Knowledge Graph Design for Data Sharing, Integration, and Reuse

Pascal Hitzler

Data Semantics Laboratory (DaSe Lab)
Center for Artificial Intelligence and Data Science (CAIDS)
Kansas State University
http://www.pascal-hitzler.de
Data Semantics Lab

- 10 PhD students, 1 Master student, 7 undergrads
- Kansas State University, Manhattan, KS. [http://daselab.org](http://daselab.org)
- Semantic Web Data Management; Neural-Symbolic Integration
Knowledge Graphs

Knowledge Graphs and their schemas are made to enable easier

- data sharing
- data discovery
- data integration
- data reuse
Laura Kelly is an American politician serving as the 48th governor of Kansas since 2019. A member of the Democratic Party, she represented the 18th district in the Kansas Senate from 2005 to 2019. Kelly ran for governor in the 2018 election and defeated the Republican nominee, Kansas Secretary of State Kris Kobach. Wikipedia

Born: January 24, 1950 (age 69 years), New York, NY
Spouse: Ted Daughety
Party: Democratic Party
Office: Governor of Kansas since 2019
Education: Indiana University, Bradley University, Indiana University Bloomington
Children: Kathleen Daughety, Molly Daughety

Indiana University is a multi-campus public university system in the state of Indiana, United States. Indiana University has a combined student body of more than 110,000 students, which includes approximately 46,000 students enrolled at the Indiana University Bloomington campus. Wikipedia

Mascot: Referred to as "The Hoosiers"
Endowment: 1.986 billion USD
Students: 110,436 university-wide
President: Michael McRobbie
Academic staff: 8,733 university-wide
Subsidiaries: Indiana University Bloomington, MORE

Michael Alexander McRobbie AO is an Australian-American computer scientist, educator and academic administrator. He became the eighteenth president of Indiana University on July 1, 2007. Wikipedia

Born: October 11, 1950 (age 69 years), Melbourne, Australia
Spouse: Laurie Burns (m. 2005)
Education: The Australian National University, The University of Queensland
Books: Automated Theorem-proving in Non-classical Logics, Automated Deduction - Cadence
Knowledge Graphs

Laura Kelly

hasEducation

Indiana University

hasPresident

Michael McRobbie

hasEducation

University of Queensland

01/24/1950

hasBirthDate

110,436

hasStudents
A good schema is critical for ease of reuse
Knowledge Graphs and their schemas are made to enable easier

- data sharing
- data discovery
- data integration
- data reuse
知识图谱标准

RDF 1.1 概念和抽象语法
W3C 推荐 2014 年 2 月 25 日

此版本:
http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

最新发布的版本:
http://www.w3.org/TR/rdf11-concepts/

前版本:
http://www.w3.org/TR/2014/PR-rdf11-concepts-20140109/

前推荐:
http://www.w3.org/TR/rdf-concepts

编辑:
Richard Cyganiak, DERI, NUI Galway
David Wood, 3 Round Stones
Markus Lanthaler, Graz University of Technology

OWL 2 万维网本体语言指南（第二版）
W3C 推荐 2012 年 12 月 11 日

此版本:
http://www.w3.org/TR/2012/REC-owl2-primer-20121211/

最新版本（系列 2）:
http://www.w3.org/TR/owl2-primer/

最新推荐:
http://www.w3.org/TR/owl-primer

前版本:
http://www.w3.org/TR/2012/PER-owl2-primer-20121018/

编辑:
Pascal Hitzler, Wright State University
Markus Krötzsch, University of Oxford
Bijan Parsia, University of Manchester
Peter F. Patel-Schneider, Nuance Communications
Sebastian Rudolph, FZI Research Center for Information
RDF in a nutshell

:LauraKelly :hasEducation :IndianaUniversity .
:LauraKelly :hasBirthDate <01/24/1950> .

:IndianaUniversity :hasPresident :MichaelMcRobbie .
:IndianaUniversity :hasStudents <110,436> .

Etc.

Identifiers are URIs.
You call these node-edge-node pieces “(RDF) triples”.
A knowledge graph is a set of RDF triples.
This syntax is called RDF Turtle syntax.
The standard prescribes a serialization in XML.
Linked Data: Volume

Geoindexed Linked Data – courtesy of Krzysztof Janowicz, 2012
http://stko.geog.ucsb.edu/location_linked_data
OWL in a nutshell

Relations between
• Classes (Types)
• Relations (Properties)
• Datatypes

Exact relationships are recorded using a formal logic.
E.g., “Every University has a President”

(\forall x) 
(University(x) \rightarrow 
(exists y) ( hasPresident(x,y) AND President(y) ) )
OWL in a nutshell

Classes: unary predicates (types)
Relations: binary predicates (properties)

Logical AND, OR, NEGATION, IMPLICATION
Some restricted use of quantifiers

In particular: You can specify
• subClass relationships ("Mammal" is subClass of "Animal")
• subProperty relationships ("hasMother" subProperty of "hasParent")
• Domains and ranges of properties.

In team modeling, most members don’t have to worry about these details. We heavily use schema diagrams to facilitate team modeling.
Enslaved: Peoples of the Historic Slave Trade

Building a Linked Open Data Platform for the study and exploration of the historical slave trade.
This is not a good Knowledge Graph!

Laura Kelly

01/24/1950

Indiana University

Michael McRobbie

University of Queensland

hasEducation

hasBirthDate

hasPresident

hasStudents

hasEducation

hasEducation

110,436
What makes a good data model?

• Structure resonates with both
  – human expert conceptualizations
  – data and use case requirements
• Generally low maintenance cost
  – Sustainable: robust for future use and re-use
  – Extendable without high management costs

• Ease of use with software and tools
• Machine processable (standards)

• Meets technical, legal, societal requirements
• Stakeholder buy-in
Some of our research

Lead Question:

How to lower knowledge graph management cost while meeting requirements.

Principles:

Our design and development process

• bridges interdisciplinary barriers,
• produces artefacts which resonate with human expert understanding,
• is fully compatible with leading standards,
• is made to save on development and management costs.
Knowledge Graph Schema Modeling

Note: “Knowledge Graph Schema” is a newer term for “Ontology”
Premise

Many ontologies are hard to understand and to re-use.

Some reasons:
• Poor (ad-hoc) modeling.
• Large, monolithic ontologies.
• Different use-case requirements on granularity (some parts too fine-grained, others too coarse).
• Different requirements on data representation for parts of the ontology (e.g., how spatial information is encoded).
Approach: Two main components

1. Modules
   - Rather than thinking of an ontology primarily as an enhanced taxonomy, we think of it as a set of interrelated (and possibly overlapping) modules.
   - Each module is essentially a part of an ontology representing a complex concept in a way which “makes sense” for a human expert. E.g., “oceanographic cruise”.

2. Use of ontology design patterns (ODPs)
   - An ODP is a solution template for a recurring ontology modeling problem.
   - ODPs are instantiated (and modified) to become modules. E.g., a general “Trajectory” ODP may be used as a template to create an “ocean science cruise trajectory” module.
Modeling Teamwork

The modeling team ideally has:
• domain experts
• data experts
• ontology engineers

Divide and Conquer
• First decide on the set of modules to be modeled, then draft modules one at a time.

Joint modeling
• Work mainly through schema diagrams and natural language with the domain and data experts.
• Ontology engineers spell out model details between meetings, and cycle back to the experts for feedback.
Modeling process – steps

1. Define **use case** or scope of use cases
2. Make **competency questions** while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify **key notions** from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at **modules**. Develop the remaining modules from scratch.
5. Add **axioms** for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly **improve** them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create **OWL** files.
A Few Pattern Examples
Joining patterns

Generic AgentRole pattern

Joined:

Generic NameStub pattern

Joined:
Patterns as templates

Joined AgentRole and NameStub patterns:

Used as a template for a concrete modeling problem:
Quantities and Units

Borrowed from the QUDT ontology
Provenance

Borrowed from PROV-O
The Stub Metapattern

Bottom: The CookingEquipmentStub derived from it.
Recipes Example

Recipes Example
Pascal Hitzler, Adila Krisnadhi
A Tutorial on Modular Ontology Modeling with Ontology Design Patterns: The Cooking Recipes Ontology.
Technical Report, DaSe Lab, Department of Computer Science and Engineering, Wright State University, Dayton, OH, August 2018. 22 pages

http://daselab.cs.wright.edu/pub2/mom-recipes-example.pdf
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
Design an ontology which can be used as part of a “recipe discovery” website. The ontology shall be set up such that content from existing recipe websites can in principle be mapped to it (i.e., the ontology gets populated with data from the recipe websites). On the discovery website, detailed graph-queries (using the ontology) shall produce links to recipes from different recipe websites as results. The ontology should be extendable towards incorporation of additional external data, e.g., nutritional information about ingredients or detailed information about cooking equipment.
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
Competency Questions

- From available data and from application use cases, devise competency questions, i.e. questions which should be convertible into queries, which in turn should be answerable using the data.

Gluten-free low-calorie desserts.
How do I make a low-carb pot roast?
How do I make a Chili without beans?
Sweet breakfast under 100 calories.
Breakfast dishes which can be prepared quickly with 2 potatoes, an egg, and some our.
How do I prepare Chicken thighs in a slow cooker?
A simple recipe with pork shoulder and spring onions.
A side prepared using Brussels sprouts, bacon, and chestnuts.
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.

3. **Identify key notions from the data and the use case** and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.

4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.

5. Add axioms for each module, informed by the pattern axioms.

6. Put the modules together and add axioms which involve several modules.

7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.

8. Create OWL files.
Key notions

• Use the competency questions.
• Possibly also query domain experts as to the main notions for the application domain.
• E.g. for the recipes scenario, these would include
  – Recipe
  – Food
  – Time
  – Equipment
  – Classification of food (e.g., as a side)
  – Difficulty level
  – Nutritional information
  – Provenance
Key notions

- Then prioritize which notions to model first. In this case, e.g.
  - recipe
  - food
  - equipment
  - classification
  - difficulty level
  - time
  - nutritional information
  - provenance
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.

3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.

4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.

5. Add axioms for each module, informed by the pattern axioms.

6. Put the modules together and add axioms which involve several modules.

7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.

8. Create OWL files.
Identifying suitable patterns

• Understand the nature of the things you are modeling.

Food: A concrete piece of food? An abstract quantity of food?
Equipment: Do we want a complex model at this stage? No. Stub
Classification: Do we want a complex model at this stage? No. Stub
Difficulty level: Do we want a complex model at this stage? No. Stub
Time: Probably already incorporated in plan?
Nutritional information: model along some existing standard?
Provenance: just that!
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
A plan, a description.

Pattern: Plan as a Description

Recipe

produces

requires

hasCookingInstructions

hasRequiredTime

hasInitialSituation

hasGoal

hasPlanDescription

Recipe

hasConstituent

QuantityOfFood
Food

An abstract **quantity** of food.

Pattern:
QuantityOfStuff (with Quantity sub-pattern)

(derived from QUDT)
Equipment

No complex model desired at this stage. We just want to use strings, i.e., use our stub meta-pattern.

Figure 2.10: Top, the Stub (meta)pattern. Bottom, its instantiation for equipment.
Classification (e.g., entrée)

No complex model desired at this stage. We just want to use strings, i.e., use our stub meta-pattern.
Difficulty level

No complex model desired at this stage. We just want to use strings, i.e., use our stub meta-pattern.
Already incorporated in plan!
Use an ontology design pattern based on PROV-O.

PROV-O derived Provenance pattern:

We’ll use only this:
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. **Develop the remaining modules from scratch.**
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
Nutritional information

Model along some existing standard.

Let’s use the U.S. FDA Nutritional Facts label standard.
Nutritional information

Model along some existing standard.

Figure 2.13: Nutritional Information module. The box indicates a modified instance of the QuantityOfStuff pattern.
Adequacy check

- Triplify sample data using the ontology. Does it work?
- Check if competency questions can be answered.
- Add axioms as appropriate (the graph is only for intuition, the OWL axioms are the actual ontology).
- (there are more post-hoc details to be taken care of, but let’s leave it at that)
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
Axiomatization

Figure 2.17: Generic node-edge-node schema diagram for explaining systematic axiomatization

1. $A \cap B \subseteq \bot$
2. $\exists R . T \subseteq A$
3. $\exists R . B \subseteq A$
4. $T \subseteq \forall R . B$
5. $A \subseteq \forall R . B$
6. $A \subseteq R . B$
7. $B \subseteq R^{-} . A$
8. $T \subseteq \leq 1 R . T$
9. $T \subseteq \leq 1 R . B$
10. $A \subseteq \leq 1 R . T$
11. $A \subseteq \leq 1 R . B$
12. $T \subseteq \leq 1 R^{-} . T$
13. $T \subseteq \leq 1 R^{-} . A$
14. $B \subseteq \leq 1 R^{-} . T$
15. $B \subseteq \leq 1 R^{-} . A$

Figure 2.18: Most common axioms which could be produced from a single edge $R$ between nodes $A$ and $B$ in a schema diagram: description logic notation.
Axiomatization

1. $A$ DisjointWith $B$
2. $R$ some owl:Thing SubClassOf $A$
3. $R$ some $B$ SubClassOf $A$
4. owl:Thing SubClassOf $R$ only $B$
5. $A$ SubClassOf $R$ only $B$
6. $A$ SubClassOf $R$ some $B$
7. $B$ SubClassOf inverse $R$ some $A$
8. owl:Thing SubClassOf $R$ max 1 owl:Thing
9. owl:Thing SubClassOf $R$ max 1 $B$
10. $A$ SubClassOf $R$ max 1 owl:Thing
11. $A$ SubClassOf $R$ max 1 $B$
12. owl:Thing SubClassOf inverse $R$ max 1 owl:Thing
13. owl:Thing SubClassOf inverse $R$ max 1 $A$
14. $B$ SubClassOf inverse $R$ max 1 owl:Thing
15. $B$ SubClassOf inverse $R$ max 1 $A$

(disjointness) (domain) (scoped domain) (range) (scoped range) (existential) (inverse existential) (functionality) (qualified functionality) (scoped functionality) (qualified scoped functionality) (inverse functionality) (inverse qualified functionality) (inverse scoped functionality) (inverse qualified scoped functionality)

Figure 2.19: Most common axioms which could be produced from a single edge $R$ between nodes $A$ and $B$ in a schema diagram: Manchester syntax.
Example Axiomatization

ofFoodType, ofQuantity: scoped range, existential
hasQuantityKind, hasQuantityValue: scoped domain, scoped range, existential, inverse existential, scoped qualified functionality
hasUnit: scoped range, existential, scoped qualified functionality
hasNumericValue: scoped range, existential, functionality
Mutually disjoint: QuantityOfFood, FoodType, QuantityKind, Quantity, QuantityValue, Unit
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
Modeling process – steps

1. Define use case or scope of use cases
2. Make competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.
3. Identify key notions from the data and the use case and identify which pattern should be used for each (if a suitable pattern is available). Many can remain “stubs” if detailed modeling is not yet necessary.
4. Instantiate these key notions from the pattern templates (if there is a suitable pattern), and adapt/change the result as needed, arriving at modules. Develop the remaining modules from scratch.
5. Add axioms for each module, informed by the pattern axioms.
6. Put the modules together and add axioms which involve several modules.
7. Reflect on all class, property and individual names and possibly improve them. Also check module axioms whether they are still appropriate after putting all modules together.
8. Create OWL files.
Tools

• We are currently developing a set of compatible tools, as Protégé plug-ins.
  
  – See http://comodide.com/

• We are also developing ODP libraries.
  
  – See https://daselab.cs.ksu.edu/content/modl-modular-ontology-design-library
Thanks!
References


References


References


References


References


