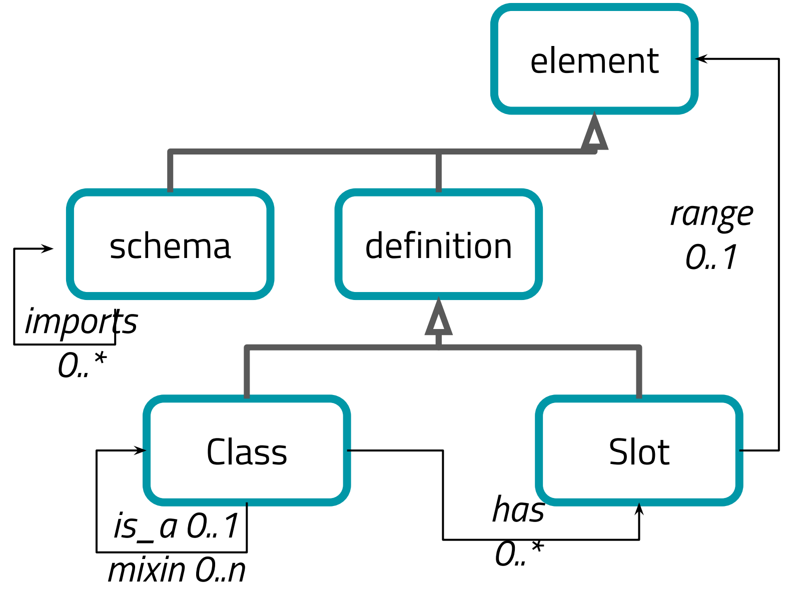




LinkML: Modeling Language & Toolkit

THE STANDARD

*A **meta-datamodel** for structuring your data*



TOOLS

Pragmatic developer and curator friendly tools for working with data

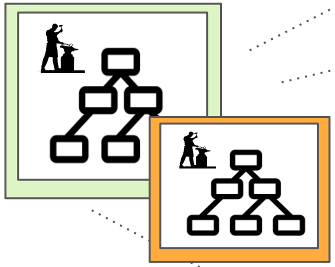
- **Validators**
- **Data Converters**
- **Compatibility tools**
- **Data entry**
- **Schema inference**

LinkML Syntax

Pacific Ocean Sample Database

depth	salinity	bacteria	latitude	longitude	sample_type
22 cm	35	x,p	44.8084° N	24.0632° W	ENVO:00001999

LinkML ❤️'s ontologies

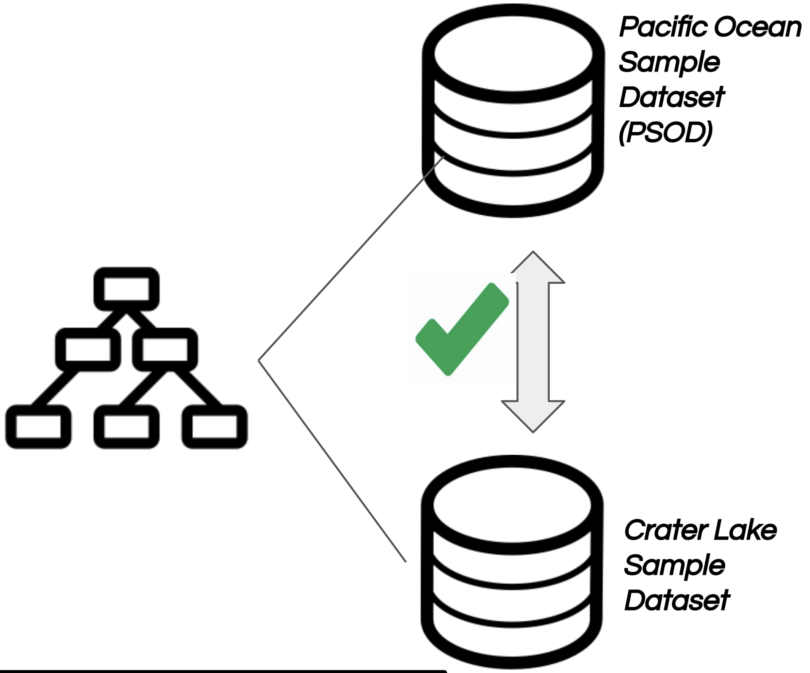


```yaml
imports:
  linkml:types
classes:
  Sample:
   description: a sample of biological material.
   attributes:
      depth:
            slot_uri: ENVO:3100031
      salinity:
          exact_mappings:
                -PATO:0085001
      bacteria:
         multivalued: true
      latitude:
          type: string
      longitude:
      sample_type:
         required: true
         type:
            range: SampleType
enums:
   SampleType:
      reachable_from:
            source_ontology: obo:envo
```

Import LinkML models from other LinkML models

```
classes:
  BioSample:
   description: a sample of biological material.
   attributes:
      longitude:
      latitude:
      depth:
        slot_uri: ENVO:3100031
      depth_units:
——---------------------------------------------------------
imports:
   -    PSOD:Biosample
LakeSample:
    is_a: BioSample
    attributes:
       salinity:
          exact_mappings:
             -PATO:0085001
```

```
slot_usage:
        depth_units:
              equals_string: cm
```


Pacific Ocean Sample Dataset (PSOD)
Crater Lake Sample Dataset

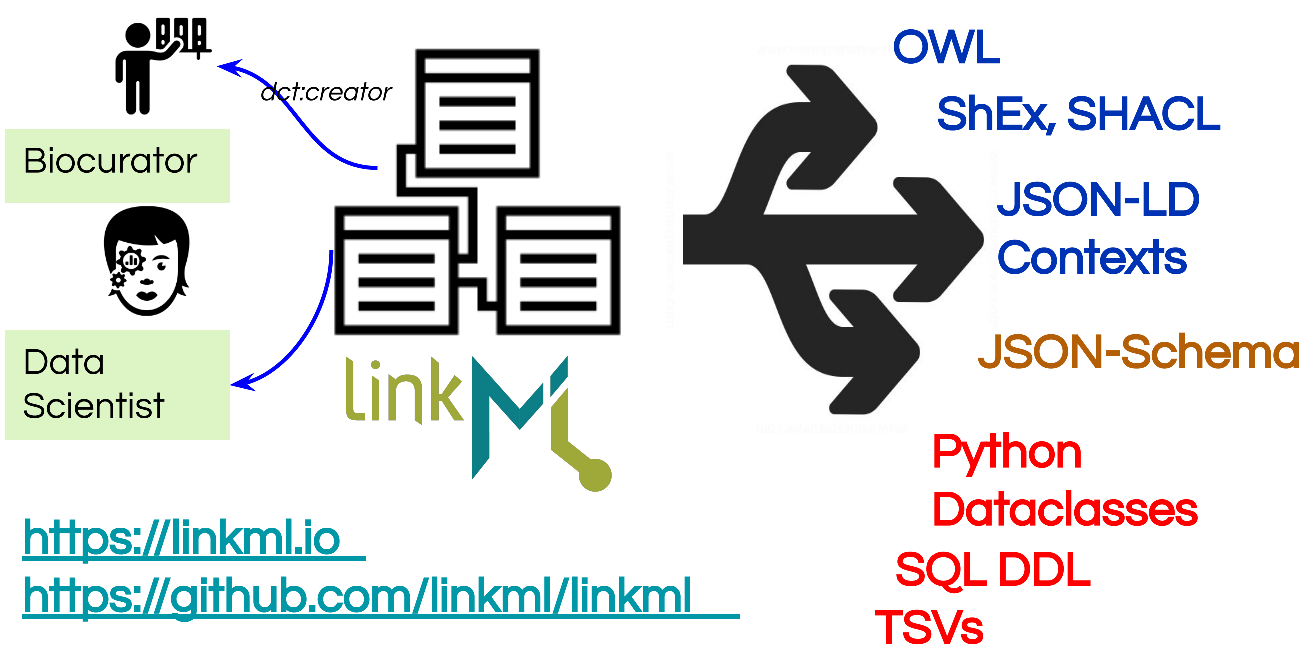
Take the tutorial! https://github.com/linkml/linkml-tutorial https://linkml.io/linkml/intro/tutorial.html

LinkML as a universal converter box

Create data models in simple YAML files, optionally annotated using ontologies

Compile to other frameworks

Choose the right tools for the job; no lock-in



Semantic Web Applications and Infrastructure

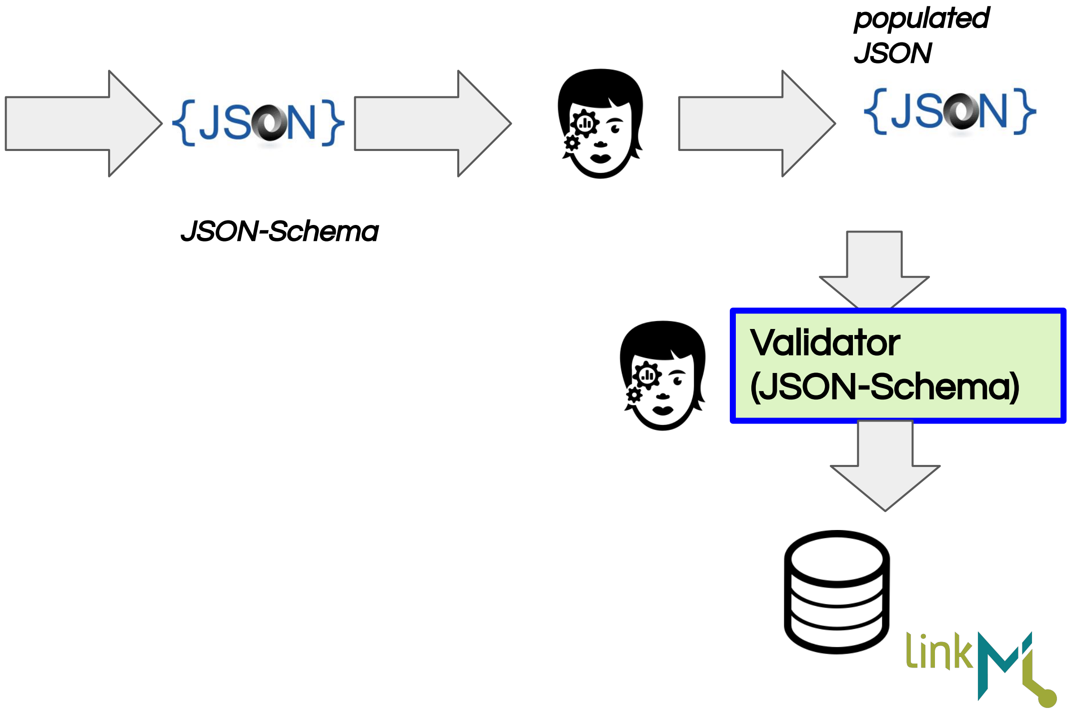
"Traditional" Applications and Infrastructure

https://linkml.io

https://github.com/linkml/linkml

LinkML has built in validators

```
classes:
  BioSample:
   description: a sample of biological material.
   attributes:
      depth:
        slot_uri: ENVO:3100031
      species:
          multivalued: true
      salinity:
          exact_mappings:
              -PATO:0085001
      longitude:
      latitude:
      type:
          required: true
          range: EnviornmentEnum
enums:
   EnvironmentEnum
      reachable_from:
          source_ontology: ENVO
```



Generate LinkML

Get intelligent assistance from auto schema tools

refine

Semi-structured data sources

Schema-automator

```yaml
id: https://example.org/linkml/hello-world
title: Really basic LinkML model
name: hello-world
version: 0.0.1

prefixes:
 linkml: https://w3id.org/linkml/
 sdo: https://schema.org/
 ex: https://example.org/linkml/hello-world/

default_prefix: ex
default_curi_maps:
 - semweb_context

imports:
 - linkml:types

classes:
 Person:
   description: Minimal information about a person
   class_uri: sdo:Person
   attributes:
     id:
       identifier: true
       slot_uri: sdo:taxID
     first_name:
       required: true
       slot_uri: sdo:givenName
       multivalued: true
     last_name:
       required: true
       slot_uri: sdo:familyName
     knows:
       range: Person
       multivalued: true
       slot_uri: foaf:knows
```

Metadata

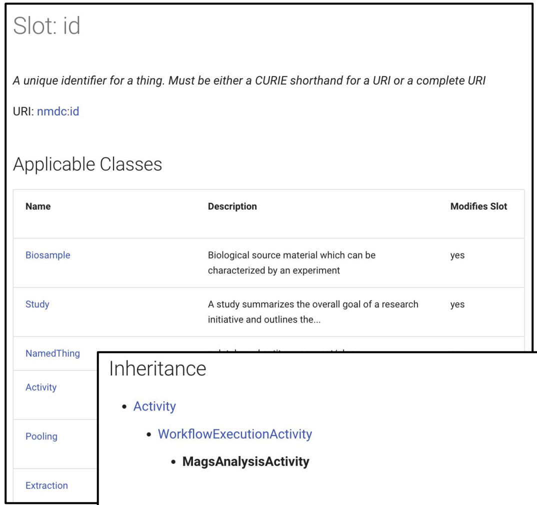
Namespaces

Dependencies

Actual Datamodel

YAML conformant to LinkML standard

LinkML auto-generates documentation



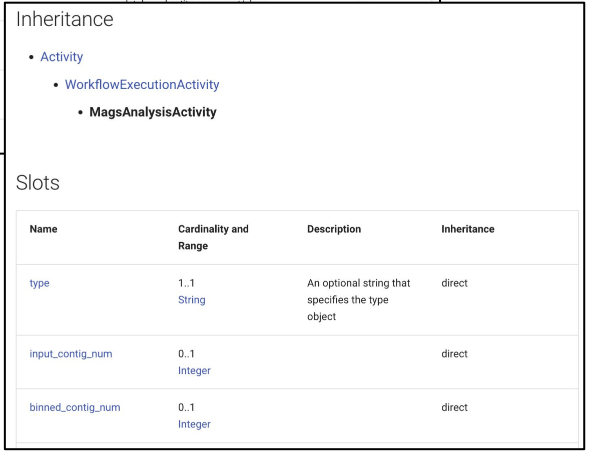
Slot: id

A unique identifier for a thing. Must be either a CURIE shorthand for a URI or a complete URI

URI: nmdc:id

Applicable Classes

Name	Description	Modifies Slot
Biosample	Biological source material which can be characterized by an experiment	yes
Study	A study summarizes the overall goal of a research initiative and outlines the...	yes
NamedThing		
Activity		
Pooling		
Extraction		

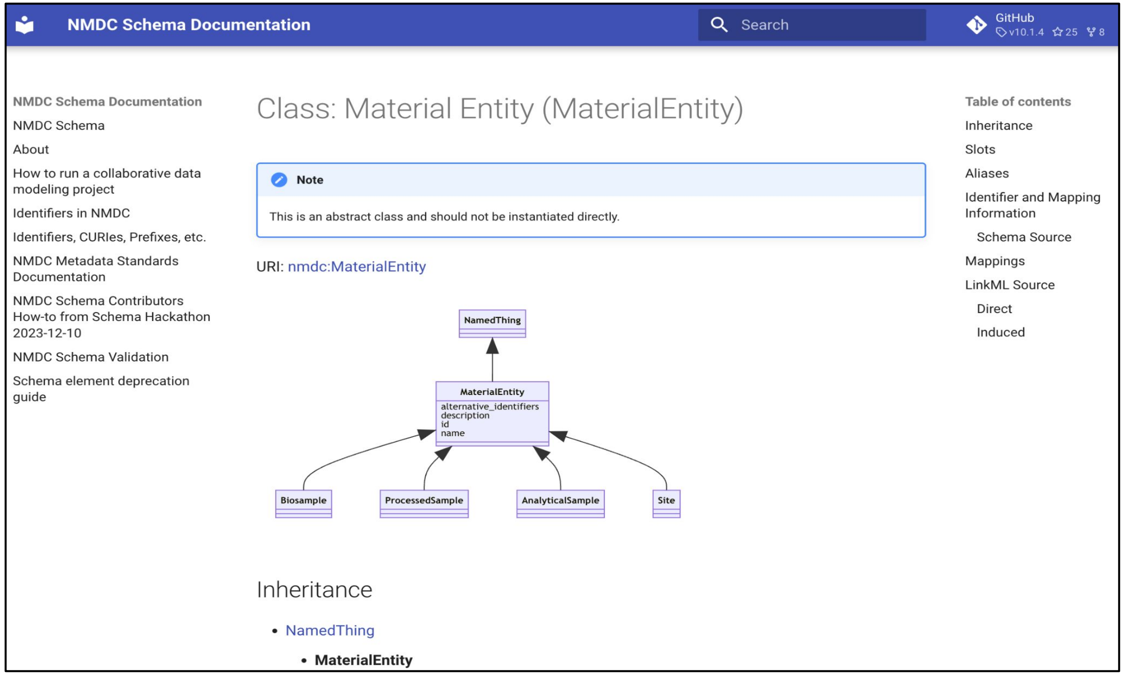


Inheritance

- Activity
  - WorkflowExecutionActivity
    - **MagsAnalysisActivity**

Slots

Name	Cardinality and Range	Description	Inheritance
type	1..1 String	An optional string that specifies the type object	direct
input_contig_num	0..1 Integer		direct
binned_contig_num	0..1 Integer		direct



NMDC Schema Documentation
- NMDC Schema
- About
- How to run a collaborative data modeling project
- Identifiers in NMDC
- Identifiers, CURIes, Prefixes, etc.
- NMDC Metadata Standards Documentation
- NMDC Schema Contributors How-to from Schema Hackathon 2023-12-10
- NMDC Schema Validation
- Schema element deprecation guide

Class: Material Entity (MaterialEntity)

Note

This is an abstract class and should not be instantiated directly.

URI: nmdc:MaterialEntity

Inheritance

- NamedThing
  - **MaterialEntity**

https://microbiomedata.github.io/nmdc-schema/

nmdc National Microbiome Data Collaborative

Onwards!

Enough with the intro already!

- Let's LinkML-ize!!!

Rest stops along the way

- **We hope you follow along with us as we build up our project!**
- If you end up lost, each section of the tutorial has a corresponding tag in the linkml-tutorial-2024 repository (github.com/linkml/linkml-tutorial-2024)
- You can clone the linkml-tutorial-2024 repository and checkout the appropriate tag.

Step 0 - basic project creation

Step 1 - modeling

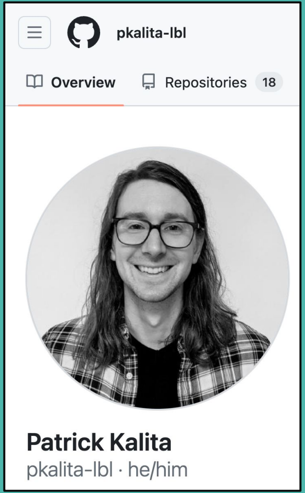
Step 2 - linting

Step 3 - documentation

Step 4 - code generation

Step 5 - validation

Section 0: Setting up a LinkML project


pkalita-lbl

Overview

Repositories 18

Patrick Kalita

pkalita-lbl · he/him

Patrick Kalita
https://github.com/pkalita-lbl

bit.ly/LinkML-2024

Option A: Write your schema in LinkML YAML

Option A: Author YAML *directly*

Optional
productivity tools

Visual Studio Code

GitHub Copilot

LINKML-LINTER
clean schemas, naturally

```yaml
id: https://example.org/linkml/hello-world
title: Really basic LinkML model
name: hello-world
version: 0.0.1

prefixes:
 linkml: https://w3id.org/linkml/
 sdo: https://schema.org/
 ex: https://example.org/linkml/hello-world/

default_prefix: ex
default_curi_maps:
 - semweb_context

imports:
 - linkml:types

classes:
 Person:
   description: Minimal information about a person
   class_uri: sdo:Person
   attributes:
     id:
       identifier: true
       slot_uri: sdo:taxID
     first_name:
       required: true
       slot_uri: sdo:givenName
       multivalued: true
     last_name:
       required: true
       slot_uri: sdo:familyName
     knows:
       range: Person
       multivalued: true
       slot_uri: foaf:knows
```

Metadata

Namespaces

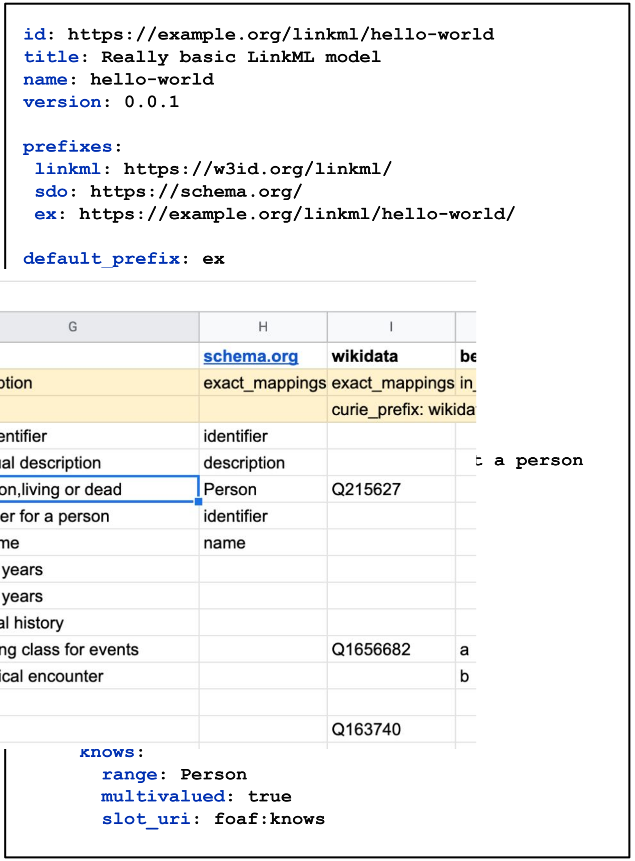
Dependencies

Actual data model

YAML conformant to LinkML standard

Option B: Use Excel or Google Sheets

Option B: Author using *schemasheets*



```
id: https://example.org/linkml/hello-world
title: Really basic LinkML model
name: hello-world
version: 0.0.1

prefixes:
 linkml: https://w3id.org/linkml/
 sdo: https://schema.org/
 ex: https://example.org/linkml/hello-world/

default_prefix: ex
```

```
: a person
```

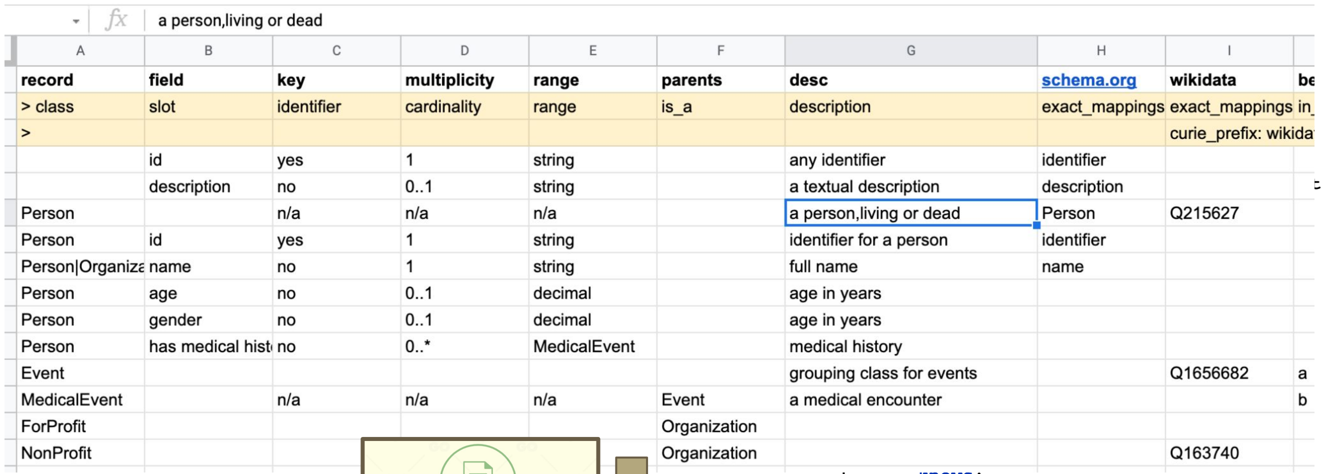
```
    knows:
      range: Person
      multivalued: true
      slot_uri: foaf:knows
```

Metadata

Namespaces

Dependencies

Actual Datamodel



a person,living or dead

A	B	C	D	E	F	G	H	I	
record	field	key	multiplicity	range	parents	desc	schema.org	wikidata	be
> class	slot	identifier	cardinality	range	is_a	description	exact_mappings	exact_mappings	in_
>								curie_prefix: wikida	
	id	yes	1	string		any identifier	identifier		
	description	no	0..1	string		a textual description	description		
Person		n/a	n/a	n/a		a person,living or dead	Person	Q215627	
Person	id	yes	1	string		identifier for a person	identifier		
Person|Organiza	name	no	1	string		full name	name		
Person	age	no	0..1	decimal		age in years			
Person	gender	no	0..1	decimal		age in years			
Person	has medical hist	no	0..*	MedicalEvent		medical history			
Event						grouping class for events		Q1656682	a
MedicalEvent		n/a	n/a	n/a	Event	a medical encounter			b
ForProfit					Organization				
NonProfit					Organization			Q163740	

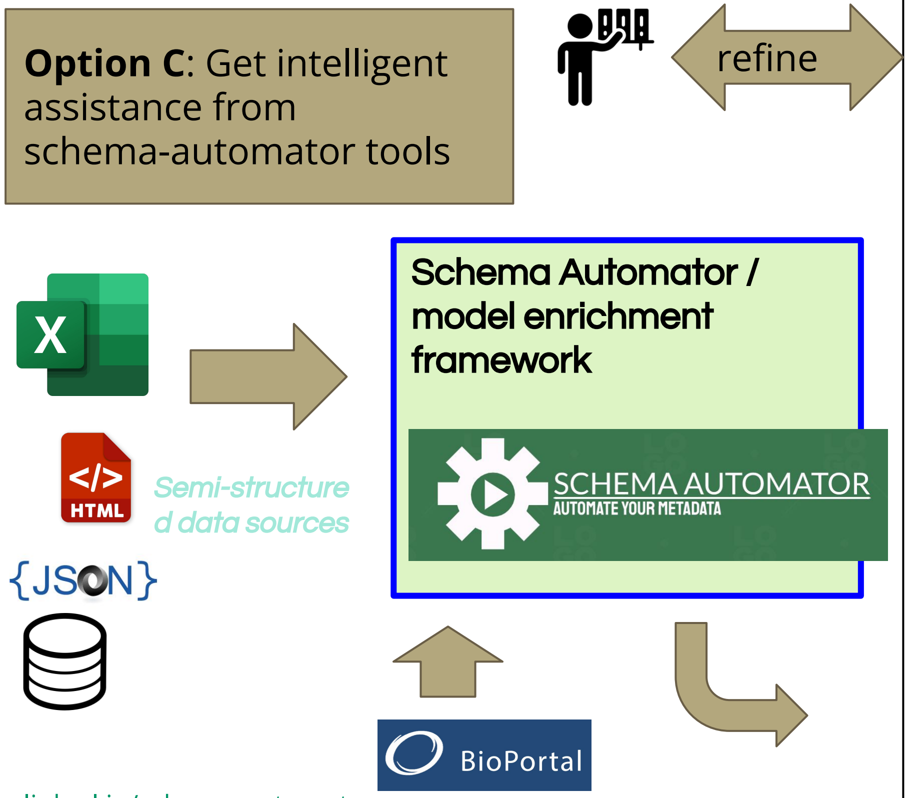
SCHEMASHEETS
easy LinkML from spreadsheets

linkml.io/schemasheets

YAML conformant to LinkML standard

Option C: Use semi-automated approaches

Option C: Get intelligent assistance from schema-automator tools


refine
Semi-structured data sources
Schema Automator / model enrichment framework
SCHEMA AUTOMATOR
AUTOMATE YOUR METADATA
HTML
{JSON}
BioPortal

linkml.io/schema-automator

```yaml
id: https://example.org/linkml/hello-world
title: Really basic LinkML model
name: hello-world
version: 0.0.1

prefixes:
 linkml: https://w3id.org/linkml/
 sdo: https://schema.org/
 ex: https://example.org/linkml/hello-world/

default_prefix: ex
default_curi_maps:
 - semweb_context

imports:
 - linkml:types

classes:
 Person:
   description: Minimal information about a person
   class_uri: sdo:Person
   attributes:
     id:
       identifier: true
       slot_uri: sdo:taxID
     first_name:
       required: true
       slot_uri: sdo:givenName
       multivalued: true
     last_name:
       required: true
       slot_uri: sdo:familyName
     knows:
       range: Person
       multivalued: true
       slot_uri: foaf:knows
```

Metadata

Namespaces

Dependencies

Actual Datamodel

YAML conformant to LinkML standard

LinkML schema repositories come with best practices

- Licensing information
- Git initialization and actions
- Generated documentation
- Schema linting
- An automatic update system so that projects do not get out of date as LinkML evolves

☀ These benefits come bundled in a template that can be customized on initialization: the linkml-project-cookiecutter

linkml-project-cookiecutter overview

- Ensure we have the right tools installed (one-time setup)
- Create and setup the project

A brief word about virtual environments

A brief word about virtual environments

linkml==1.7.3

pydantic>=2.0

click

...

A brief word about virtual environments

linkml==1.7.3

pydantic>=2.0

click

...

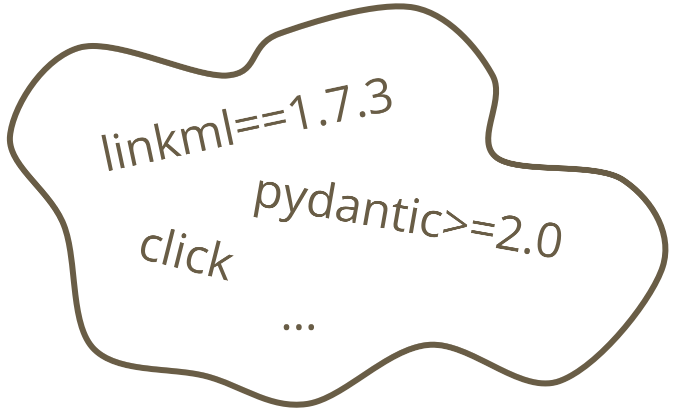
linkml==1.6.7

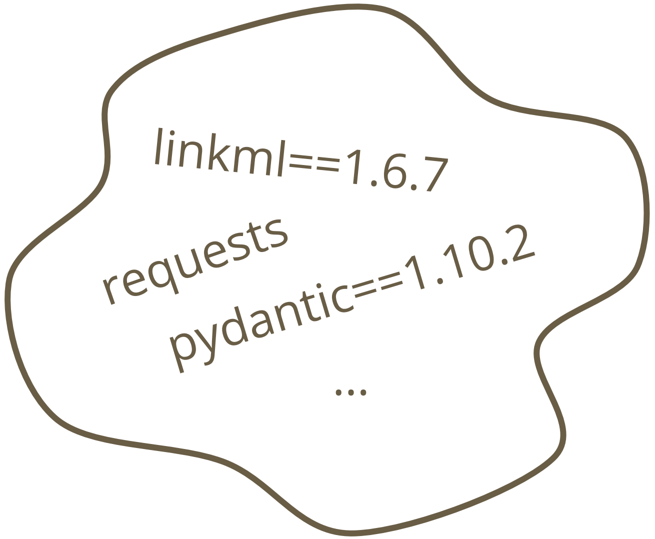
requests

pydantic==1.10.2

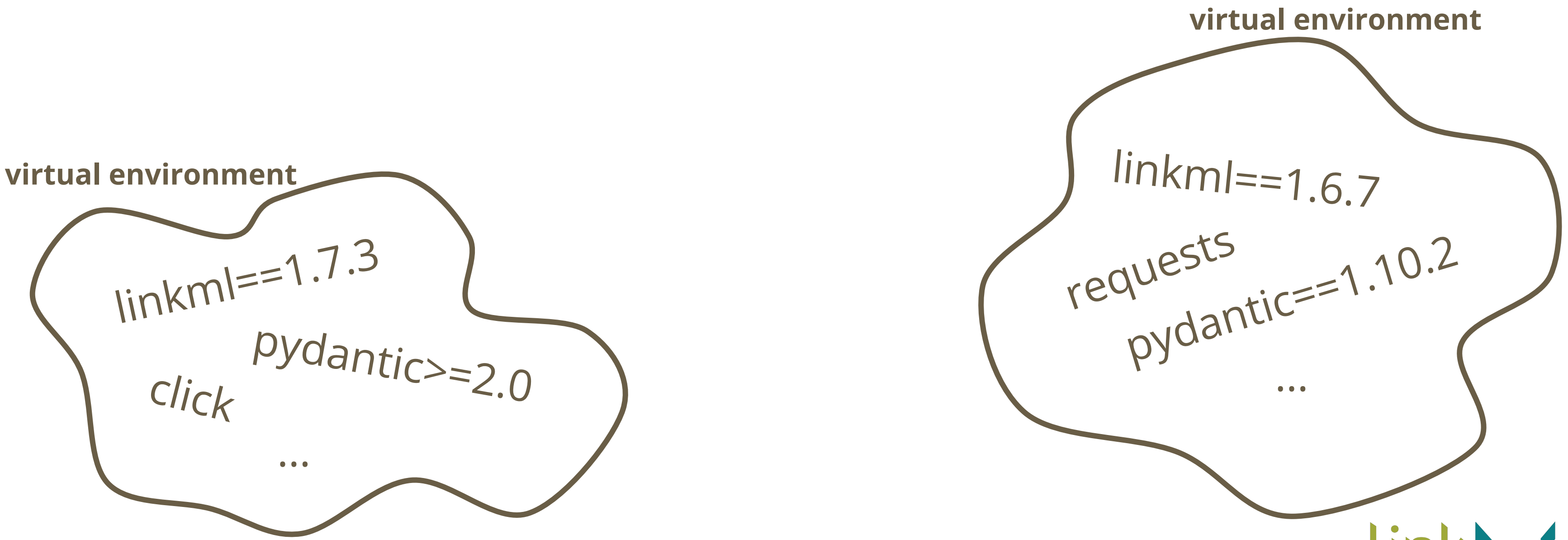
...

A brief word about virtual environments

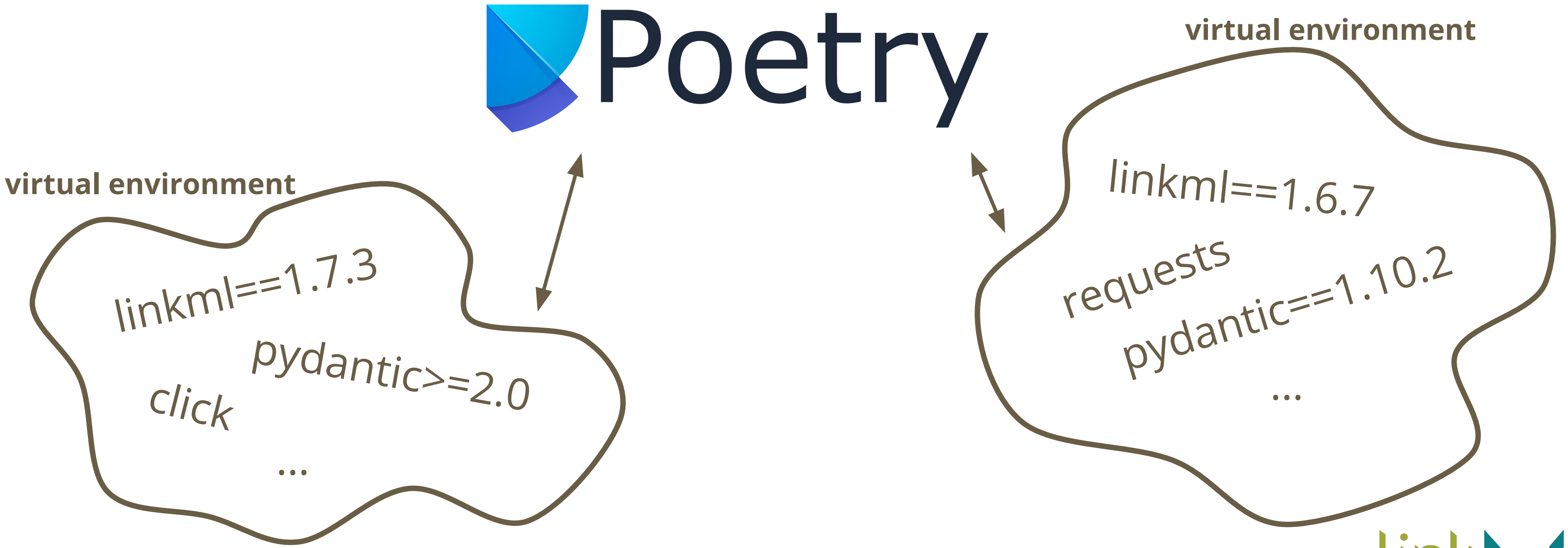




A brief word about virtual environments



A brief word about virtual environments



Cookiecutter step 1: install prerequisites

- Check to see if you already have Poetry installed

```
> poetry --version
```

- If you get a “command not found” error, install Poetry with pipx

```
> pipx install poetry
```

- Don’t have pipx installed? See https://pipx.pypa.io/
- Verify Poetry is installed

```
> poetry --version
```

Cookiecutter step 1: install prerequisites

- Check to see if you already have cruft installed

```
> cruft --help
```

- If you get a “command not found” error, install cruft with pipx

```
> pipx install cruft
```

- Verify that cruft is installed

```
> cruft --help
```

Cookiecutter step 1: install prerequisites

- Check to see if git is configured correctly

```
> git config --global user.name
> git config --global user.email
> git config --global init.defaultBranch
```

- If any do not return something, set the values as needed

```
> git config --global user.name "Your Name"
> git config --global user.email "you@example.org"
> git config --global init.defaultBranch main
```

Cookiecutter step 2: create LinkML project

```
> cd <directory where you want to create your new project directory>

> cruft create https://github.com/linkml/linkml-project-cookiecutter
```

- You will be prompted to enter a few values, like:
  - ***name***: linkml-tutorial-2024
  - ***github_org***: <org-name>
  - ***description***: brief one line description of schema
  - ***full_name***: full name of schema author
  - ***email***: email id of schema author
  - ***main_schema_class***: Person
  - ***create_python_project***: set project up as Python project with dataclasses

Cookiecutter step 3: set up LinkML project

- The setup process takes care of 3 things for you:
    - Creation of a virtual environment and installation of listed dependencies within it
    - Generation of all artifacts by LinkML suite of generators
    - Generation of Markdown and HTML documentation
    - Initialization of schema project as Git repository

```
> cd linkml-tutorial-2024
> make setup
```

Cookiecutter bonus step 4: pushing to GitHub

- Go to https://github.com/new and follow the instructions
  - Being sure to NOT add a README,. gitignore file or a LICENSE file (the cookiecutter template will take care of this for you)
- Add the remote to your local git repository

```
> git remote add origin https://github.com/<my-org>/linkml-tutorial-2024.git
> git push -u origin main
```

Checking in...Rest Stops Along the Way

- **We hope you follow along with us as we build up our project!**
- If you end up lost, each section of the tutorial has a corresponding tag in the linkml-tutorial-2024 repository (github.com/linkml/linkml-tutorial-2024)
- You can clone the linkml-tutorial-2024 repository and checkout the appropriate tag.

Section 0 - basic project creation

Step 1 - modeling

Step 2 - linting

Step 3 - documentation

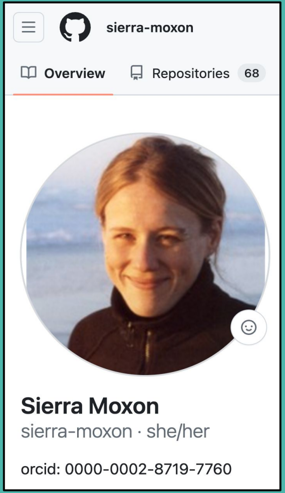
Step 4 - code generation

Step 5 - validation

Feeling lost?

```
> git clone https://github.com/linkml/linkml-tutorial-2024.git
> git checkout step_0_project_setup
```

Section 1: Developing the Model


sierra-moxon

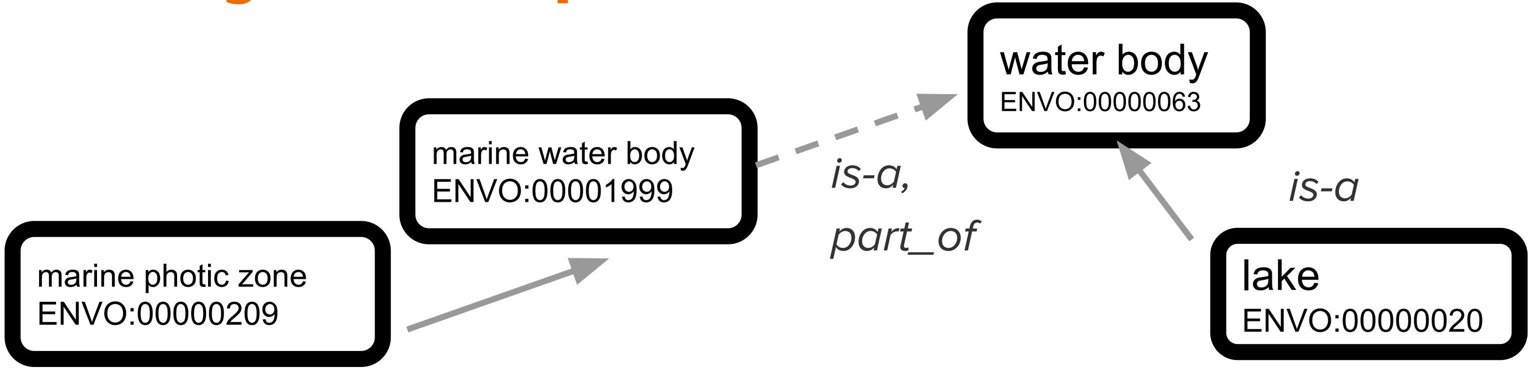
Overview Repositories 68

Sierra Moxon
sierra-moxon · she/her

orcid: 0000-0002-8719-7760

Sierra Moxon
https://github.com/sierra-moxon

Describing a Biosample



Pacific Ocean Sample Dataset

id	depth	species	type
PO1	22 cm	p	ENVO:00000209
PO2	22 cm	p	ENVO:00000209

id	depth	species	type
CL1	22 inches	x	ENVO:00000020
CL2	15 feet	x, p	ENVO:00000020

LinkML Metamodel Syntax

LinkML Model (YAML)

```
Classes:
  Biosample:
Slots:
   latitude:
     range: integer
```

Java

```
package org.biosample.model

import java.util.List;
import lombok.*;

@Data
@EqualsAndHashCode(callSuper=false)
public class Biosample {private Integer latitude}
```

Pydantic Classes

```
class Biosample(NamedThing):
    latitude: Optional[int] = Field(None,
description="""the latitude of the sample""")
```

OWL (ttl)

```
owl:Person a owl:Class

person:age_in_years a
owl:ObjectProperty;
rdfs:label “age in years”;
rdfs:range int;
```

SQLDDL

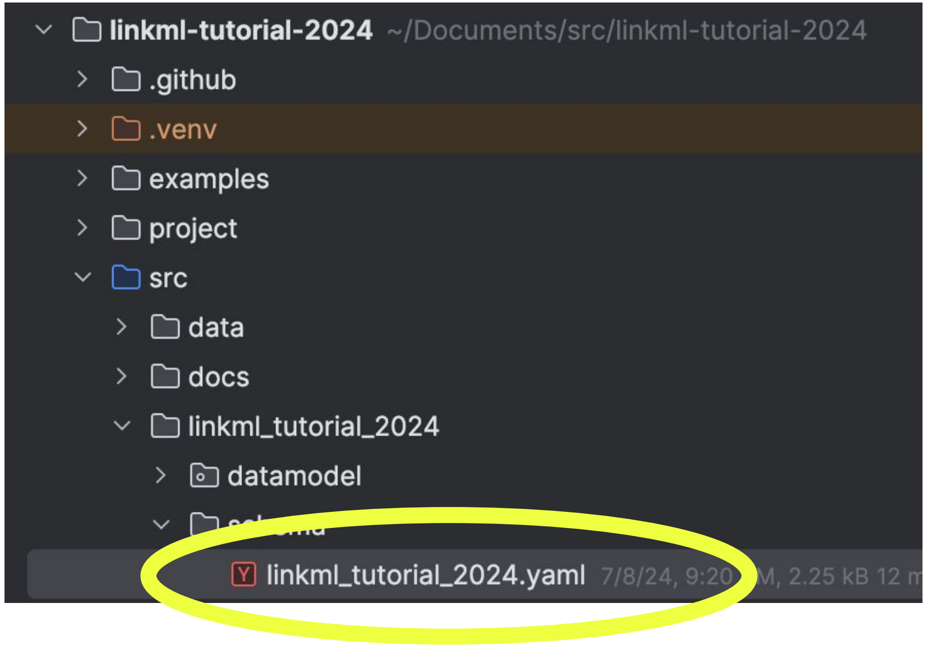
```
CREATE TABLE "Biosample"
(latitude INTEGER)
```

JSONSchema

```
“Biosample”: {
   “properties”: {
        “latitude”: {
            “type”: “integer”
        }
    }
}
```

Step 1: Model Components

Documentation: https://linkml.io/linkml/schemas/models.html



```
⌄ linkml-tutorial-2024  ~/Documents/src/linkml-tutorial-2024
  › .github
  › .venv
  › examples
  › project
  ⌄ src
    › data
    › docs
    ⌄ linkml_tutorial_2024
      › datamodel
      ⌄ schema
          linkml_tutorial_2024.yaml  7/8/24, 9:20 AM, 2.25 kB 12 m
```

Assignment: Add a class with attributes

Problem statement 1: A researcher is collecting data on microbial content of water samples and they want to ensure that each sample has the same set of collection metrics.

The researcher wants to collect information about the sample site, including depth, latitude, longitude, the species of microbe(s) found, and a consistent way of representing the biome of the sample site.

LinkML Classes and Slots

id	depth	species	type
PO1	22 cm	p	ENVO:00000209
PO2	22 cm	p	ENVO:00000209

LinkML modeling language:

- description
- aliases
- identifier
- range
- multivalued
- required

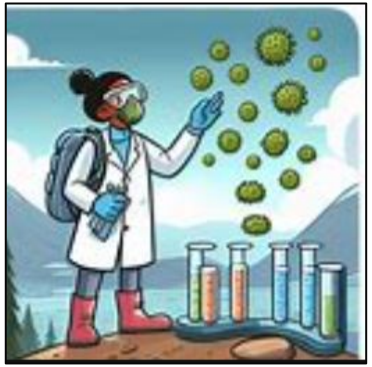
LinkML

```
Sample:
    aliases: ["Biosample", "Environmental Sample"]
    description: >-
      A sample is a limited quantity of something (e.g. an individual or set of individuals from a population, or a portion of a substance) to be used for testing, analysis, inspection, investigation, demonstration, or trial use.
    slots:
      - id
      - latitude
      - longitude
      - species
      - sample_biome
```

Assignment: Establish a hierarchy of classes

Problem statement 2: Researcher establishes a collaboration with another PI who wants to collect samples from the air. Air has different attributes than water samples, but many are the same. The second researcher would like to analyze data from both collected datasets.

Extend the model to allow collection of both kinds of data.



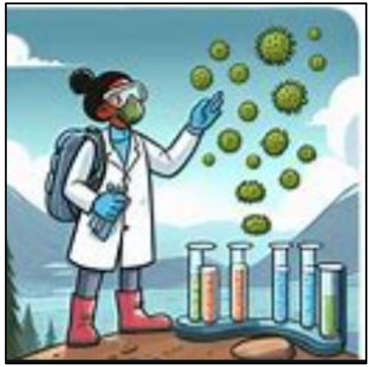
Hierarchies

- Hierarchical classes

depth
latitude
longitude
biome

shared characteristics

latitude
longitude
biome



altitude
latitude
longitude
biome

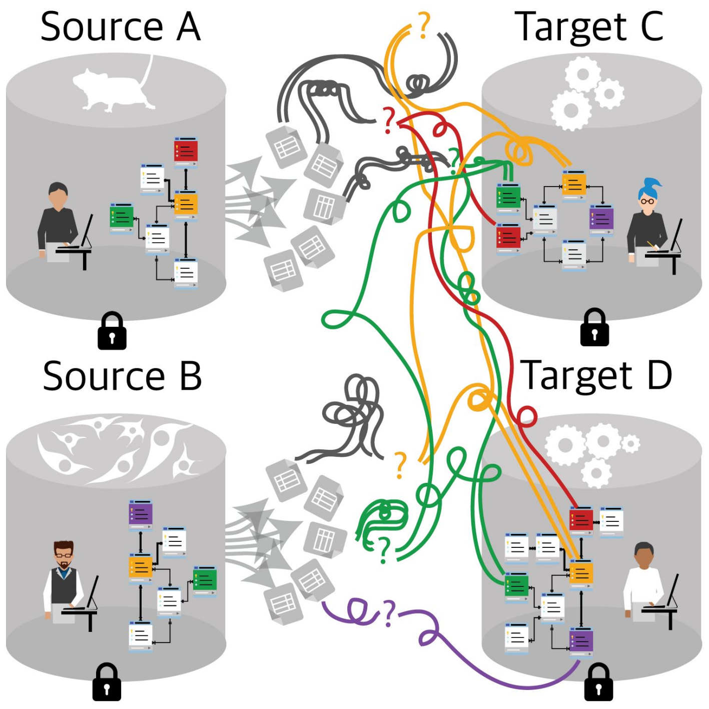
- Helpful testing infrastructure
  - Sample_Collection
    - tree_root

Assignment: Make sure model is interoperable

Problem statement: After publication, other researchers start reaching out and making Researcher 1 and 2 aware of alternative models for capturing Sample metadata. They ask the data producers to give them a translation table to map collected samples to the National Microbiome Data Collaborative model. Then they ask how the new model relates to the work done at SIO ontology.

Ensure there is a computable way to map a class in the model to other models.

Definition refinement, mappings, and URIs



LinkML To The Rescue!

- mappings
- URIs

Assignment: Constrain and Train

Problem statement: The schema Researcher 1 and 2 have put together has become widely used for several groups collecting sample data. So much so that research 1 and 2 have had to delegate maintenance of the schema and datasets to their staff. Over time, it's apparent that training helps with consistency of data entry. But neither researcher 1 nor 2 have time each semester to train their students. They need the model to do the heavy lifting in terms of keeping the data harmonized.

beach biome?

Beach
Sandbox
Desert
Freshwater lake sand

Ontology support via value sets (enums)

```
prefixes:
  COB: http://purl.obolibrary.org/obo/COB_
  BFO: http://purl.obolibrary.org/obo/BFO_
  RO: http://purl.obolibrary.org/obo/RO_
  CHEBI: http://purl.obolibrary.org/obo/CHEBI_
  CHEMINF: http://semanticscience.org/resource/CHEMINF_
  SIO: http://semanticscience.org/resource/SIO_
  PUBCHEM.ELEMENT: https://pubchem.ncbi.nlm.nih.gov/element/
  LANL.ELEMENT: https://periodic.lanl.gov/
```

```
enums:

  nanostructure_morphology_enum:
    permissible_values:
      nanotube:
        meaning: CHEBI:50796
      nanoparticle:
        meaning: CHEBI:50803
      nanorod:
        meaning: CHEBI:50805
      nanotubosome:
        meaning: CHEBI:50806
      quantum dot:
        meaning: CHEBI:50853
      nanofibre:
        meaning: CHEBI:52518
      nanocrystal:
        meaning: CHEBI:52529
      nanoribbon:
        meaning: CHEBI:52530
      nanosheet:
        meaning: CHEBI:52531
      nanowire:
        meaning: CHEBI:52593
```

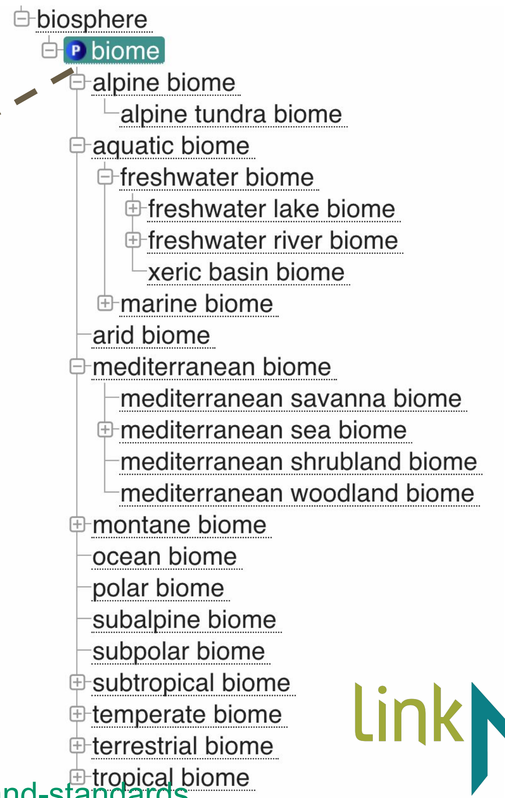
ChEBI

- *Used to constrain categorical values*
- *Better than free text strings*
- *Can drive drop-downs*

https://github.com/chemkg/chemrof
(chemical entity data model)

Ontology support via value sets (enums)

```
enums:
  Biome:
    reachable_from:
      source_ontology: obo:envo
      source_nodes:
        - ENVO:00000428  ## biome
      include_self: false
      is_direct: false
      relationship_types:
        - rdfs:subClassOf
```



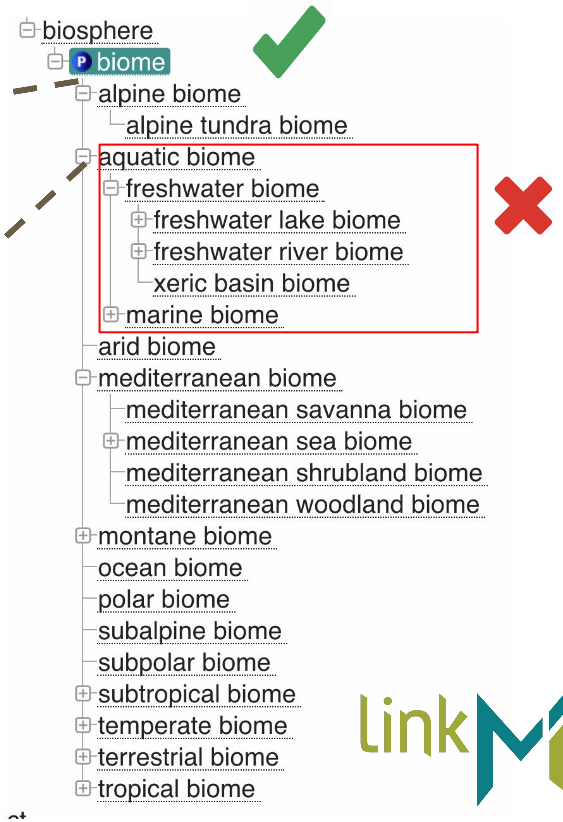
- biosphere
  - biome
    - alpine biome
      - alpine tundra biome
    - aquatic biome
      - freshwater biome
        - freshwater lake biome
        - freshwater river biome
        - xeric basin biome
      - marine biome
    - arid biome
    - mediterranean biome
      - mediterranean savanna biome
      - mediterranean sea biome
      - mediterranean shrubland biome
      - mediterranean woodland biome
    - montane biome
    - ocean biome
    - polar biome
    - subalpine biome
    - subpolar biome
    - subtropical biome
    - temperate biome
    - terrestrial biome
    - tropical biome

http://environmentontology.org

https://douroucouli.wordpress.com/2022/07/15/using-ontologies-within-data-models-and-standards

Boolean combinations supported

```
enums:
  NonAquaticBiome:
    reachable_from:
      source_ontology: obo:envo
      source_nodes:
        - ENVO:00000428  ## biome
      include_self: false
      is_direct: false
      relationship_types:
        - rdfs:subClassOf
      minus:
        reachable_from:
        source_ontology: obo:envo
        source_nodes:
          - ENVO:00002030  ##
aquatic
        include_self: true
        is_direct: false
        relationship_types:
          - rdfs:subClassOf
```



Enumerations

- Basic enumerations annotated with ontologies
    - meaning
    - reachable_from

Other constraints

- Validating slot values with patterns
  - slot_usage
  - pattern constraints
  - id_prefixes

Rules syntax: https://linkml.io/linkml/schemas/advanced.html#rules

Assignment: Deprecation

Problem statement: Latitude and longitude are becoming less prevalent in the data as more advanced technology becomes accessible. Instead, researchers are starting use GPS coordinates. Refactor the schema so that "GPS location" replaces latitude and longitude. Be sure to think about all the labs using the model and how they will programmatically migrate their existing data and tools to use the new model

Why not delete elements?

- Give notice to downstream consumers.
- Be backward compatible with earlier versions of the schema that might be used to validate data in the wild.
- Preserve links to schema elements that may have already been published, referred to by other pages or otherwise publicly available previously.
- Reminder of the decisions that have been made.

Modeling step 7: Deprecation

- Use “deprecated” metamodel values
- Track provenance of deprecation
- Track date/time of deprecation

LinkML Metamodel components

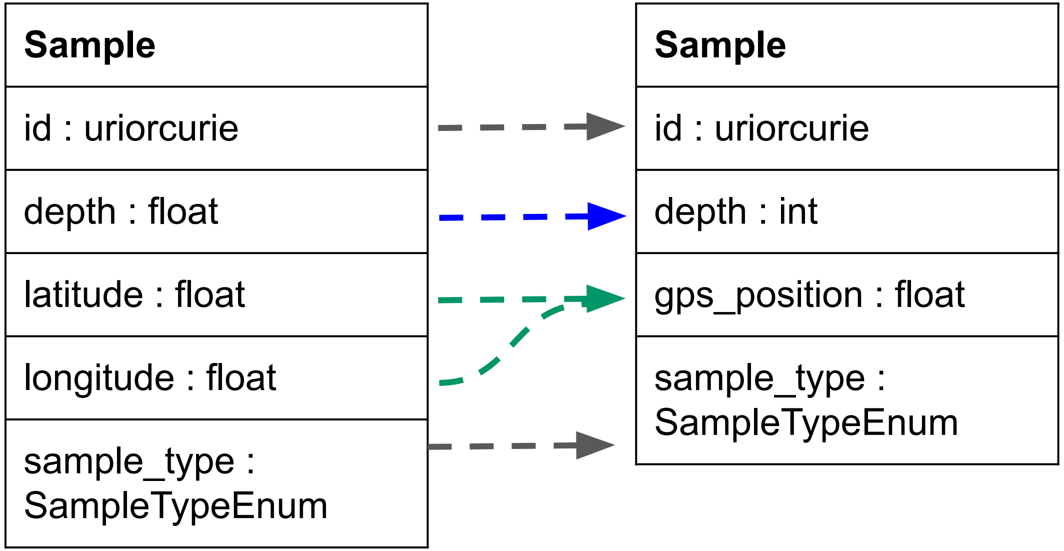
1. deprecated
2. deprecated_element_has_exact_replacement
3. last_updated_on
4. modified_by

```
> poetry run gen-doc --include
src/linkml_tutorial_2024/schema/deprecated.yaml
src/schema/linkml_tutorial_2024/linkml-tutorial-2024.yaml
```

Declarative Transformations (Beta)

```
> poetry add linkml-map
```

https://linkml.io/linkml-map/#examples/Tutorial/



```
class_derivations:
   Sample:
      populated_from: Sample
      slot_derivations:
         gps_position:
            expr: "{latitude} + ' ' + {longitude}"
```

```
> poetry run map-data -T
transformation.yaml -s
linkml-tutoria-2024.yaml
src/data/examples/Sample-001.yaml
```

LinkML & LLMs

- Use free versions of LLM tools (e.g. Gemini, ChatGPT, etc.)
- Don’t be afraid to ask direct questions - LLMs are like teenagers, they “know” everything but they need some redirection to give the answers you’re looking for.
- LLMs can generate LinkML schemas.
- LLMs love to hallucinate ontology ids.
- LLMs make syntax mistakes sometimes.

Use LLMs as a tool to help you bootstrap your knowledge of schema development with LinkML.

Checking in...Rest Stops Along the Way

- **We hope you follow along with us as we build up our project!**
- If you end up lost, each section of the tutorial has a corresponding tag in the linkml-tutorial-2024 repository (github.com/linkml/linkml-tutorial-2024)
- You can clone the linkml-tutorial-2024 repository and checkout the appropriate tag.

Step 0 - basic project creation

Step 1 - modeling

Step 2 - linting

Step 3 - documentation

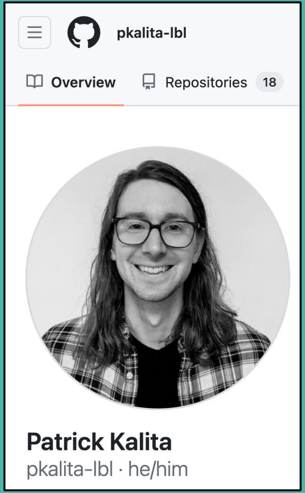
Step 4 - code generation

Step 5 - validation

Feeling lost?

```
> git checkout step_1_modeling
```

Section 2: Linting



Patrick Kalita
pkalita-lbl · he/him

Patrick Kalita
https://github.com/pkalita-lbl

bit.ly/LinkML-2024

Motivating Example

schema.yaml

```
id: https://example.org/my-schema
name: my-schema

classes:
  latitude longitude:
    slots:
      ...
```

Motivating Example

schema.yaml $\xrightarrow{\text{Generate Python}}$ schema.py

```yaml
id: https://example.org/my-schema
name: my-schema

classes:
  latitude longitude:
    slots:
      ...
```

```python
...

class latitude longitude:
    pass

...
```

Motivating Example

schema.yaml → Generate Python → schema.py

```yaml
id: https://example.org/my-schema
name: my-schema

classes:
  latitude longitude:
    slots:
      ...
```

```python
...

class latitu      ongitude:
    pass

...
```

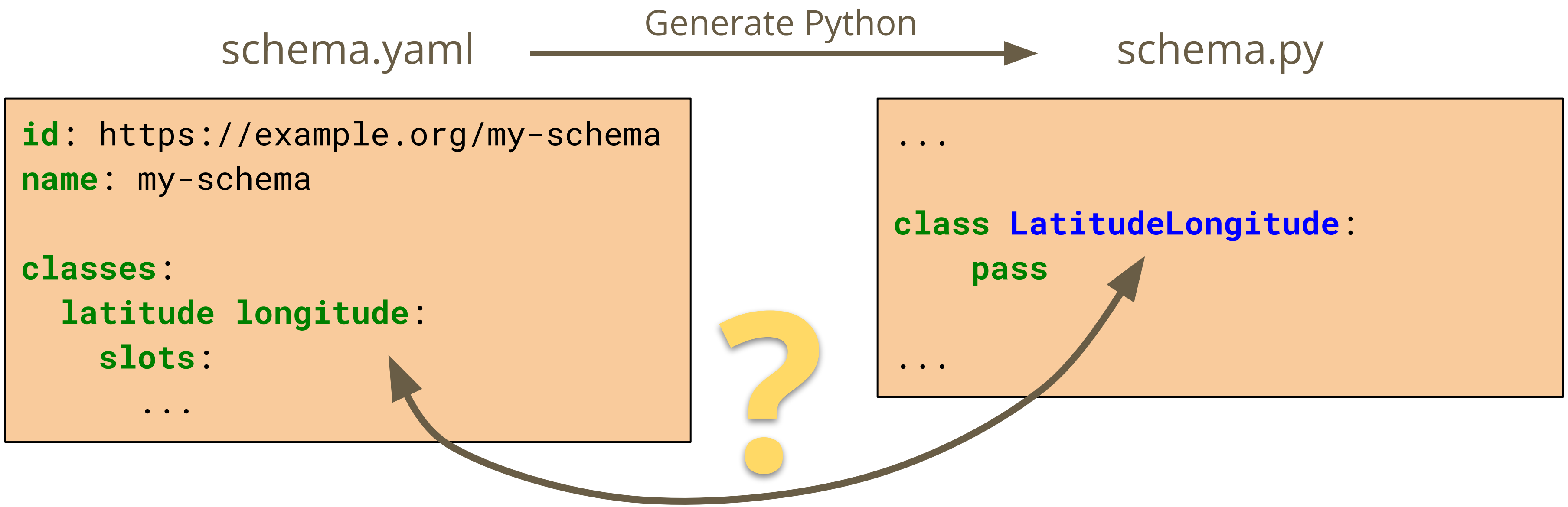
Motivating Example

schema.yaml —Generate Python→ schema.py

```yaml
id: https://example.org/my-schema
name: my-schema

classes:
  latitude longitude:
    slots:
      ...
```

```python
...

class LatitudeLongitude:
    pass

...
```

Motivating Example



```yaml
id: https://example.org/my-schema
name: my-schema

classes:
  latitude longitude:
    slots:
      ...
```

```python
...

class LatitudeLongitude:
    pass

...
```

Motivating Example

schema.yaml $\xrightarrow{\text{Generate Python}}$ schema.py

```yaml
id: https://example.org/my-schema
name: my-schema

classes:
  LatitudeLongitude:
    slots:
      ...
```

```python
...

class LatitudeLongitude:
    pass

...
```

LinkML Linter

The Linter is a rule-based, configurable command line utility to help enforce best practices and identify suspicious patterns in your schema.

CLI:

```
> linkml-lint [--config myconfig.yaml] [schema file or directory]
```

Documentation: https://linkml.io/linkml/schemas/linter.html

Lint Our Schema

Run the Linter on our schema using the default configuration

```
> poetry run linkml-lint src/linkml_tutorial_2024/schema
```

Lint Our Schema

Run the linter on our schema using the default configuration

```
> poetry run linkml-lint src/linkml_tutorial_2024/schema
```

```
warning  Class 'AirSample' does not have recommended slot 'description'  (recommended)
warning  Class 'SoilSample' does not have recommended slot 'description'  (recommended)
warning  Schema maps prefix 'biolink' to namespace 'https://w3id.org/biolink/' instead of namespace 'https://w3id.org/biolink/vocab/'  (canonical_prefixes)
warning  Schema maps prefix 'example' to namespace 'https://example.org/' instead of namespace 'http://www.example.org/rdf#'  (canonical_prefixes)
warning  Schema maps prefix 'SIO' to namespace 'http://semanticscience.org/resource/' instead of namespace 'http://identifiers.org/sio/'  (canonical_prefixes)
```

Lint Our Schema

Run the linter on our schema using the default configuration

```
> poetry run linkml-lint src/linkml_tutorial_2024/schema
```

```
warning  Class 'AirSample' does not have recommended slot 'description'  (recommended)
warning  Class 'SoilSample' does not have recommended slot 'description'  (recommended)
warning  Schema maps prefix 'biolink' to namespace 'https://w3id.org/biolink/' instead of namespace 'https://w3id.org/biolink/vocab/'  (canonical_prefixes)
warning  Schema maps prefix 'example' to namespace 'https://example.org/' instead of namespace 'http://www.example.org/rdf#'  (canonical_prefixes)
warning  Schema maps prefix 'SIO' to namespace 'http://semanticscience.org/resource/' instead of namespace 'http://identifiers.org/sio/'  (canonical_prefixes)
```

Severity (warning or error)

Lint Our Schema

Run the linter on our schema using the default configuration

```
> poetry run linkml-lint src/linkml_tutorial_2024/schema
```

```
warning  Class 'AirSample' does not have recommended slot 'description'  (recommended)
warning  Class 'SoilSample' does not have recommended slot 'description'  (recommended)
warning  Schema maps prefix 'biolink' to namespace 'https://w3id.org/biolink/' instead of namespace 'https://w3id.org/biolink/vocab/'  (canonical_prefixes)
warning  Schema maps prefix 'example' to namespace 'https://example.org/' instead of namespace 'http://www.example.org/rdf#'  (canonical_prefixes)
warning  Schema maps prefix 'SIO' to namespace 'http://semanticscience.org/resource/' instead of namespace 'http://identifiers.org/sio/'  (canonical_prefixes)
```

Rule name (see documentation for details)

Lint Our Schema

Run the linter on our schema using the default configuration

```
> poetry run linkml-lint src/linkml_tutorial_2024/schema
```

```
warning  Class 'AirSample' does not have recommended slot 'description'  (recommended)
warning  Class 'SoilSample' does not have recommended slot 'description'  (recommended)
warning  Schema maps prefix 'biolink' to namespace 'https://w3id.org/biolink/' instead of namespace 'https://w3id.org/biolink/vocab/'  (canonical_prefixes)
warning  Schema maps prefix 'example' to namespace 'https://example.org/' instead of namespace 'http://www.example.org/rdf#'  (canonical_prefixes)
warning  Schema maps prefix 'SIO' to namespace 'http://semanticscience.org/resource/' instead of namespace 'http://identifiers.org/sio/'  (canonical_prefixes)
```

$\underbrace{\hspace{20em}}_{}$

Detailed description

Resolve the Warnings

- Add missing descriptions to the indicated classes

Resolve the Warnings

- Add missing descriptions to the indicated classes
- Change the `canonical_prefixes` rule with a configuration file

```yaml
# .linkmllint.yaml
extends: recommended    # Start with the recommended configuration
rules:                  # Customize rules here
  canonical_prefixes:
    prefixmaps_contexts:
      - obo
```

Resolve the Warnings

- Add missing descriptions to the indicated classes
- Change the `canonical_prefixes` rule with a configuration file

```yaml
# .linkmllint.yaml
extends: recommended    # Start with the recommended configuration
rules:                  # Customize rules here
  canonical_prefixes:
    prefixmaps_contexts:
      - obo
```

```
> poetry run linkml-lint --config .linkmllint.yaml src/linkml_tutorial_2024/schema
```

Resolve the Warnings

- Add missing descriptions to the indicated classes
- Change the `canonical_prefixes` rule with a configuration file

```yaml
# .linkmllint.yaml
extends: recommended    # Start with the recommended configuration
rules:                  # Customize rules here
  canonical_prefixes:
    prefixmaps_contexts:
      - obo
```

```
> poetry run linkml-lint --config .linkmllint.yaml src/linkml_tutorial_2024/schema
```

Resolve the Warnings

- Add missing descriptions to the indicated classes
- Change the `canonical_prefixes` rule with a configuration file

```yaml
# .linkmllint.yaml
extends: recommended    # Start with the recommended configuration
rules:                  # Customize rules here
  canonical_prefixes:
    prefixmaps_contexts:
      - obo
```

```
> poetry run linkml-lint --config .linkmllint.yaml src/linkml_tutorial_2024/schema
```

$\underbrace{\texttt{--config .linkmllint.yaml}}$

Not required when using the conventional `.linkmllint.yaml` filename

Resolve the Warnings

- Add missing descriptions to the indicated classes
- Change the `canonical_prefixes` rule with a configuration file

```yaml
# .linkmllint.yaml
extends: recommended    # Start with the recommended configuration
rules:                  # Customize rules here
  canonical_prefixes:
    prefixmaps_contexts:
      - obo
```

```
> poetry run linkml-lint src/linkml_tutorial_2024/schema
```

Let’s be More Strict

- Enabling the `tree_root_class` rule in the configuration file

```yaml
# .linkmllint.yaml
extends: recommended    # Start with the recommended configuration
rules:                  # Customize rules here
  canonical_prefixes:
    prefixmaps_contexts:
      - obo
  tree_root_class:
    level: error
    root_class_name: SampleCollection
    validate_existing_class_name: true
```

Let’s be More Strict

- Enabling the `tree_root_class` rule in the configuration file
- Add the `--validate` command line flag

```
> poetry run linkml-lint --validate src/linkml_tutorial_2024/schema
```

Checking in...Rest Stops Along the Way

- **We hope you follow along with us as we build up our project!**
- If you end up lost, each section of the tutorial has a corresponding tag in the linkml-tutorial-2024 repository (github.com/linkml/linkml-tutorial-2024)
- You can clone the linkml-tutorial-2024 repository and checkout the appropriate tag.

Step 0 - basic project creation

Step 1 - modeling

Step 2 - linting

Step 3 - documentation

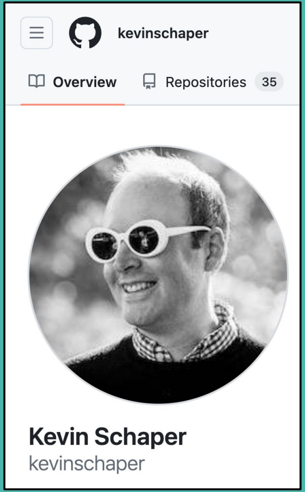
Step 4 - code generation

Step 5 - validation

Feeling lost?

```
> git checkout step_2_linting
```

Section 3: Generating documentation


kevinschaper

Overview Repositories 35

Kevin Schaper
kevinschaper

Kevin Schaper
https://github.com/kevinschaper

bit.ly/LinkML-2024

Documentation

- Critical and undervalued (until it's needed) resource for any schema.
- Keeping documentation close to the "code" is important.
  - onboarding new members is faster
  - small, iterative changes are easier
- Not all users navigate YAML easily.
- Socializing a model is almost as important as writing one.

LinkML can generate nice documentation websites while still allowing developers to keep documentation for specific components close to the code.

Generate and deploy documentation

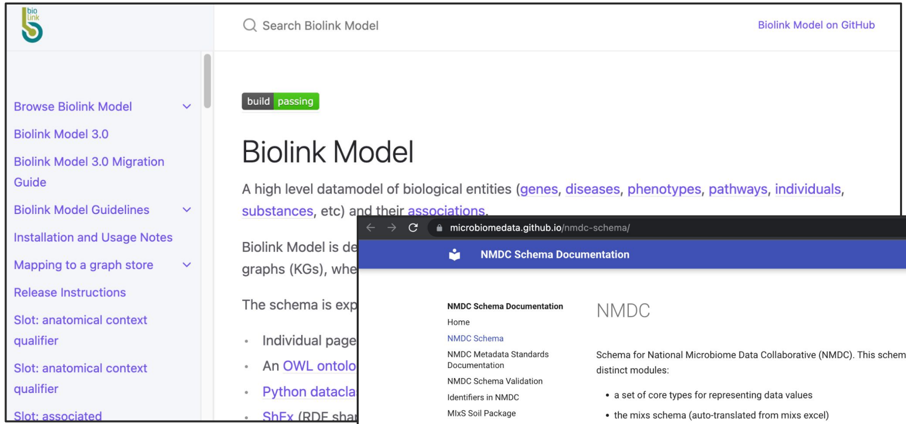
- LinkML has an automatic Markdown and HTML documentation generator
- There are GitHub Actions workflows that are available for you to use when you create a LinkML cookiecutter project
    - **`deploy-docs.yaml`**: automatically build and publish mkdocs style web documentation pages to GitHub Pages, accessible on the **`gh-pages`** branch of your project repo

- Running the following commands will allow you to preview the HTML documentation locally before deploying it to GitHub Pages

```
> make testdoc
```

```
http://127.0.0.1:8000/my-project/
```

Documentation customization


Search Biolink Model

Biolink Model on GitHub

- Browse Biolink Model
- Biolink Model 3.0
- Biolink Model 3.0 Migration Guide
- Biolink Model Guidelines
- Installation and Usage Notes
- Mapping to a graph store
- Release Instructions
- Slot: anatomical context qualifier
- Slot: anatomical context qualifier
- Slot: associated

build passing

Biolink Model

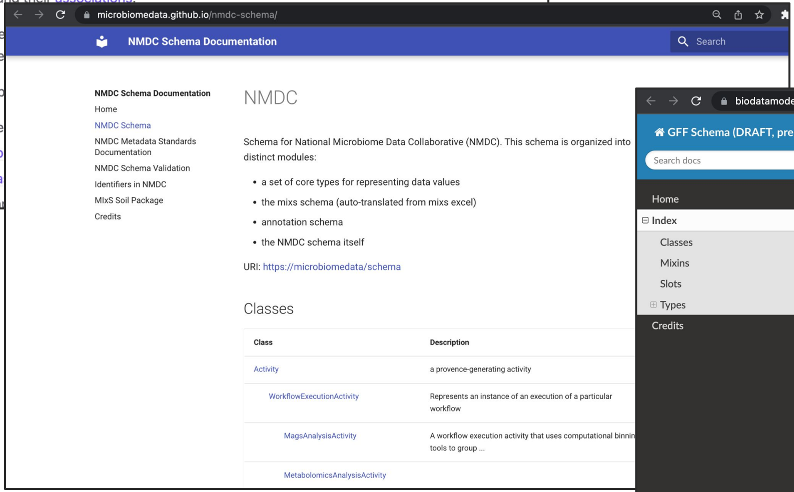
A high level datamodel of biological entities (genes, diseases, phenotypes, pathways, individuals, substances, etc) and their associations.

Biolink Model is de

graphs (KGs), whe

The schema is exp

- Individual page
- An OWL ontolo
- Python datacla
- ShEx (RDF shar


microbiomedata.github.io/nmdc-schema/

NMDC Schema Documentation

Search

- NMDC Schema Documentation
- Home
- NMDC Schema
- NMDC Metadata Standards Documentation
- NMDC Schema Validation
- Identifiers in NMDC
- MIxS Soil Package
- Credits

NMDC

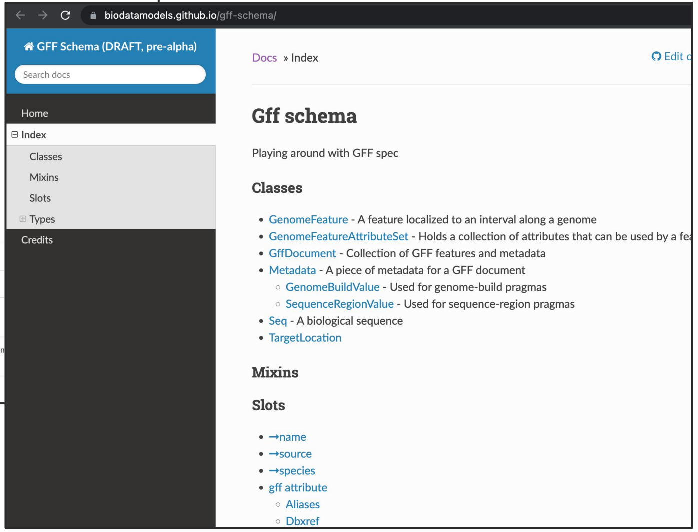
Schema for National Microbiome Data Collaborative (NMDC). This schema is organized into distinct modules:

- a set of core types for representing data values
- the mixs schema (auto-translated from mixs excel)
- annotation schema
- the NMDC schema itself

URI: https://microbiomedata/schema

Classes

Class	Description
Activity	a provence-generating activity
WorkflowExecutionActivity	Represents an instance of an execution of a particular workflow
MagsAnalysisActivity	A workflow execution activity that uses computational binnin tools to group ...
MetabolomicsAnalysisActivity	


biodatamodels.github.io/gff-schema/

GFF Schema (DRAFT, pre-alpha)

Search docs

- Home
- Index
  - Classes
  - Mixins
  - Slots
  - Types
- Credits

Docs » Index

Edit

Gff schema

Playing around with GFF spec

Classes

- GenomeFeature - A feature localized to an interval along a genome
- GenomeFeatureAttributeSet - Holds a collection of attributes that can be used by a fe
- GffDocument - Collection of GFF features and metadata
- Metadata - A piece of metadata for a GFF document
  - GenomeBuildValue - Used for genome-build pragmas
  - SequenceRegionValue - Used for sequence-region pragmas
- Seq - A biological sequence
- TargetLocation

Mixins

Slots

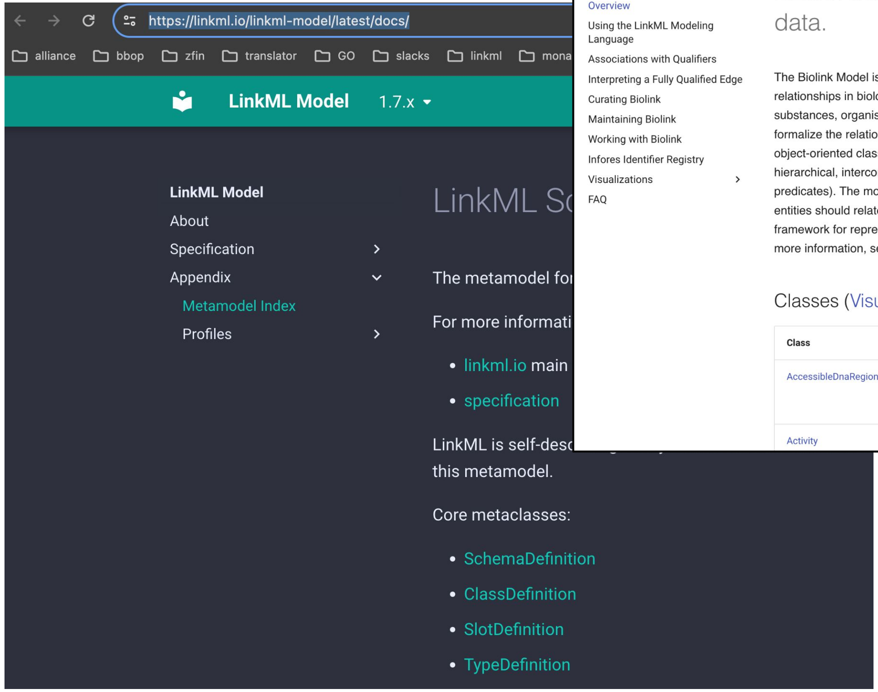
- →name
- →source
- →species
- gff attribute
  - Aliases
  - Dbxref

Just The Docs:
https://biolink.github.io/biolink-model/

Mkdocs Material:
https://microbiomedata.github.io/nmdc-schema/

ReadTheDocs:
https://github.com/biodatamodels/gff-schema

Documentation customization (themes)



https://linkml.io/linkml-model/latest/docs/

alliance bbop zfin translator GO slacks linkml mona

LinkML Model 1.7.x

- LinkML Model
- About
- Specification
- Appendix
  - Metamodel Index
  - Profiles

LinkML Sc

The metamodel fo

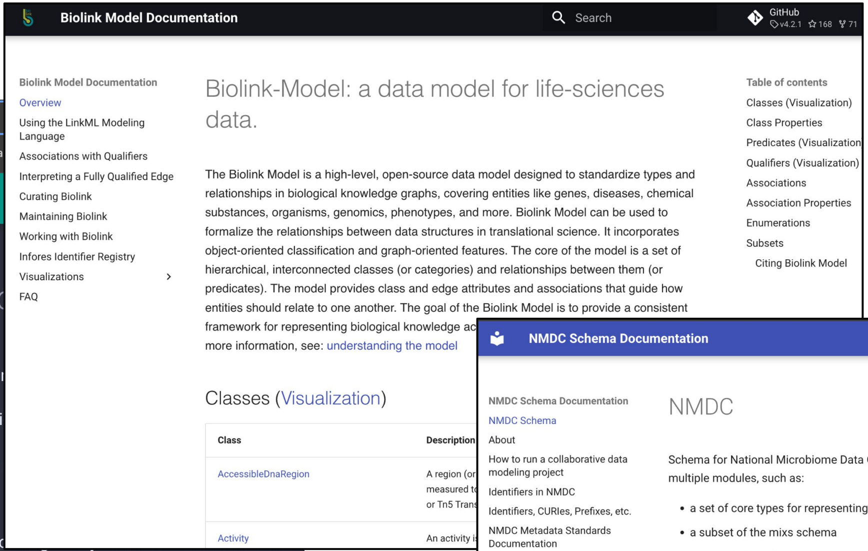
For more informati

- linkml.io main
- specification

LinkML is self-desc
this metamodel.

Core metaclasses:

- SchemaDefinition
- ClassDefinition
- SlotDefinition
- TypeDefinition



Biolink Model Documentation Search GitHub v4.2.1 168 71

- Biolink Model Documentation
- Overview
- Using the LinkML Modeling Language
- Associations with Qualifiers
- Interpreting a Fully Qualified Edge
- Curating Biolink
- Maintaining Biolink
- Working with Biolink
- Infores Identifier Registry
- Visualizations
- FAQ

Biolink-Model: a data model for life-sciences data.

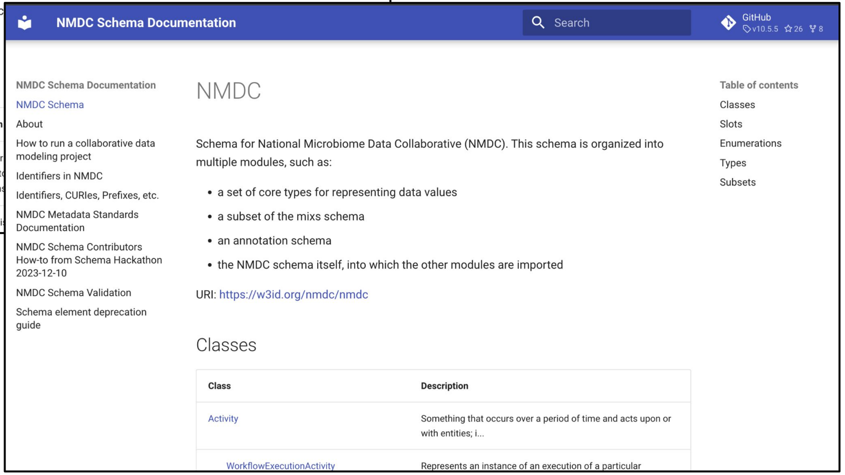
The Biolink Model is a high-level, open-source data model designed to standardize types and relationships in biological knowledge graphs, covering entities like genes, diseases, chemical substances, organisms, genomics, phenotypes, and more. Biolink Model can be used to formalize the relationships between data structures in translational science. It incorporates object-oriented classification and graph-oriented features. The core of the model is a set of hierarchical, interconnected classes (or categories) and relationships between them (or predicates). The model provides class and edge attributes and associations that guide how entities should relate to one another. The goal of the Biolink Model is to provide a consistent framework for representing biological knowledge ac
more information, see: understanding the model

Classes (Visualization)

Class	Description
AccessibleDnaRegion	A region (or measured t or Tn5 Trans
Activity	An activity i

Table of contents

- Classes (Visualization)
- Class Properties
- Predicates (Visualization
- Qualifiers (Visualization)
- Associations
- Association Properties
- Enumerations
- Subsets
  - Citing Biolink Model



NMDC Schema Documentation Search GitHub v10.5.5 26 8

- NMDC Schema Documentation
- NMDC Schema
- About
- How to run a collaborative data modeling project
- Identifiers in NMDC
- Identifiers, CURIes, Prefixes, etc.
- NMDC Metadata Standards Documentation
- NMDC Schema Contributors How-to from Schema Hackathon 2023-12-10
- NMDC Schema Validation
- Schema element deprecation guide

NMDC

Schema for National Microbiome Data Collaborative (NMDC). This schema is organized into multiple modules, such as:

- a set of core types for representing data values
- a subset of the mixs schema
- an annotation schema
- the NMDC schema itself, into which the other modules are imported

URI: https://w3id.org/nmdc/nmdc

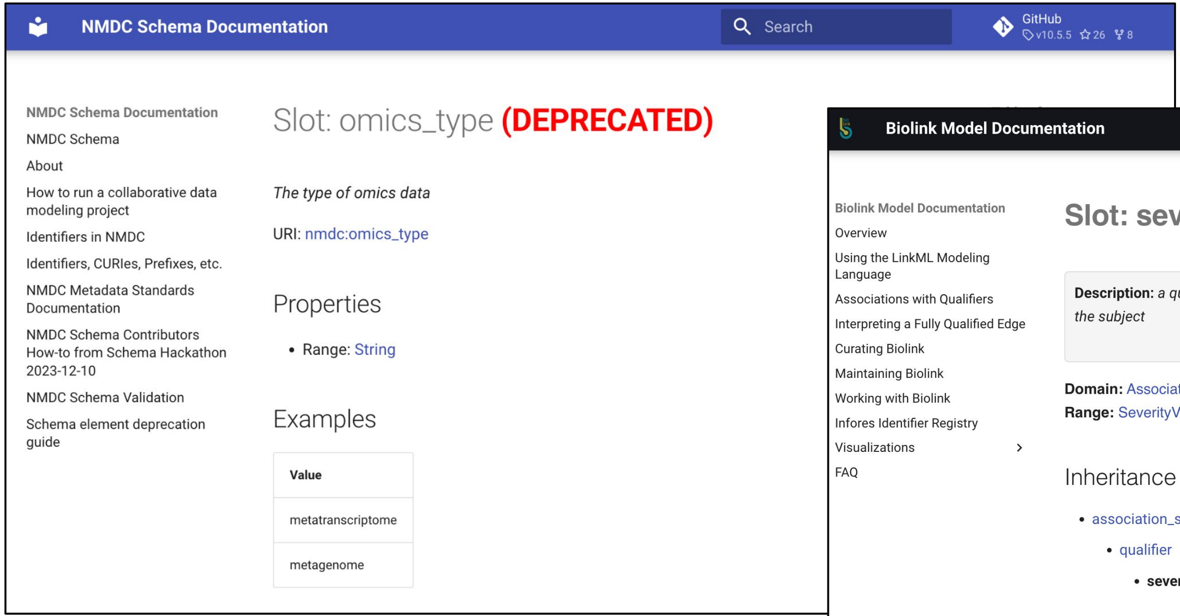
Classes

Class	Description
Activity	Something that occurs over a period of time and acts upon or with entities; i...
WorkflowExecutionActivity	Represents an instance of an execution of a particular

Table of contents

- Classes
- Slots
- Enumerations
- Types
- Subsets

Documentation customization (templates)



NMDC Schema Documentation

Search

GitHub v10.5.5 26 8

NMDC Schema Documentation
NMDC Schema
About
How to run a collaborative data modeling project
Identifiers in NMDC
Identifiers, CURIes, Prefixes, etc.
NMDC Metadata Standards Documentation
NMDC Schema Contributors How-to from Schema Hackathon 2023-12-10
NMDC Schema Validation
Schema element deprecation guide

Slot: omics_type **(DEPRECATED)**

The type of omics data

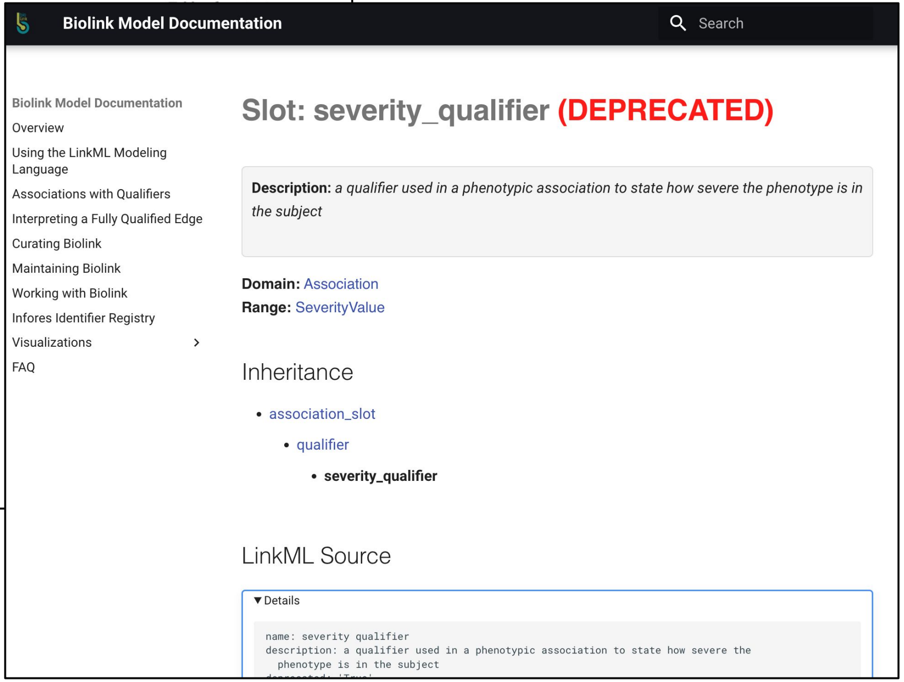
URI: nmdc:omics_type

Properties

- Range: String

Examples

Value
metatranscriptome
metagenome



Biolink Model Documentation

Search

Biolink Model Documentation
Overview
Using the LinkML Modeling Language
Associations with Qualifiers
Interpreting a Fully Qualified Edge
Curating Biolink
Maintaining Biolink
Working with Biolink
Infores Identifier Registry
Visualizations
FAQ

Slot: severity_qualifier (DEPRECATED)

Description: *a qualifier used in a phenotypic association to state how severe the phenotype is in the subject*

Domain: Association
Range: SeverityValue

Inheritance

- association_slot
  - qualifier
    - **severity_qualifier**

LinkML Source

▼Details

```
name: severity qualifier
description: a qualifier used in a phenotypic association to state how severe the
  phenotype is in the subject
```

Assignment: Customizing Automatic Documentation

Problem statement: Researcher 1 askes her postdoc to show deprecated elements are shown documentations pages.

Modify the existing jinja template for class and slot pages, to include “deprecated” metadata when an element is deprecated.

Documentation customization (jinja templates)

- Jinja is a templating system that allows pages to be displayed dynamically via makedocs.
- Customizing elements on the page requires some understanding of LinkML's metamodel (which is also a LinkML model)

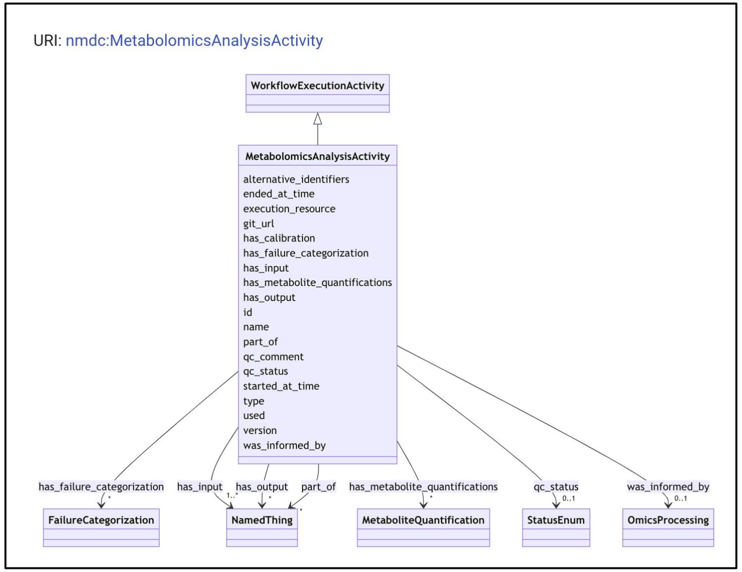
```
edit: src/doc-templates/class.md.jinja2
```

```
> make doctest
```

```
https://linkml.io/linkml-model/latest/docs/
```

Documentation: schema diagrams

- By default, each generated LinkML class documentation page, has a narrowly scoped schema diagram for the class included.

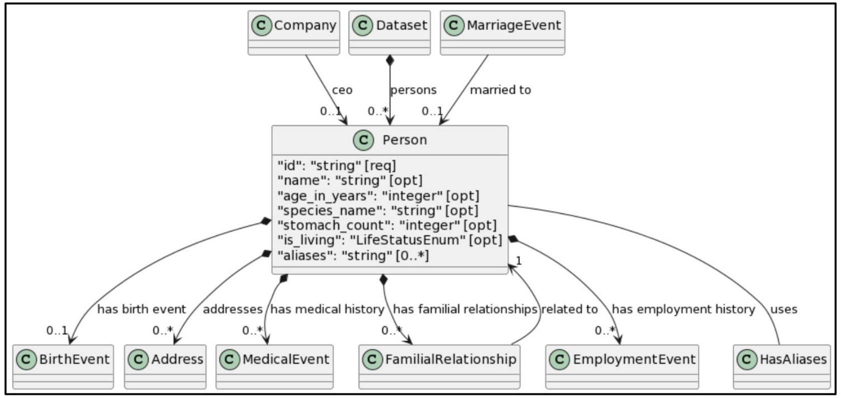


URI: nmdc:MetabolomicsAnalysisActivity

- parent classes
- child classes
- slot names for the class of interest
- cardinality and identification of the slot that relates parent to child

Documentation: schema diagrams

- Customization is possible, and more kinds of diagrams are possible including: PlantUML diagrams and ER-diagrams



PlantUML rendering

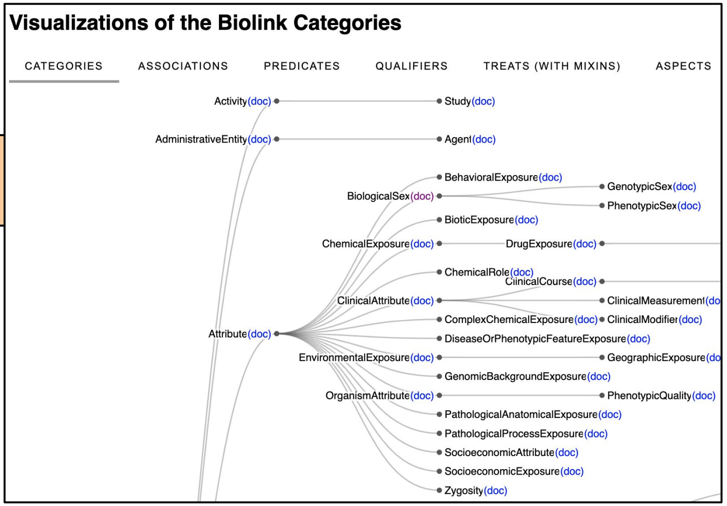
- Similar features to the default mermaid diagrams
- Includes slot types as well as slot names.
- More feature-rich UML implementation

Documentation: full model diagrams

```
> poetry run gen-plantuml > out.puml
```

https://www.plantuml.com/plantuml/ (test viewer)

```
> poetry run gen-erdiagram > out.er
```



- Useful for looking at more than a single family of classes at once.
- Can be overwhelming for larger models
- Some models develop custom javascript
- Consult the LinkML registry for different implementations

Documentation: Auto-deployment to GH pages

*Note: our tutorial site has been building and deploying a GH pages site on every push to the “main” branch by default.

https://linkml.io/linkml-tutorial-2024/

Gotcha:

Make sure you configure GitHub project to use the gh-pages branch to display the doc.

Checking in...Rest Stops Along the Way

- **We hope you follow along with us as we build up our project!**
- If you end up lost, each section of the tutorial has a corresponding tag in the linkml-tutorial-2024 repository (github.com/linkml/linkml-tutorial-2024)
- You can clone the linkml-tutorial-2024 repository and checkout the appropriate tag.

Section 0 - basic project creation

Section 1 - modeling

Section 2 - linting

Section 3 - documentation

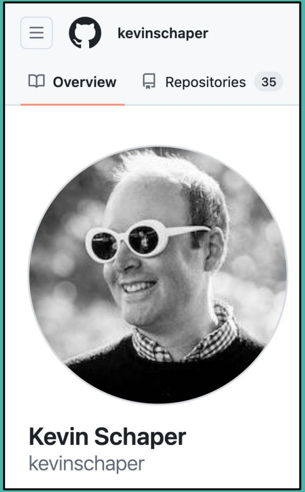
Section 4 - code generation

Section 5 - validation

Feeling lost?

```
> git checkout step_3_documentation
```

Section 4: Code Generation


kevinschaper
Overview
Repositories 35
Kevin Schaper
kevinschaper

Kevin Schaper
https://github.com/kevinschaper

bit.ly/LinkML-2024

Why distribute your model in many serializations?

Case Study: Alliance of Genome Resources and National Microbiome Data Collaborative use of LinkML

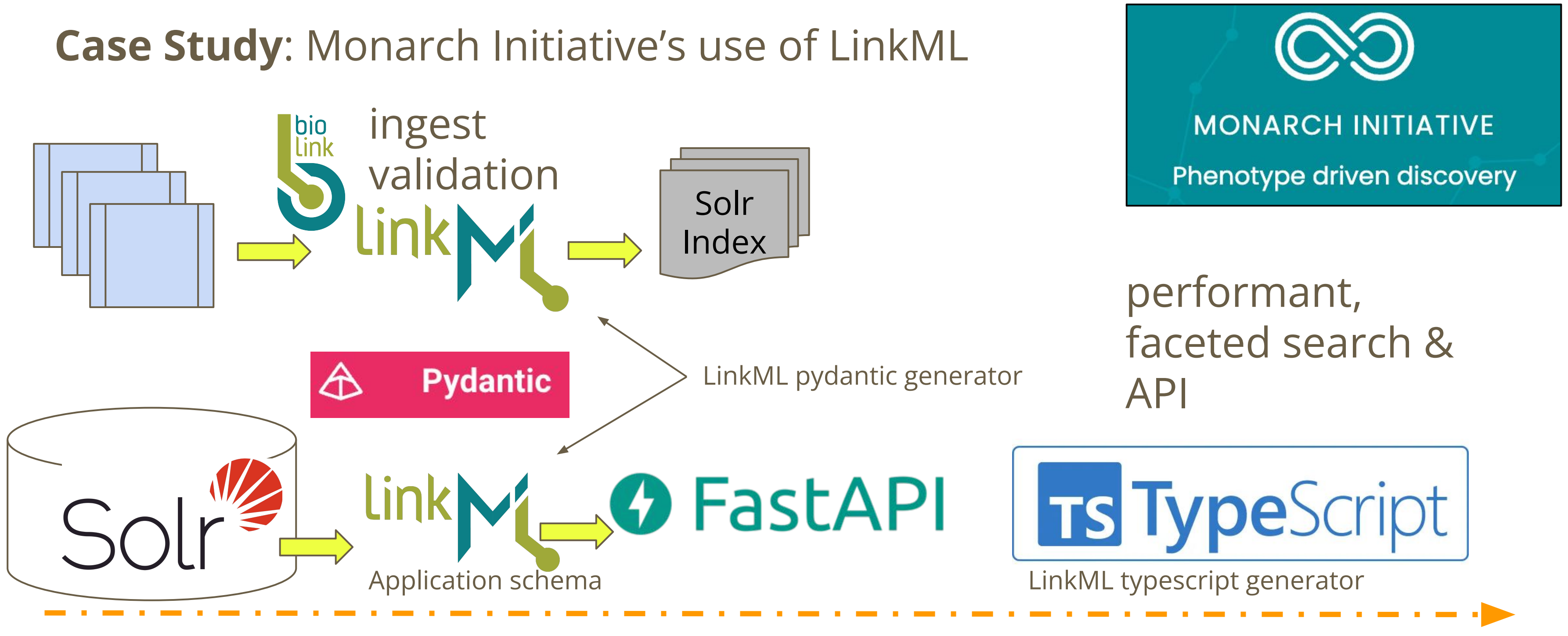
Serialize LinkML Model as JSONSchema and/or PostgreSQL DDL in order to constrain data submission

Data Ingest

Display

Why distribute your model in many serializations?

Case Study: Monarch Initiative's use of LinkML



Data Ingest

Display

Assignment: Generate and push pydantic models to PyPi

Problem statement: Working with a group of bioinformaticians, you're given the requirement that they would like to write tools to display the sample data on a webpage. They use python, they want a code-based representation of the model.

Generate python and/or pydantic data models from your LinkML YAML model and distribute them on pypi for use in python applications

Step 5: Run a generator directly

```
> poetry run gen-pydantic
src/linkml_tutorial_2024/schema/linkml_tutorial_2024.yaml >
linkml_tutorial_2024_pydantic_model.py
```

Optional experiment, generate a typescript model:

```
>  poetry run gen-typescript
src/linkml_tutorial_2024/schema/linkml_tutorial_2024.yaml >
linkml_tutorial_2024_typescript_model.ts
```

Step 5: gen-project

```
> make gen-project
```

- We can configure this builder on the command line (see Makefile).
  - The LinkML cookiecutter generates a Makefile target to run gen-project out of the box with some default serializations.
- Configured with a config.yaml file.

```
edit: config.yaml
```

Step 5: gen-project

Configuring with a config.yaml file.

```
edit: config.yaml
```

config.yaml is a simple YAML dictionary

- Pass custom generator parameters via **generator_args**
- "includes" are the serializations to **include** in gen-project
- "excludes" are the serializations to **exclude** in gen-project
- **merge_imports** controls whether you want to include imported LinkML schemas in the resulting serialization (default here is "true")

```
generator_args:
  jsonschema:

includes:
  - jsonschema
  - python
excludes:
  - graphql
  - owl
  - shex
  - shacl
  - sqlschema
  - protobuf
  - prefixmap
mergeimports: true
```

Step 5: Generate and deploy model serializations

- make is a wrapper around poetry
- gen-project is a grouping of many popular model serialization generators (JSONSchema, python dataclasses, pydantic classes, doc, OWL, etc.)
- we can always run individual generators via poetry

```
> make gen-project
```

Deployment to PyPI

- LinkML cookiecutter does most of the work for you.
  - Creates a project structure that automatically makes python serializations available in the correct directory to be bundled into a pypi package
  - Creates a GitHub action that will run “on release” and push your “main” branch to PyPI
- Two additional steps:
  - Create a pending publisher: https://docs.pypi.org/trusted-publishers/creating-a-project-through-oidc/#github-actions
  - Make the GH action use that pending publisher (official docs have more information, this is the TL;DR recipe that can provide a nice summary of the effort): https://pgjones.dev/blog/trusted-plublishing-2023/ (on first use, the pending publisher becomes a trusted publisher)

Checking in...Rest Stops Along the Way

- **We hope you follow along with us as we build up our project!**
- If you end up lost, each section of the tutorial has a corresponding tag in the linkml-tutorial-2024 repository (github.com/linkml/linkml-tutorial-2024)
- You can clone the linkml-tutorial-2024 repository and checkout the appropriate tag.

Step 0 - basic project creation

Step 1 - modeling

Step 2 - linting

Step 3 - documentation

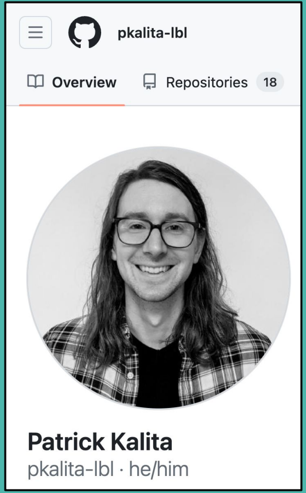
Step 4 - code generation

Step 5 - validation

Feeling lost?

```
> git checkout step_4_code_generation
```

Section 5: Validating Data


pkalita-lbl

Overview Repositories 18

Patrick Kalita

pkalita-lbl · he/him

Patrick Kalita
https://github.com/pkalita-lbl

bit.ly/LinkML-2024

Motivating Scenario

- A soil researcher wants to submit data to us
- We want to verify that their data conform to our schema before accepting it

Validating an Example Data File

The linkml-validate command is a configurable command line utility for validating data instances against a schema.

CLI:

```
> linkml-validate --schema [schema file] [data source...]
```

Documentation: https://linkml.io/linkml/data/validating-data.html

Validating an Example Data File

```yaml
# src/data/examples/SoilSample.yaml
id: "soilsample:002"
latitude: 40.7128
longitude: -74.0060
species:
  - NCBITaxon:1423
sample_biome: desert
depth: 15
sample_type: SoilSample
```

Validating an Example Data File

```
> poetry run linkml-validate \
    --schema src/linkml_tutorial_2024/schema/linkml_tutorial_2024.yaml \
    --target-class SoilSample \
    src/data/examples/SoilSample.yaml
```

```
No issues found
```

Validating an Example Data File

```yaml
# src/data/examples/SoilSample.yaml
id: "soilsample:002"
latitude: 40.7128
longitude: -74.0060
species:
  - NCBITaxon:1423
sample_biome: backyard
depth: 15
sample_type: SoilSample
```

Not a permissible value in BiomeTypeEnum

Validating an Example Data File

```
> poetry run linkml-validate \
    --schema src/linkml_tutorial_2024/schema/linkml_tutorial_2024.yaml \
    --target-class SoilSample \
    src/data/examples/SoilSample.yaml
```

```
[ERROR] [src/data/examples/SoilSample.yaml/0] 'backyard' is not one of ['forest',
'lake', 'ocean', 'desert', 'air'] in /sample_biome
```

Validating an Example Data File

```
> poetry run linkml-validate \
    --schema src/linkml_tutorial_2024/schema/linkml_tutorial_2024.yaml \
    --target-class SoilSample \
    src/data/examples/SoilSample.yaml
```

```
[ERROR] [src/data/examples/SoilSample.yaml/0] 'backyard' is not one of ['forest',
'lake', 'ocean', 'desert', 'air'] in /sample_biome
```

Severity (info, warning, error, fatal)

Validating an Example Data File

```
> poetry run linkml-validate \
    --schema src/linkml_tutorial_2024/schema/linkml_tutorial_2024.yaml \
    --target-class SoilSample \
    src/data/examples/SoilSample.yaml
```

```
[ERROR] [src/data/examples/SoilSample.yaml/0] 'backyard' is not one of ['forest',
'lake', 'ocean', 'desert', 'air'] in /sample_biome
```

Location of the invalid data instance

Validating an Example Data File

```
> poetry run linkml-validate \
    --schema src/linkml_tutorial_2024/schema/linkml_tutorial_2024.yaml \
    --target-class SoilSample \
    src/data/examples/SoilSample.yaml
```

```
[ERROR] [src/data/examples/SoilSample.yaml/0] 'backyard' is not one of ['forest',
'lake', 'ocean', 'desert', 'air'] in /sample_biome
```

Pointer within the data instance where the issue occurred

Validating an Example Data File

```
> poetry run linkml-validate \
    --schema src/linkml_tutorial_2024/schema/linkml_tutorial_2024.yaml \
    --target-class SoilSample \
    src/data/examples/SoilSample.yaml
```

```
[ERROR] [src/data/examples/SoilSample.yaml/0] 'backyard' is not one of ['forest',
'lake', 'ocean', 'desert', 'air'] in /sample_biome
```

Detailed description of the issue

Validating an Example Data File

```yaml
# src/data/examples/SoilSample.yaml
id: "soilsample:002"
latitude: 40.7128
longitude: -74.0060
species:
  - NCBITaxon:1423
sample_biome: desert
depth: 15
sample_type: SoilSample
---
id: "soilsample:003"
latitude: 123.456
longitude: -0.1278
species:
  - NCBITaxon:287
sample_biome: forest
depth: 30
sample_type: SoilSample
```

Validating an Example Data File

```
> poetry run linkml-validate \
    --schema src/linkml_tutorial_2024/schema/linkml_tutorial_2024.yaml \
    --target-class SoilSample \
    src/data/examples/SoilSample.yaml
```

```
No issues found
```

Validating an Example Data File

```yaml
# src/data/examples/SoilSample.yaml
id: "soilsample:002"
latitude: 40.7128
longitude: -74.0060
species:
  - NCBITaxon:1423
sample_biome: desert
depth: 15
sample_type: SoilSample
---
id: "soilsample:003"
latitude: 123.456
longitude: -0.1278
species:
  - NCBITaxon:287
sample_biome: forest
depth: 30
sample_type: SoilSample
```

🤔

Validating an Example Data File

```yaml
# src/linkml_tutorial_2024/schema/linkml_tutorial_2024.yaml

slots:
...
    latitude:
        description: >-
          Latitude is a geographic coordinate which refers to the angle from
          a point on the Earth's surface to the equatorial plane.
        range: float
        slot_uri: schema:latitude
...
```

Validating an Example Data File

```yaml
# src/linkml_tutorial_2024/schema/linkml_tutorial_2024.yaml

slots:
...
    latitude:
        description: >-
          Latitude is a geographic coordinate which refers to the angle from
          a point on the Earth's surface to the equatorial plane.
        range: float
        minimum_value: -90
        maximum_value: 90
        slot_uri: schema:latitude
...
```

Validating an Example Data File

```
> poetry run linkml-validate \
    --schema src/linkml_tutorial_2024/schema/linkml_tutorial_2024.yaml \
    --target-class SoilSample \
    src/data/examples/SoilSample.yaml
```

```
[ERROR] [src/data/examples/SoilSample.yaml/1] 123.456 is greater than the maximum of
90 in /latitude
```

Validation as a Schema Development Aid

The linkml-run-examples command ensures that all data files in a examples directory pass validation and that all data files in a counter-examples directory do not pass validation.

CLI:

```
> poetry run linkml-run-examples --help
```

Validation as a Schema Development Aid

```
src/
└── data/
    └── examples/
        ├── invalid/
        │   └── SoilSample-latitude-too-high.yaml
        └── valid/
            └── SoilSample.yaml
```

Validation as a Schema Development Aid

```
src/
└── data/
    └── examples/
        ├── invalid/
        │   └── SoilSample-latitude-too-high.yaml
        └── valid/
            └── SoilSample.yaml
```

```
> poetry run linkml-run-examples \
        --output-formats yaml \
        --counter-example-input-directory src/data/examples/invalid \
        --input-directory src/data/examples/valid \
        --output-directory examples/output \
        --schema src/linkml_tutorial_2024/schema/linkml_tutorial_2024.yaml
```

Validation as a Schema Development Aid

```
src/
└── data/
    └── examples/
        ├── invalid/
        │   └── SoilSample-latitude-too-high.yaml
        └── valid/
            └── SoilSample.yaml
```

See also: the `test-examples` target in Makefile.

Going Further with Validation

- Programmatic validation in Python code
- Advanced command line configuration
- See: https://linkml.io/linkml/data/validating-data.html

Checking in...Rest Stops Along the Way

- **We hope you follow along with us as we build up our project!**
- If you end up lost, each section of the tutorial has a corresponding tag in the linkml-tutorial-2024 repository (github.com/linkml/linkml-tutorial-2024)
- You can clone the linkml-tutorial-2024 repository and checkout the appropriate tag.

Step 0 - basic project creation

Step 1 - modeling

Step 2 - linting

Step 3 - documentation

Step 4 - code generation

Step 5 - validation

Feeling lost?

```
> git checkout step_5_validation
```

Questions? Discussion?

Learning more and staying connected

- Our website: https://linkml.io
- GitHub:
  - Issues: https://github.com/linkml/linkml/issues
  - All feature requests, comments, questions are welcome!
  - We 🤎 pull requests!
- Connecting directly
  - Developers currently meet on OBO Workspace slack
    - https://obo-communitygroup.slack.com/archives/C04EU7JL1NF

Join the LinkML community!

- LinkML documentation: https://linkml.io/linkml
- Issue Tracker: https://github.com/linkml/linkml/issues (Feature voting)
- Mailing list: https://groups.google.com/g/linkml-community
- LinkML Community Slack channel: https://obo-communitygroup.slack.com/archives/C04EU7JL1NF
- LinkedIn group: https://www.linkedin.com/groups/14303246/
- Mastodon: https://qoto.org/@linkml

ISMB tutorial survey

Please fill out this survey:
https://docs.google.com/forms/d/e/1FAIpQLSfrjbxbGdE45OFPQgM58JDt59B--gonwjuKw1HkHrR4FLvhhw/viewform