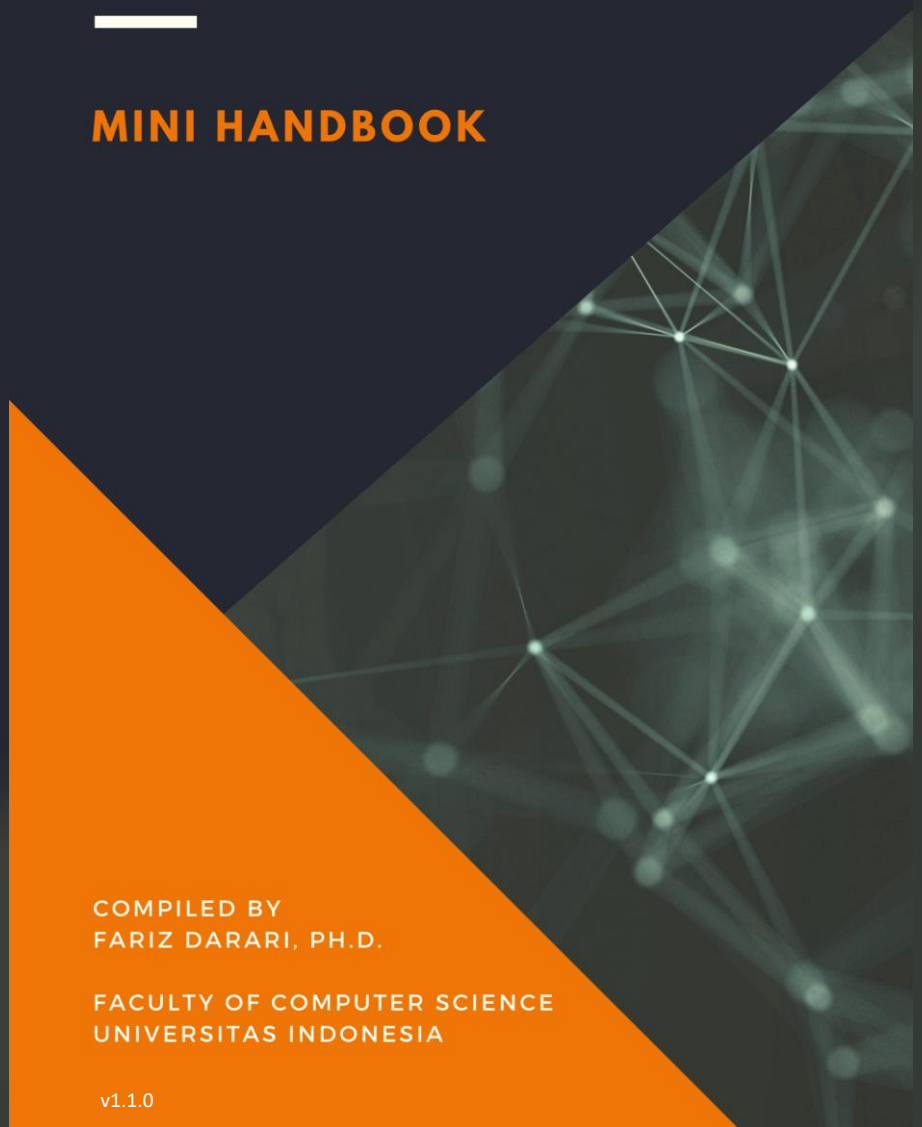
KNOWLEDGE GRAPHS

MINI HANDBOOK



COMPILED BY
FARIZ DARARI, PH.D.

FACULTY OF COMPUTER SCIENCE
UNIVERSITAS INDONESIA

v1.1.0

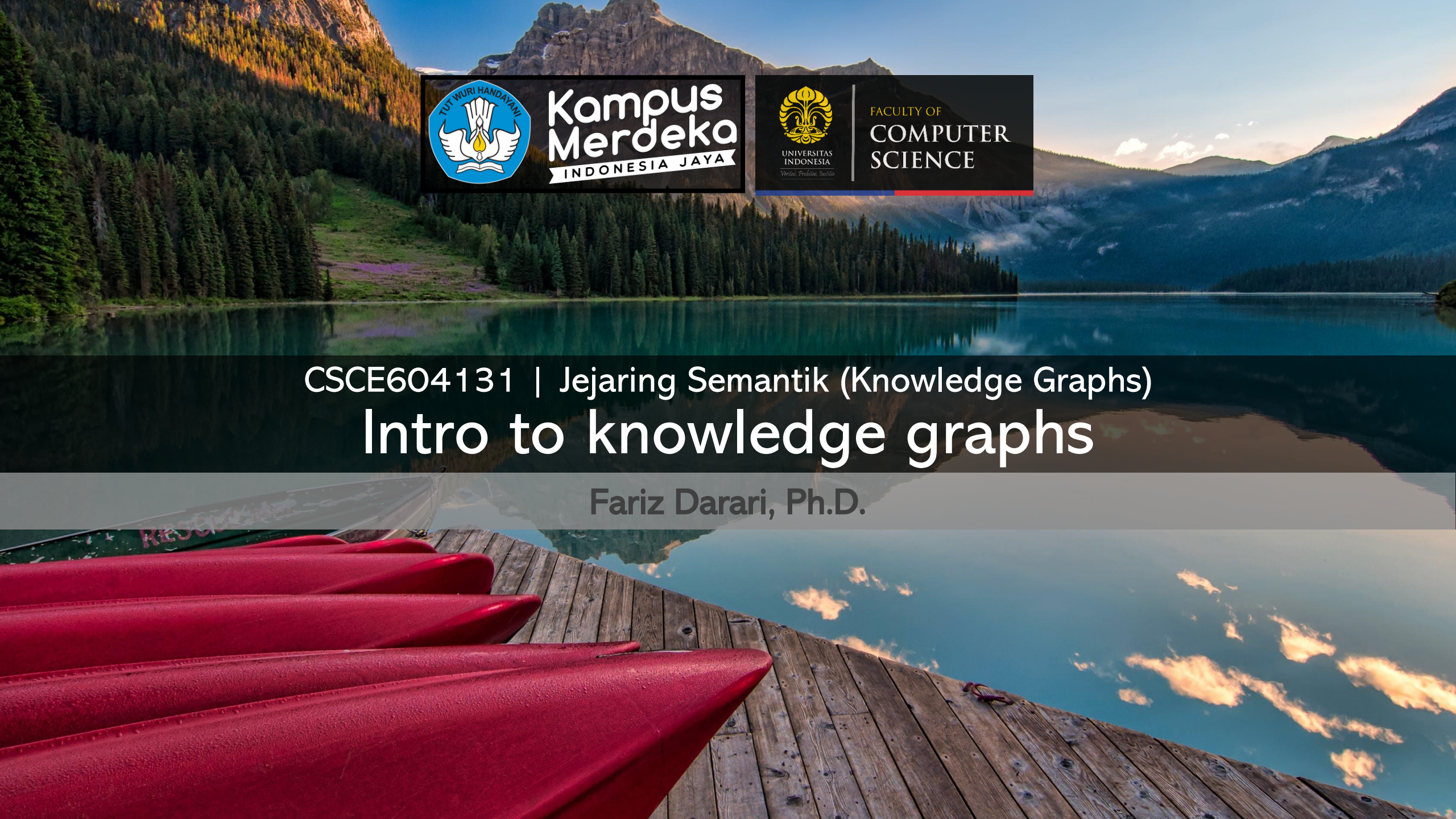
What this mini handbook is about

- This mini handbook is basically a collection of lecture notes of a MOOC course on knowledge graphs (KGs) by Fariz Darari, Ph.D.
- This mini handbook is meant to be a set of triggers for readers to look up for further and deeper exploration on knowledge graph topics
- This mini handbook is made possible by a lot of material already provided by knowledge graph communities, thank you!
- Any feedback or improvement suggestions are very welcome! Ping me at: fariz ----at---- ui ----dot---- ac ----dot---- id
- Enjoy this knowledge graph mini handbook as much as I enjoy compiling it!

Compiled by Fariz Darari

(Fariz Darari, Nov 2021)

Table of Contents (clickable)



CSCE604131 | Jejaring Semantik (Knowledge Graphs)

Intro to knowledge graphs

Fariz Darari, Ph.D.

Let's suppose that ...

... you would like to build an application to **visualize and analyze COVID-19.***

What would you primarily need?

*Fariz Darari. COVIWD: COVID-19 Wikidata Dashboard. Jurnal Ilmu Komputer dan Informasi (Journal of Computer Science and Information), 14(1):39-47, 2021. Available in English at: https://s.id/coviwd-pdf

Let's suppose that ...

... you would like to build an application to **visualize and analyze COVID-19.***

What would you <u>primarily</u> need? **Data!**

*Fariz Darari. COVIWD: COVID-19 Wikidata Dashboard. Jurnal Ilmu Komputer dan Informasi (Journal of Computer Science and Information), 14(1):39-47, 2021. Available in English at: https://s.id/coviwd-pdf

Closed vs. Open Data

Have you ever been in a situation where you would like to build an application about something (that requires data), but you **don't have access** to the data?
You are not alone!

vs.

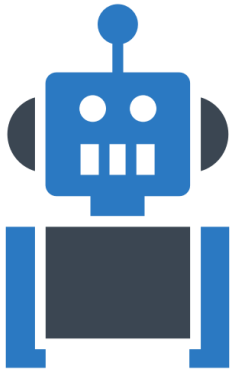
Compiled by Fariz Darari

Further initiative on data openness (in higher education domain):
https://s.id/tridharma-terbuka-vid

Human Readable vs. Machine Readable

Have you ever been in a situation where you would like to build an application, and you already have the data, but the data is just **hard to process**?
You are not alone!

VS.



JPG, PNG, PDF

XLS, CSV

Tabular vs. Graph

Have you ever been in a situation where you would like to build an application, and you want not only data that is machine-readable, but also is flexible and human-interpretable? **You are not alone!**

VS.

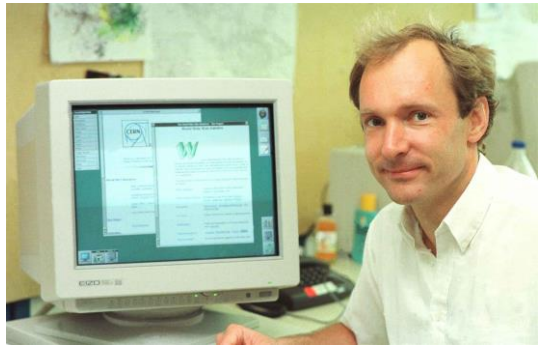
Tabular

Graph

Separated vs. Integrated/Linked

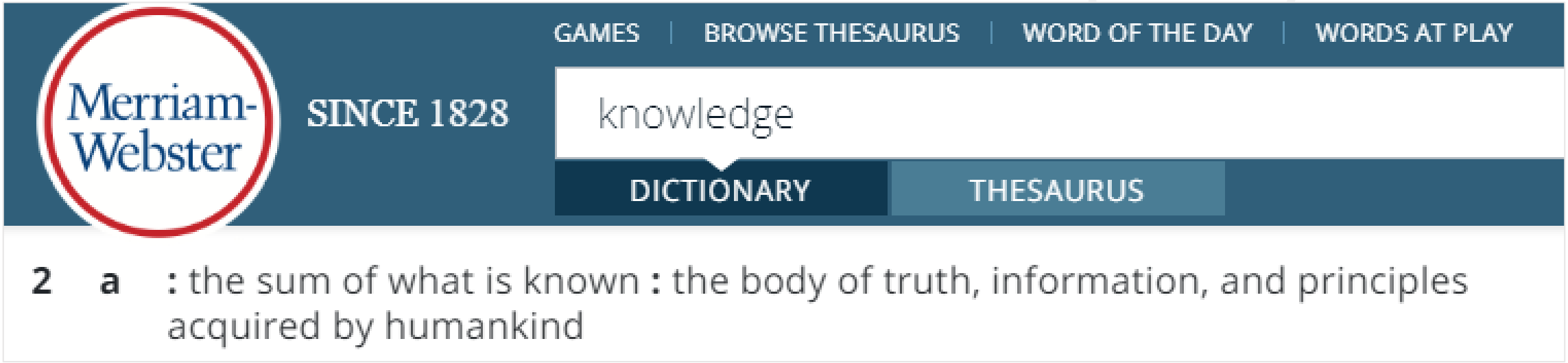
Have you ever been in a situation where you would like to build an application, but the data you want for the application is **separated across different sources**?
You are not alone!

5-Star Data



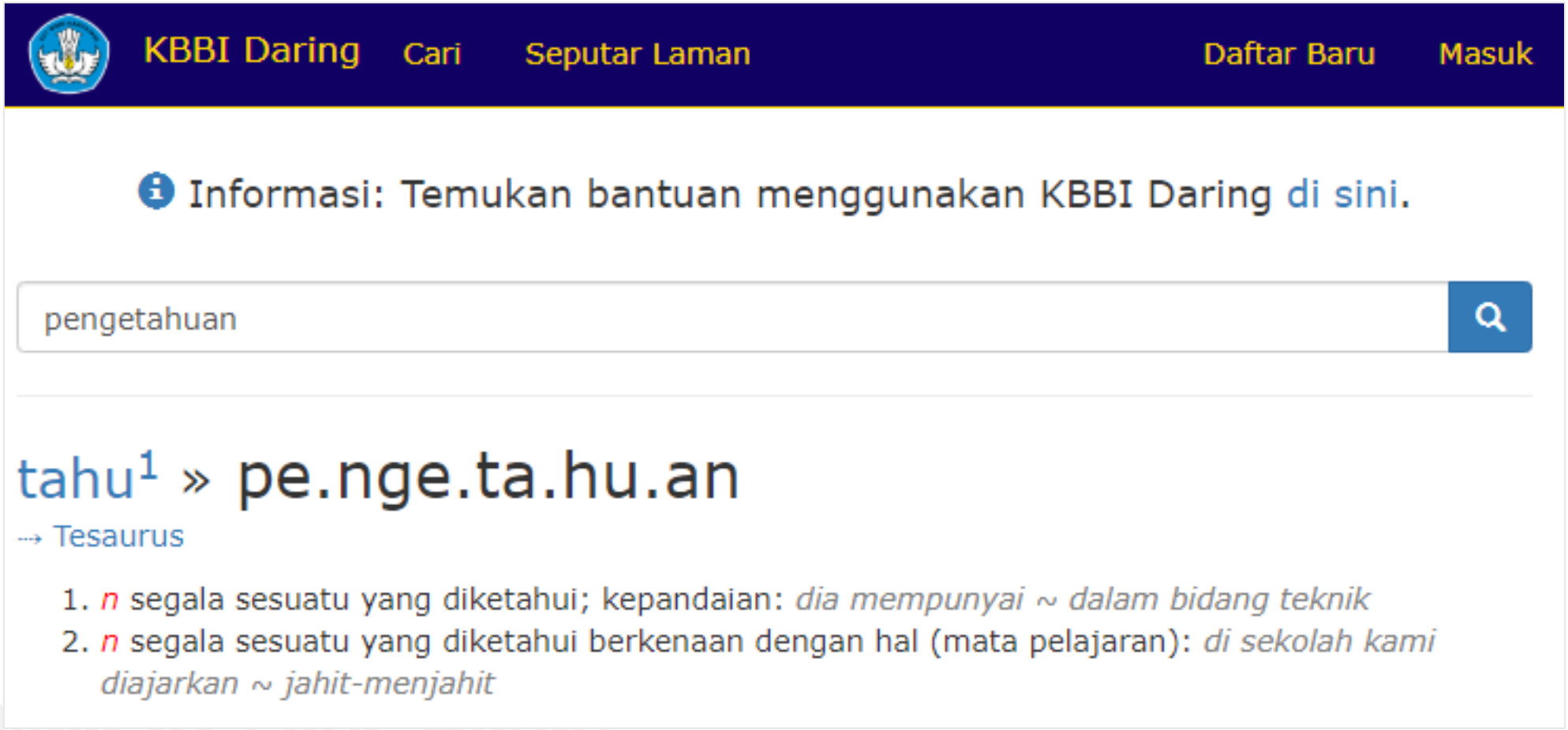
Tim-Berners Lee
(Web inventor)
Image from CERN

What is “knowledge”?

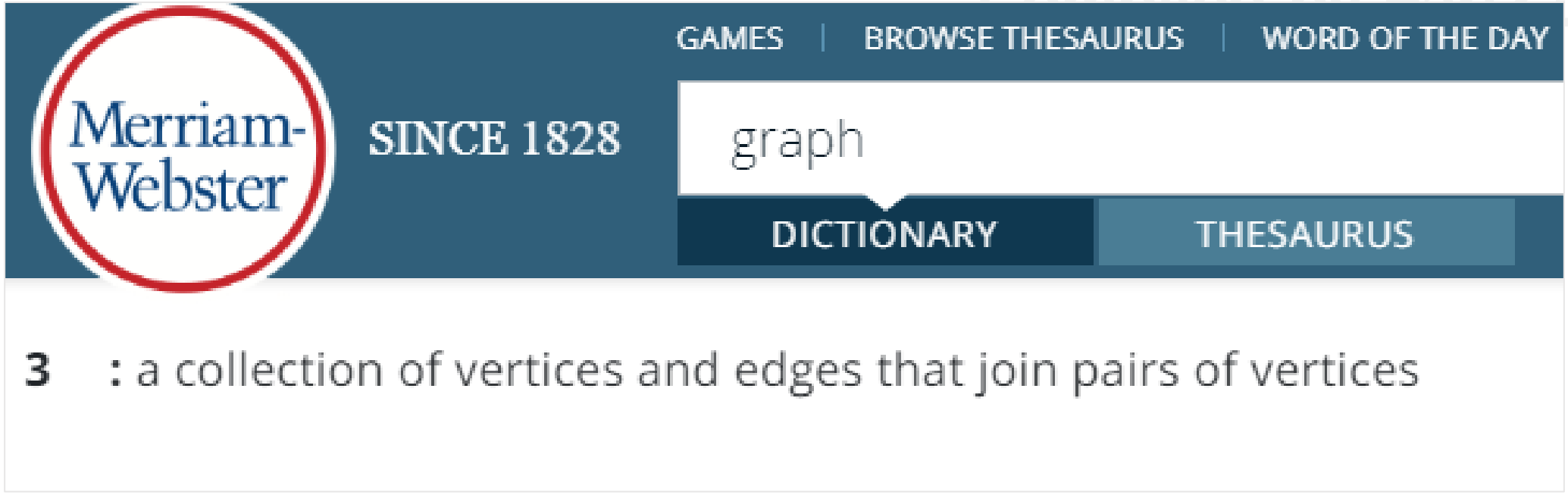


Compiled by Fariz Darari

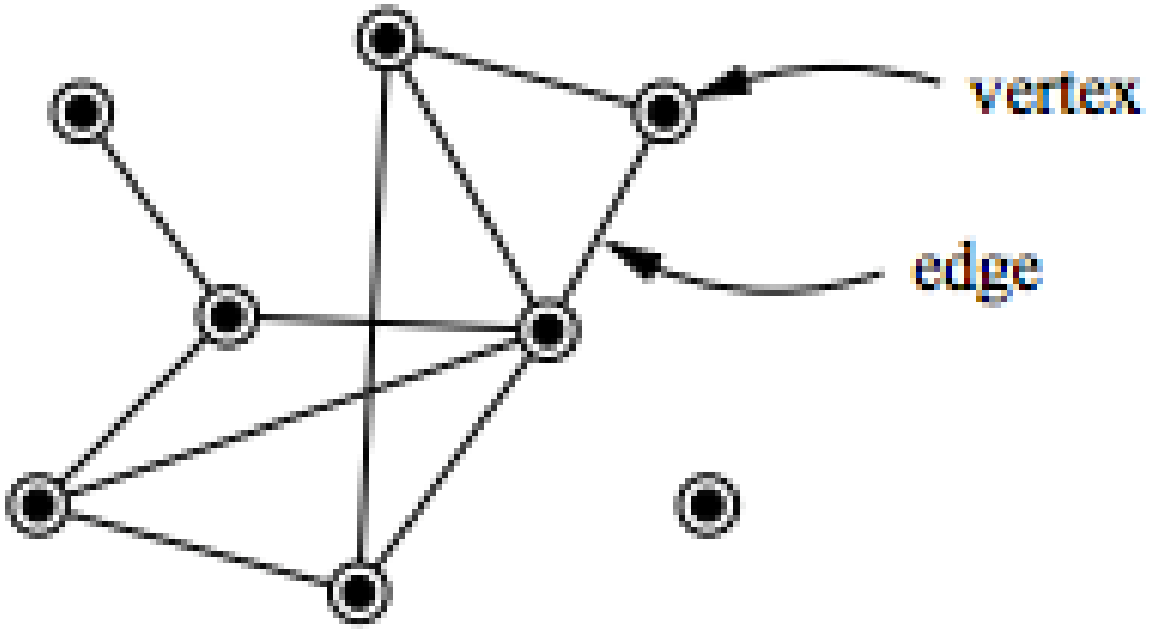
What is “knowledge”?



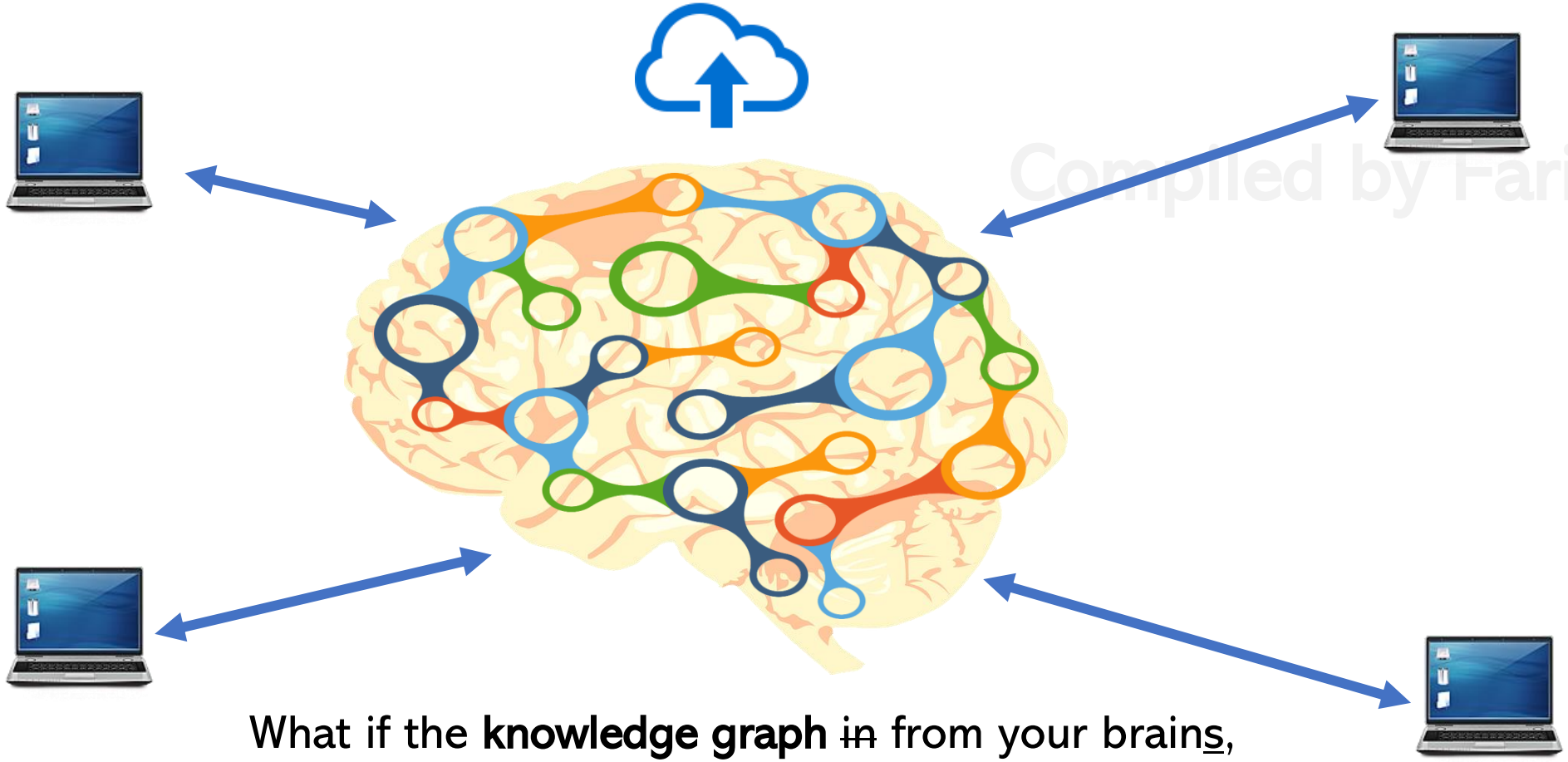
What is “graph”?



What is “graph”?

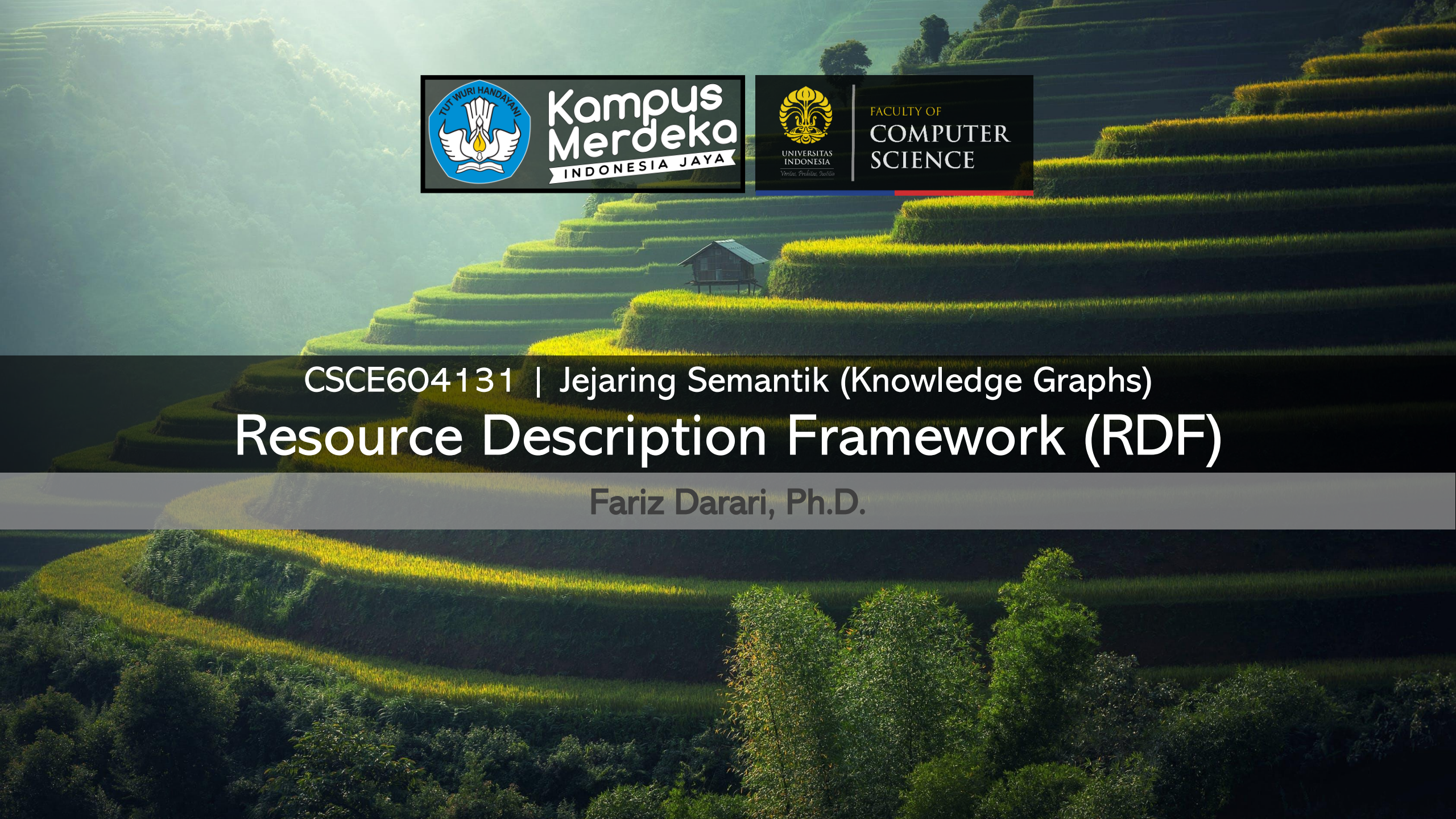


So a “knowledge graph” is …



What if the **knowledge graph** ~~in~~ from your brain**s**,
can be uploaded onto the **Web**, integrated with knowledge graphs
from others, and then processed and queried by computers?

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia

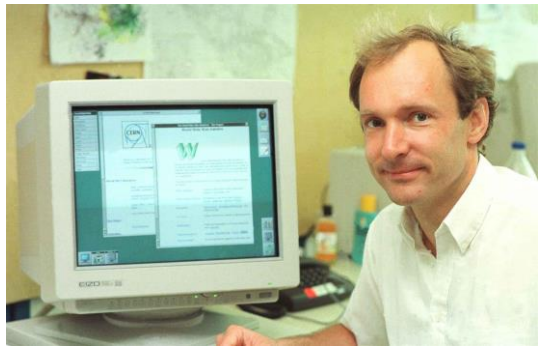


CSCE604131 | Jejaring Semantik (Knowledge Graphs)

Resource Description Framework (RDF)

Fariz Darari, Ph.D.

5-Star Data



Tim-Berners Lee
(Web inventor)
Image from CERN

KG standards

the **data** part

the **schema** part

the **query** part

KG standard: RDF

- Data model, based on **S-P-O triple** structure (Subject, Predicate, Object)
- Used for describing things, yes, every, single, thing
  And anyways, RDF = **Resource Description Framework**
- Key features:
  > RDF is intuitive
  > RDF is flexible
  > RDF is machine-readable and can be exported into many formats
  > RDF describes & links things, not just documents
  > RDF links are typed

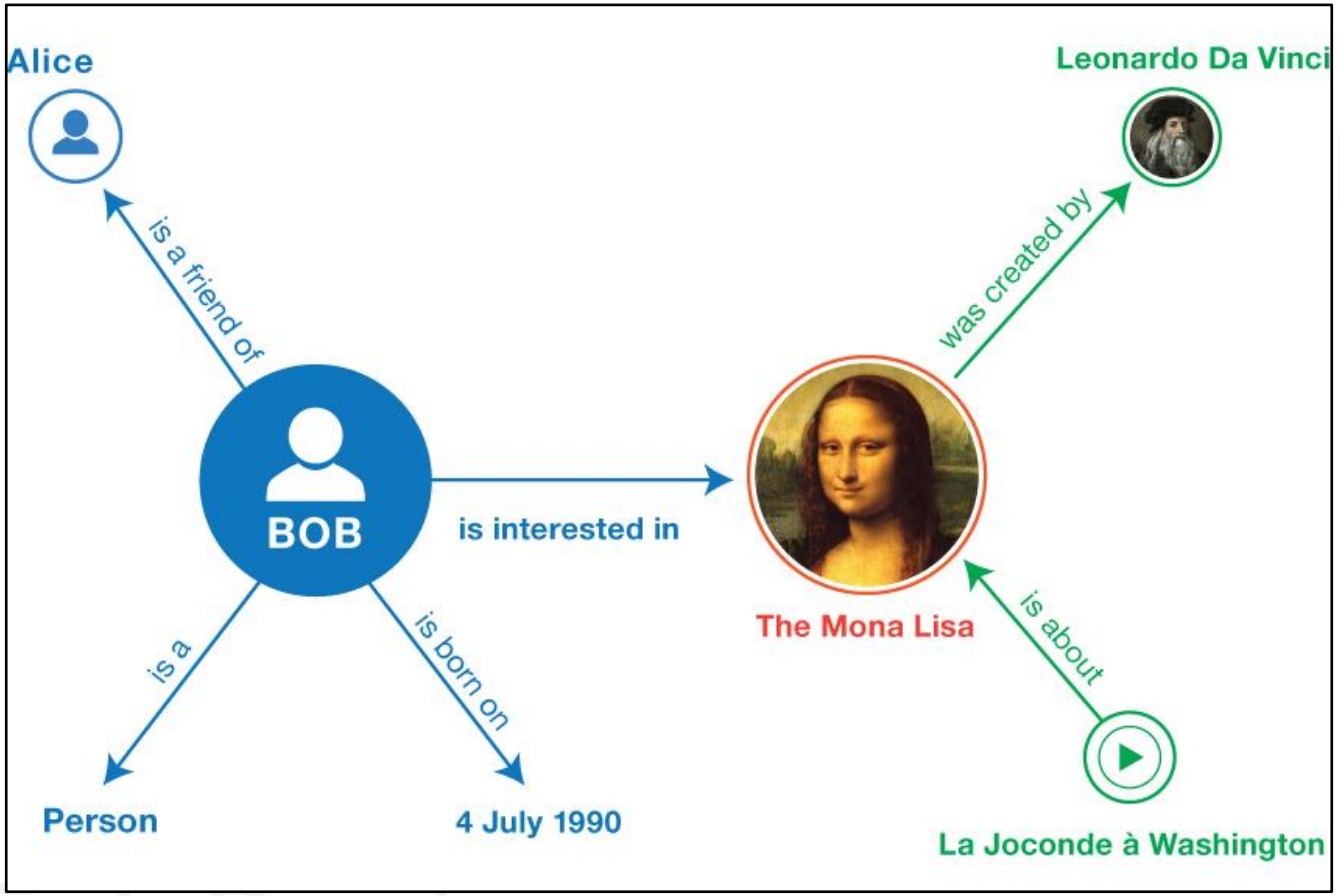
Compiled by Fariz Darari

KG standard: RDF Knowledge Graph (in text)


```
<Bob> <is a> <person>.
<Bob> <is a friend of> <Alice>.
<Bob> <is born on> <the 4th of July 1990>.
<Bob> <is interested in> <the Mona Lisa>.
<the Mona Lisa> <was created by> <Leonardo da Vinci>.
<the video 'La Joconde à Washington'> <is about> <the Mona Lisa>.
```


https://www.w3.org/TR/rdf11-primer/

KG standard: RDF Knowledge Graph



https://www.w3.org/TR/rdf11-primer/

Four Key Principles for RDF Graphs

- Uniform Resource Identifiers (URIs) as identifiers
- HTTP URI as information access protocol

Four Key Principles for RDF Graphs

So basically RDF is built on top of the (proven) web standards!

- Uniform Resource Identifiers (URIs) as identifiers
- HTTP URI as information access protocol

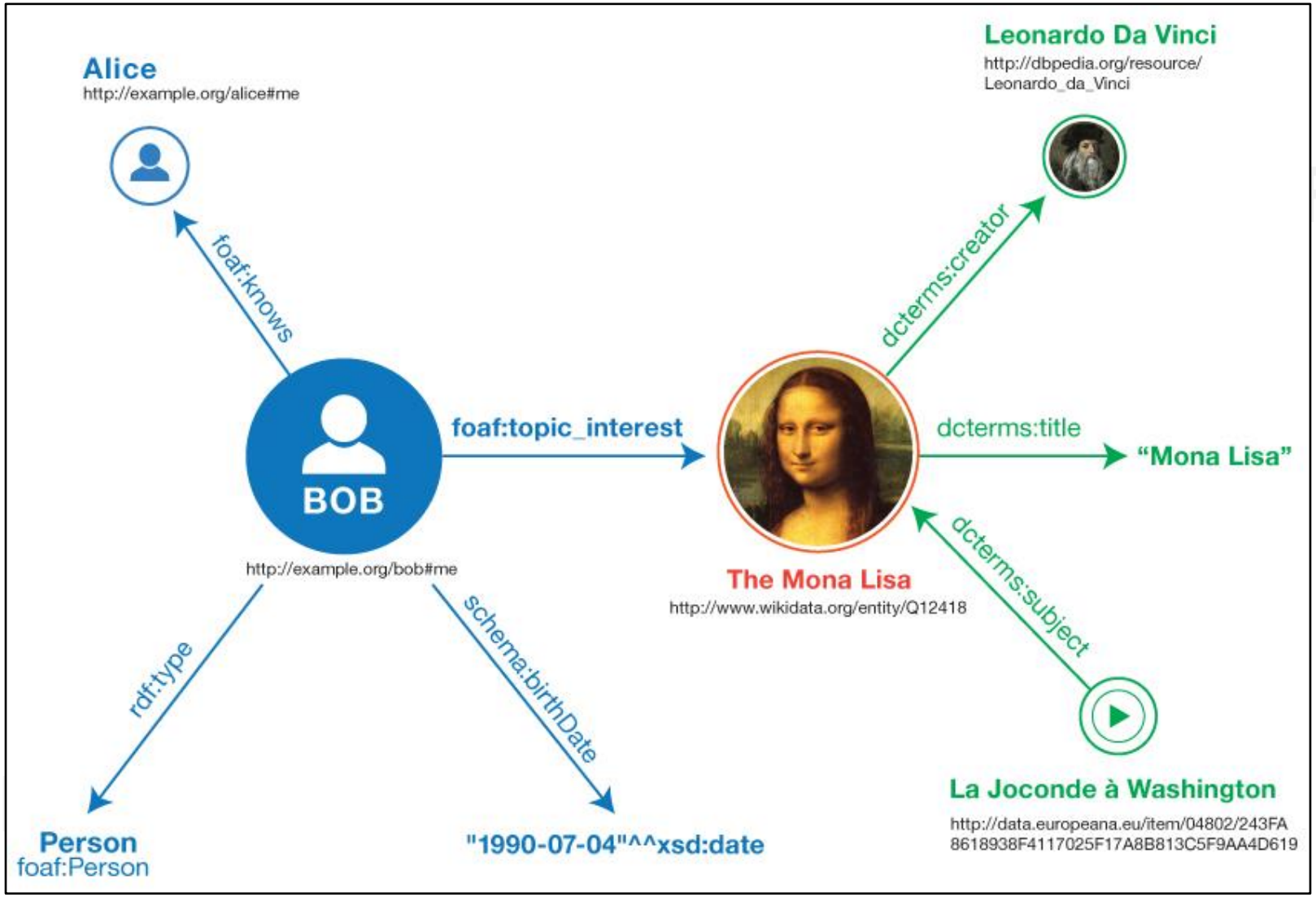
Four Key Principles for RDF Graphs

So basically RDF is built on top of the (proven) web standards!

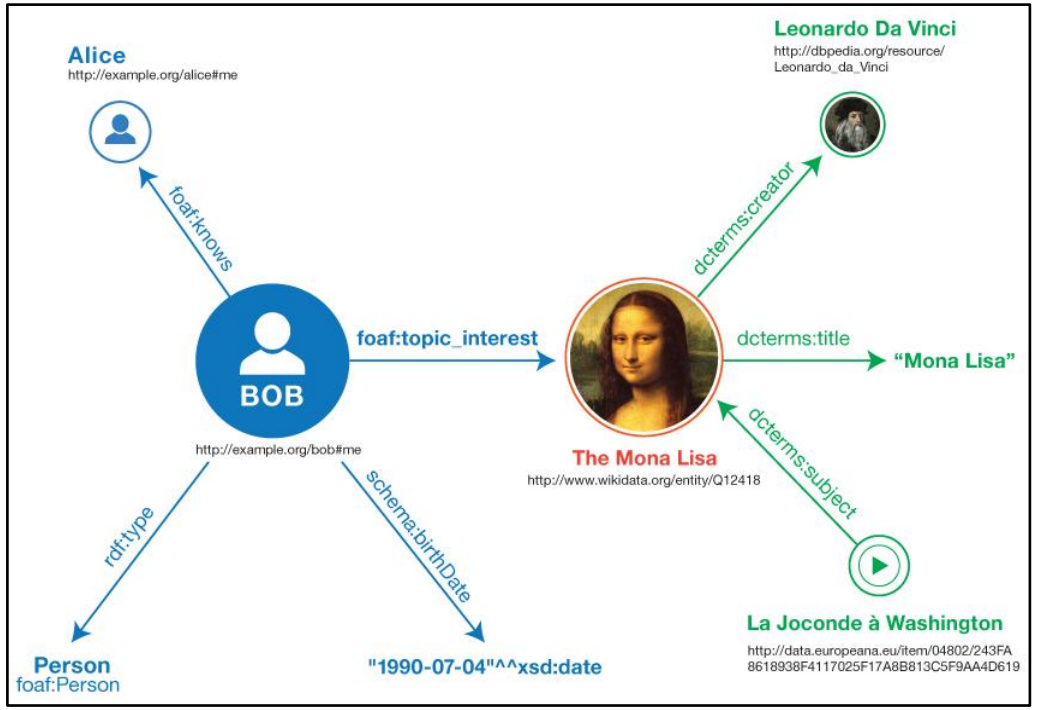
- Uniform Resource Identifiers (URIs) as identifiers
- HTTP URI as information access protocol
- RDF and SPARQL as knowledge representation and querying languages
- Link information from one source to other

Compiled by Fariz Darari

RDF Knowledge Graph Revisited



RDF Knowledge Graph Revisited



Bob can be identified through the URI http://example.org/bob#me and Mona Lisa through the URI http://www.wikidata.org/entity/Q12418.

Property birthDate is identified through http://schema.org/birthDate and title through URI http://purl.org/dc/terms/.

A KG can contain literal values such as "Mona Lisa " and "1990-07-04".

Literals in RDF

- Literals are basic values that are not URIs.
- Examples of literals include strings such as "La Joconde", dates such as "the 14th of July, 1990" and numbers such as "3.14159".
- Literals are associated with a datatype (like the ones defined by XML Schema) enabling such values to be parsed and interpreted correctly. For example:
  > "La Joconde"^^xsd:string
  > "1990-07-04"^^xsd:date
  > "3.14159"^^xsd:double
- String literals can optionally be associated with a language tag. Example:
  > "Thank You"@en
  > "Grazie"@it
  > "Danke"@de

Blank Nodes in RDF

- Through blank nodes, RDF can represent anonymous resources
- So to capture something like: There exists a student with pet name "Tom"
- Blank nodes can have local identifiers but not global identifiers
- For that reason, use blank nodes carefully
- Typical uses:
    - > You are not sure of the URI
    - > You do not want to provide a URI
    - > Logical grouping of related properties

Compiled by Fariz Darari

Thus, the elements of an RDF triple are...

- Three types of elements can be used in a triple:
  - > URIs: Uniform Resource Identifiers
  - > Literals: Basic/primitive values which are not URIs
  - > Blank nodes: Anonymous/local resources
- Rules on their use in an RDF triple:

	Subject	Predicate	Object
URI	✓	✓	✓
Literal	✗	✗	✓
Blank Node	✓	✗	✓

https://people.montefiore.uliege.be/binot/INFO8005/Theory/Semantic-data-ch3.pdf

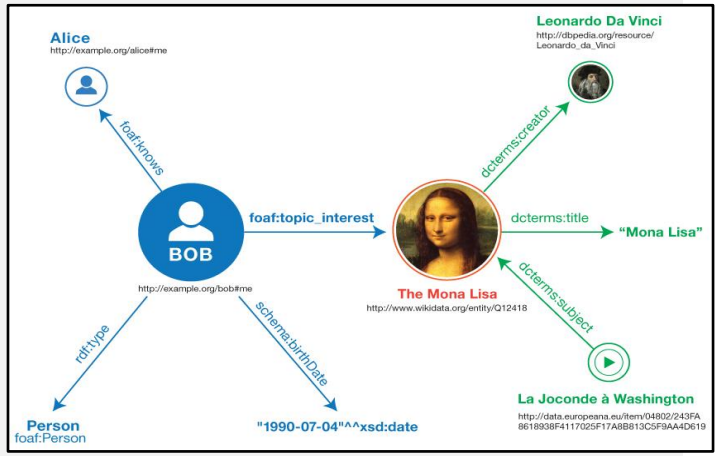
RDF Serialization

- An RDF graph must be serialized to be stored into files.
- Several syntaxes exist:
  - > RDF/XML, first serialization format for RDF
  - > N-Triples, trivial to parse/serialize
  - > Turtle, popular & more human-readable
  - > JSON-LD, JSON-based & more familiar to web devs

Compiled by Fariz Darari

RDF Graph in Turtle Format

```
BASE   <http://example.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX schema: <http://schema.org/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX wd: <http://www.wikidata.org/entity/>

<bob#me>
    a foaf:Person ;
    foaf:knows <alice#me> ;
    schema:birthDate "1990-07-04"^^xsd:date ;
    foaf:topic_interest wd:Q12418 .
# continue next page ...
```



RDF Graph in Turtle Format (cont.)

```
wd:Q12418
    dcterms:title "Mona Lisa" ;
    dcterms:creator
<http://dbpedia.org/resource/Leonardo_da_Vinci> .

<http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813
C5F9AA4D619>
        dcterms:subject wd:Q12418 .
```

Real-world Example: Indonesia on DBpedia KG

- URI to identify the resource Indonesia:
  http://dbpedia.org/resource/Indonesia

- URI providing human-readable information (HTML)
  http://dbpedia.org/page/Indonesia
-
  URI providing machine-readable information (RDF)
  http://dbpedia.org/data/Indonesia

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia

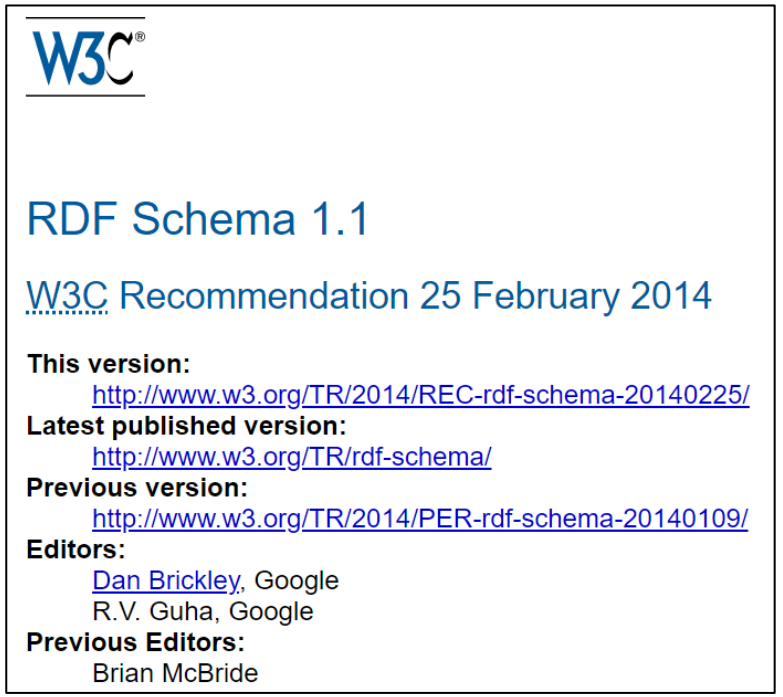
CSCE604131 | Jejaring Semantik (Knowledge Graphs)

RDF Schema (RDFS)

Fariz Darari, Ph.D.

RDF Schema (RDFS)

- RDFS is a vocabulary for modeling schemas used in RDF data.
- RDFS provides mechanisms for defining classes and properties.
- Via RDFS, e.g., one can provide the domains and ranges of properties.
- The official document of RDFS is available:
  https://www.w3.org/TR/rdf-schema/


W3C

RDF Schema 1.1

W3C Recommendation 25 February 2014

This version:
http://www.w3.org/TR/2014/REC-rdf-schema-20140225/
Latest published version:
http://www.w3.org/TR/rdf-schema/
Previous version:
http://www.w3.org/TR/2014/PER-rdf-schema-20140109/
Editors:
Dan Brickley, Google
R.V. Guha, Google
Previous Editors:
Brian McBride

RDFS Classes

- Resources (or things) can be organized into groups called **classes**.
- The members of a class are called **instances** of the classes.
- The property **rdf:type** is used to state that a resource is an instance of a class.

Compiled by Fariz Darari

Properties

- An **RDF property** is a relation between subject resources and object resources.
- Through RDFS, one can state the **domain** of a property:
  any resource that has a given property is an instance of a class.
- Through RDFS, one can state the **range** of a property:
  the values of a property are instances of a class.

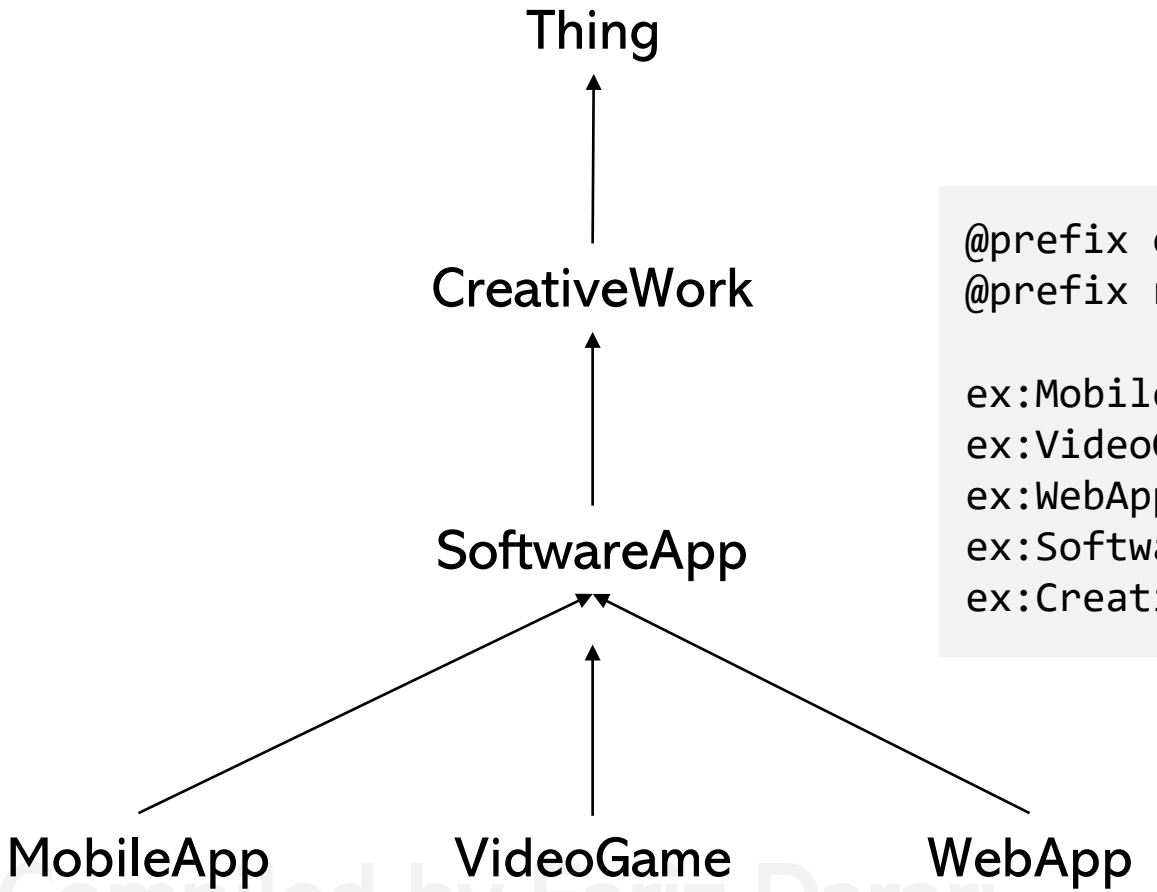
Compiled by Fariz Darari

RDFS subClassOf and subPropertyOf

- The property **rdfs:subClassOf** is used to state that all instances of one class are instances of another class.

- The property **rdfs:subPropertyOf** is used to state that all resources related by one property are also related by another property.

RDFS subClassOf Example




```
@prefix ex: <https://example.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

ex:MobileApp    rdfs:subClassOf  ex:SoftwareApp.
ex:VideoGame    rdfs:subClassOf  ex:SoftwareApp.
ex:WebApp       rdfs:subClassOf  ex:SoftwareApp.
ex:SoftwareApp  rdfs:subClassOf  ex:CreativeWork.
ex:CreativeWork rdfs:subClassOf  ex:Thing.
```

Other RDFS properties: Documentation

- rdfs:seeAlso
  Defines a relation of a resource to another, which explains it.
- rdfs:comment
  Comment, usually as text.
- rdfs:label
  Human-readable name of a resource (contrary to ID).

RDFS Example about Cultural Heritage (in RDF Turtle)

```
@prefix bkb: <https://budayakb.cs.ui.ac.id/ns#> .
@prefix bkbr:  <https://budayakb.cs.ui.ac.id/resource/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

bkb:WarisanBudaya a rdfs:Class . # a is shortcut for rdf:type
bkb:WarisanBudaya rdfs:label "Cultural Heritage"@en .

bkb:WarisanBudayaBenda rdfs:subClassOf bkb:WarisanBudaya .
bkb:WarisanBudayaBenda rdfs:label "Tangible Cultural Heritage"@en .

bkb:originFromProvince a rdf:Property .
bkb:originFromProvince rdfs:label "origin from province"@en .
bkb:originFromProvince rdfs:range bkb:Provinsi .
```

RDFS Example about Cultural Heritage (cont.)

```
bkbr:GamelanBali a bkb:WarisanBudayaBenda ;
   bkb:originFromProvince bkbr:Bali .
```

In the example above, Gamelan Bali is not only a Warisan Budaya Benda (Tangible Cultural Heritage) but also a Warisan Budaya (Cultural Heritage) due to the RDFS statement that Warisan Budaya Benda is a subclass of Warisan Budaya.

Furthermore, we know from the range of originFromProvince that Bali is indeed of type Provinsi.

RDFS Example about Greenhouse Gas (by Prof. Harald Sack)

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@orefix owl:  <http://www.w3.org/2002/07/owl#>
@prefix :     <http://example.org/Climate#> .
```

```
:Greenhouse_gas      rdf:type         rdfs:Class ;
                     rdfs:subClassOf  :Air_pollutant .
:Person              rdf:type         rdfs:Class .
:Scientist           rdfs:subClassOf  :Person .
:Physicist           rdfs:subClassOf  :Scientist .
:Chemist             rdfs:subClassOf  :Scientist .
```

Class Definitions

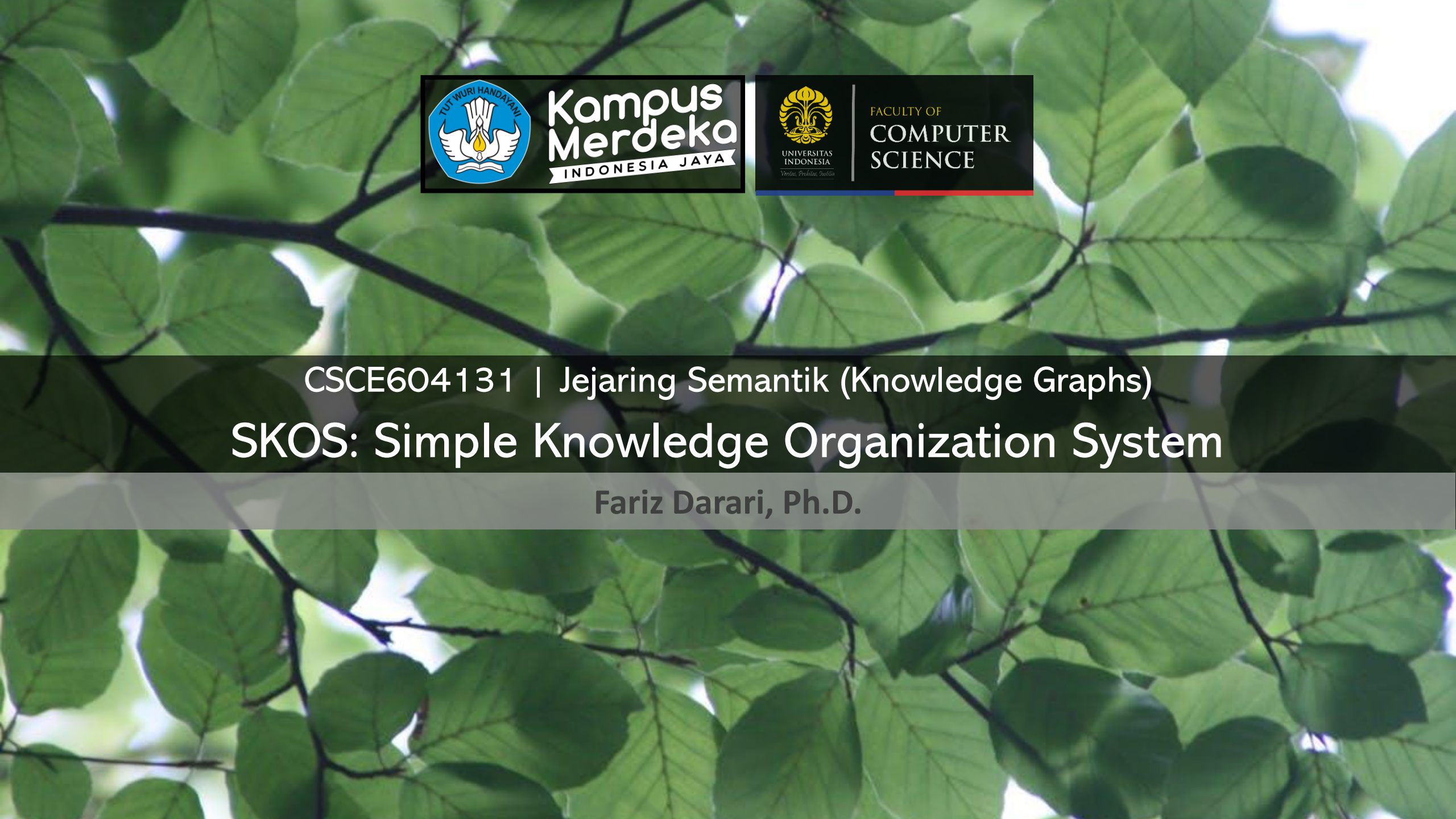
```
:discoverer     rdf:type     rdf:Property ;
                rdfs:domain  owl:Thing .
                rdfs:range   :Person .
```

Property Definitions

```
:Carbon_dioxide             rdf:type      :Greenhouse_gas ;
                            :discoverer   :Jan_Baptist_van_Helmont ;
                            :discoverer   :Joseph_Black .
:Jan_Baptist_van_Helmont    rdf:type      :Physicist .
:Joseph_Black               rdf:type      :Chemist ;
                            rdfs:label    "Joseph Black"@en ;
                            rdfs:comment  "co-discovered CO2" .
```

Instance Definitions

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia



CSCE604131 | Jejaring Semantik (Knowledge Graphs)

SKOS: Simple Knowledge Organization System

Fariz Darari, Ph.D.

SKOS

- SKOS is short for Simple Knowledge Organization System.
- It is a vocabulary for expressing the basic structure and content of concept schemes such as thesauri and taxonomies.
- SKOS allows concepts and relationships among concepts to be composed and published on the World Wide Web.
- The official documentation links of SKOS are at:
  > https://www.w3.org/TR/skos-primer/
  > https://www.w3.org/TR/skos-reference/

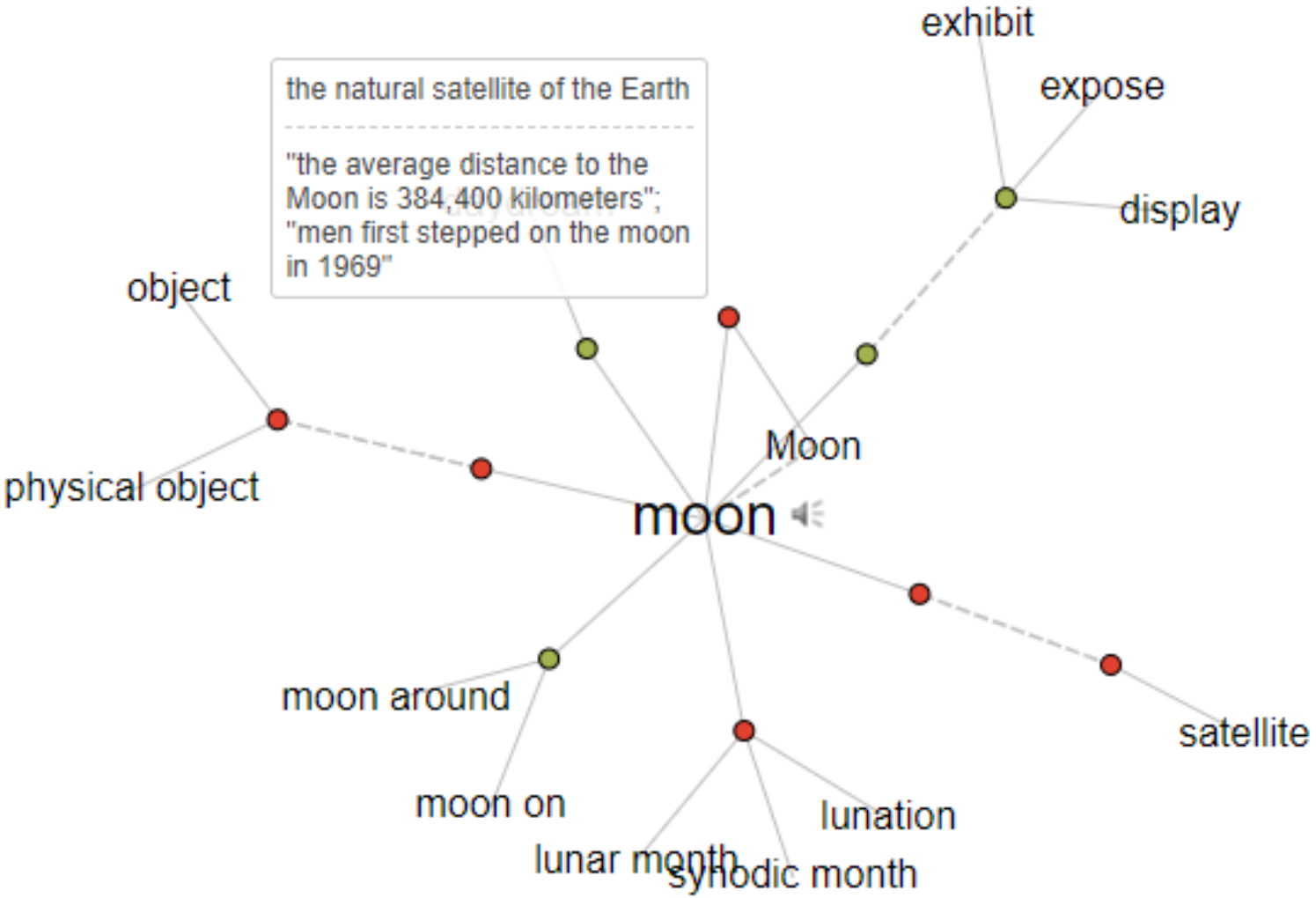
Compiled by Fariz Darari

SKOS to create controlled vocabularies

So basically, SKOS enables us to create a controlled vocabulary.

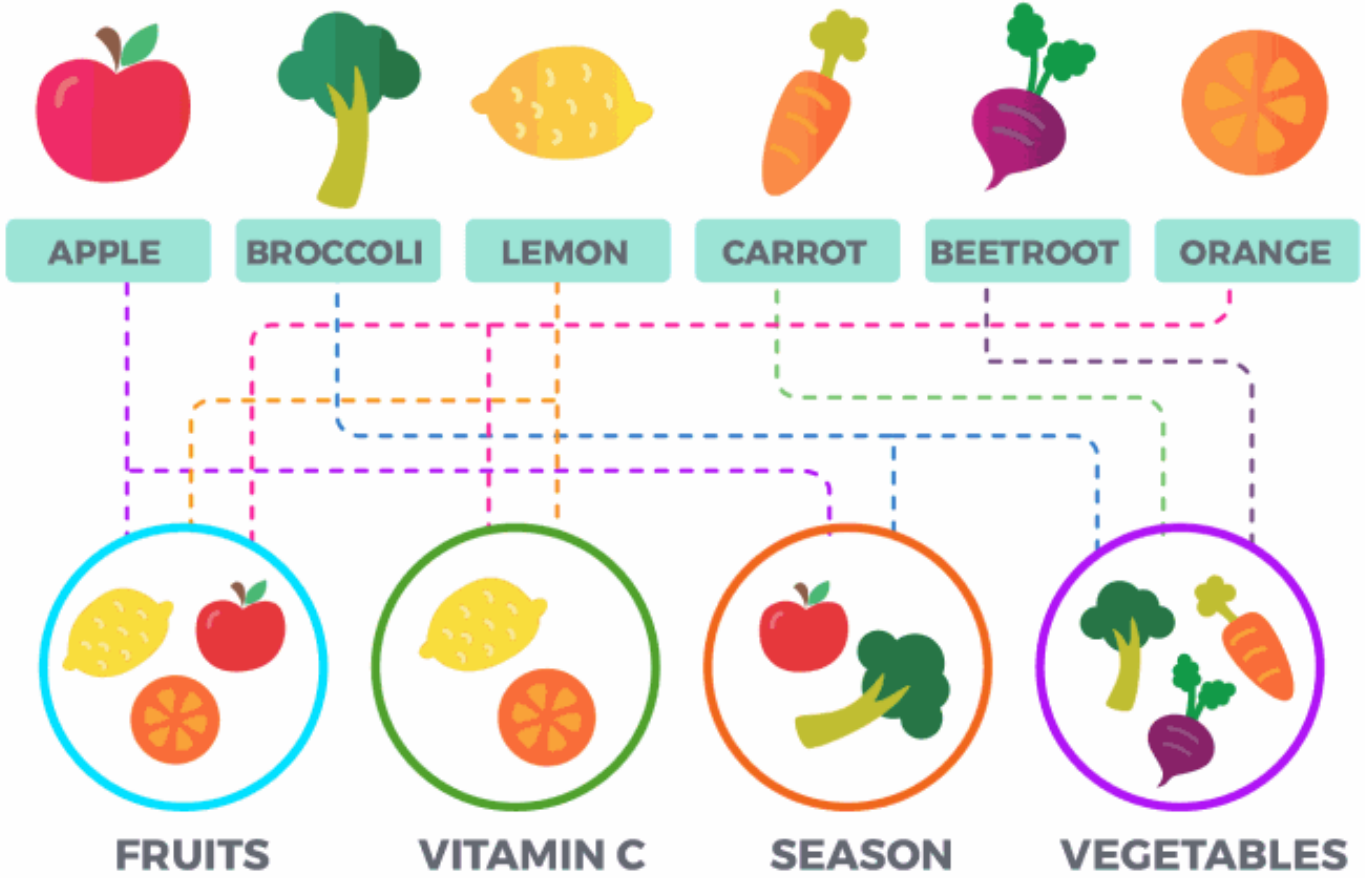
A **controlled vocabulary** is an organized arrangement of words and phrases used to index content and/or to retrieve content through browsing or searching. It typically includes preferred and variant terms and has a defined scope or describes a specific domain. (Harpring, P., 2010)

Thesaurus



Credits: https://www.visualthesaurus.com/

TAXONOMY: Label or tag content into defined categories so we can rearrange, reuse, organize, search, and distribute it.



Credits: Simple A

SKOS Basics

- Concept is the central element of SKOS
- To create a concept, one has to create a Uniform Resource Identifier (URI) for the concept and assign the URI using rdf:type skos:Concept
- SKOS provides properties to attach labels to a concept: skos:prefLabel, skos:altLabel, and skos:hiddenLabel.

```
ex:cat rdf:type skos:Concept;
  skos:prefLabel "cat"@en;
  skos:altLabel  "kitten"@en;
  skos:prefLabel "Katze"@de.
```

SKOS Semantic Relationships

skos:broader and **skos:narrower** assert hierarchical relationships between concepts: that one concept has a broader concept or a narrower concept


```
ex:animals rdf:type skos:Concept;
  skos:prefLabel "animals"@en;
  skos:narrower  ex:mammals.

ex:mammals rdf:type skos:Concept;
  skos:prefLabel "mammals"@en;
  skos:broader   ex:animals.
```

SKOS Semantic Relationships (cont.)

skos:related enables the representation of associative relationships between two concepts.


```
ex:FormulaOne skos:related ex:MichaelSchumacher .
ex:Vegetable  skos:related ex:Fruit .
ex:Pencil     skos:related ex:Writing .
```

Compiled by Fariz Darari

SKOS Collection & Membership

SKOS collections are used to describe groups of SKOS concepts.


```
ex:Veggies a skos:Collection ;
  skos:prefLabel "Sayur-mayur"@id ;
  skos:member ex:lettuce, ex:tomato, ex:onion,
              ex:cucumber, ex:carrot .
```

Compiled by Fariz Darari

SKOS Documentation

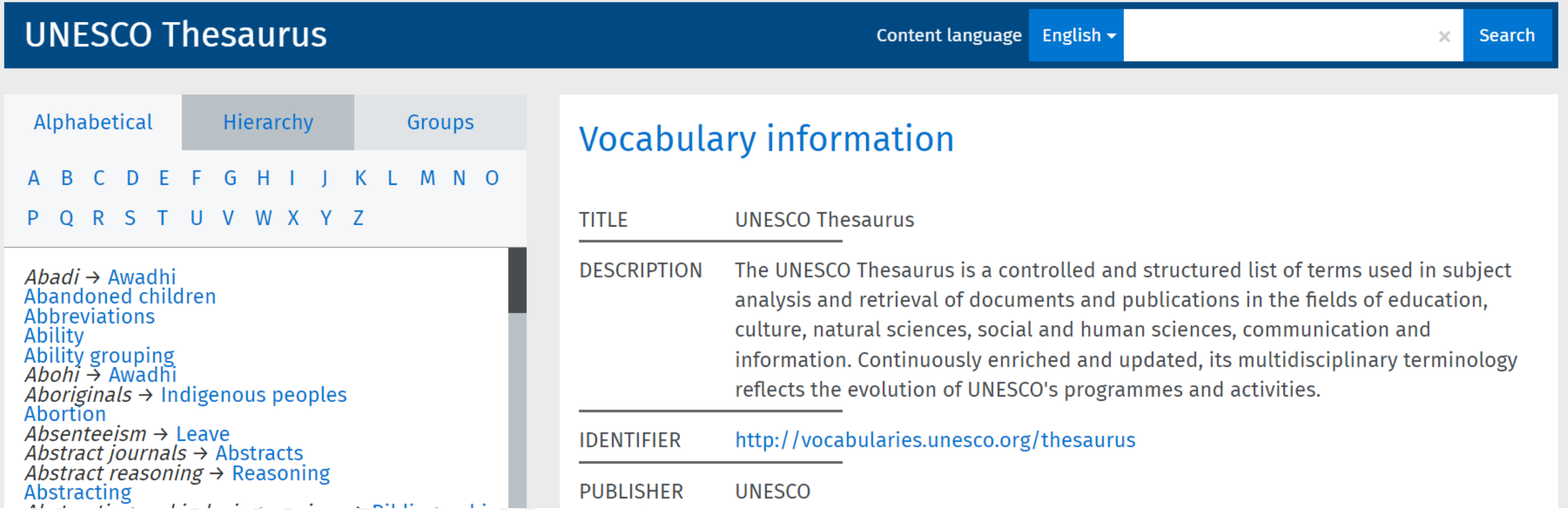
- skos:note: general documentation purposes
- skos:scopeNote: some information about a concept
- skos:definition: complete explanation of a concept
- skos:example: an example of the use of a concept
- skos:historyNote: significant changes to the meaning or form of a concept
- skos:editorialNote: administrative aid related to a concept
- skos:changeNote: fine-grained changes to a concept

Compiled by Fariz Darari

SKOS Documentation Example

```
ex:cat rdf:type skos:Concept;
  skos:prefLabel "cat"@en;
  skos:prefLabel "Katze"@de;
  skos:altLabel "kitten"@en;
  skos:hiddenLabel "katze"@de;
  skos:definition "A small carnivorous mammal with soft fur, a
short snout, and retractable claws."@en;
  skos:editorialNote "Review this term after merge."@en;
  skos:changeNote "Added hidden label."@en.
```

SKOS example: UNESCO Thesaurus



http://vocabularies.unesco.org/browser/thesaurus/en/

SKOS example: UNESCO Thesaurus (cont.)

UNESCO Thesaurus

ASEAN countries > Indonesia
Islamic countries > Indonesia
South East Asia > Indonesia
E-9 countries > Indonesia

PREFERRED TERM	**Indonesia** Search in UNESDOC
BROADER CONCEPT	ASEAN countries E-9 countries Islamic countries South East Asia
RELATED CONCEPTS	Timor-Leste
BELONGS TO GROUP	Countries and country groupings > Asia and the Pacific
IN OTHER LANGUAGES	اندونيسيا Arabic **Indonésie** French **Индонезия** Russian **Indonesia** Spanish
URI	http://vocabularies.unesco.org/thesaurus/concept830
Download this concept:	RDF/XML TURTLE JSON-LD Last modified 12/15/19

References:

- https://www.w3.org/TR/skos-primer/
- https://campus.dariah.eu/resource/controlled-vocabularies-and-skos
- https://www.ala.org/alcts/resources/z687/skos

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia

CSCE604131 | Jejaring Semantik (Knowledge Graphs)

SPARQL

Fariz Darari, Ph.D.

SPARQL

- If RDF captures knowledge, then SPARQL retrieves knowledge, querying knowledge captured by RDF!
- Short for: SPARQL Protocol and RDF Query Language
- Example:

W3C RDF

```
@prefix ex:   <https://example.org/resource/> .
@prefix prop: <https://example.org/property/> .

ex:matt prop:knows ex:scott .
ex:matt prop:twitterAccount <https://twitter.com/mattt7812> .
```

```
SELECT ?friend WHERE {
  ex:matt prop:knows ?friend }
```

Who are Matt's friends?

W3C SPARQL

SPARQL (cont.)


```
@prefix ex:   <https://example.org/resource/> .
@prefix prop: <https://example.org/property/> .

ex:matt prop:knows ex:scott .
ex:matt prop:twitterAccount <https://twitter.com/mattt7812> .
```

```
SELECT ?twitter WHERE {
  ?person prop:twitterAccount ?twitter .
  ?person prop:knows ex:scott }
```

What are the Twitter accounts of people knowing Scott?

SPARQL Documentation

W3C Recommendation

https://www.w3.org/TR/sparql11-query/

SPARQL 1.1 Query Language

W3C Recommendation 21 March 2013

This version:
http://www.w3.org/TR/2013/REC-sparql11-query-20130321/
Latest version:
http://www.w3.org/TR/sparql11-query/
Previous version:
http://www.w3.org/TR/2012/PR-sparql11-query-20121108/
Editors:
Steve Harris, Garlik, a part of Experian
Andy Seaborne, The Apache Software Foundation
Previous Editor:
Eric Prud'hommeaux, W3C

Please refer to the **errata** for this document, which may include some normative corrections.

See also **translations**.

Abstract

RDF is a directed, labeled graph data format for representing information in the Web. This specification defines the syntax and semantics of the SPARQL query language for RDF. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports aggregation, subqueries, negation, creating

SPARQL Example

RDF

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name   "Johnny Lee Outlaw" .
_:a  foaf:mbox   <mailto:jlow@example.com> .
_:b  foaf:name   "Peter Goodguy" .
_:b  foaf:mbox   <mailto:peter@example.org> .
_:c  foaf:mbox   <mailto:carol@example.org> .
```



```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
  { ?x foaf:name ?name .
    ?x foaf:mbox ?mbox }
```

Give me the name and email address of people!

SPARQL

name	mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

SPARQL Example (2)

RDF

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix :     <http://example.org/book/> .
@prefix ns:   <http://example.org/ns#> .

:book1  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```



```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title WHERE
{ ?x dc:title ?title
  FILTER regex(?title, "^SPARQL") }
```

Give me book titles started with SPARQL

SPARQL

title
"SPARQL Tutorial"

SPARQL Example (3)

RDF

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix :     <http://example.org/book/> .
@prefix ns:   <http://example.org/ns#> .

:book1  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```



```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title WHERE
{ ?x dc:title ?title
  FILTER regex(?title, "web", "i") }
```

Give me book titles containing "web" case insensitive

SPARQL

title
"The Semantic Web"

SPARQL Example (4)

W3C RDF

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix :     <http://example.org/book/> .
@prefix ns:   <http://example.org/ns#> .

:book1  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```



```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price WHERE {
  ?x ns:price ?price .
  FILTER (?price < 30.5)
  ?x dc:title ?title }
```

Give me book titles priced less than 30.5 USD

title	price
"The Semantic Web"	23

W3C SPARQL

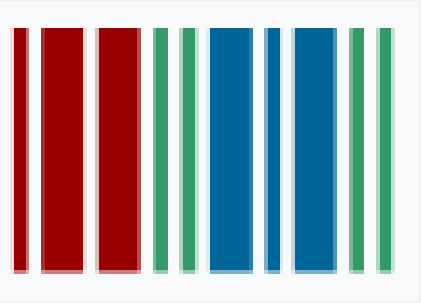
SPARQL Components

In general, a SPARQL query consists of four components:

- Prefixes
- Result form
- Query pattern (that is based on triple patterns)
- Solution modifier

SPARQL Components (cont.)

```
# query: legislation by UK Parliament
# link query: http://bit.ly/2NjRXhE

# prefixes
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>

# result form
SELECT ?lawLabel

# query pattern
{
  ?law wdt:P467 wd:Q11010 . # ?law - legislated by - Parliament of the UK
  ?law rdfs:label ?lawLabel .
  FILTER (LANG(?lawLabel) = "en")
}

# solution modifier
LIMIT 20
```



Over the Wikidata knowledge graph, give legislation by UK Parliament!

SPARQL Components (cont.)

lawLabel
Coinage (Measurement) Act 2011
Coinage Act 1870
Coinage Offences Act 1936
Coinage Offences Act 1832
Commissioner for Older People (Wales) Act 2006
Commissioners for Revenue and Customs Act 2005
Common Informers Act 1951
Commonhold and Leasehold Reform Act 2002
Commons Act 2006

. . . . .

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia

CSCE604131 | Jejaring Semantik (Knowledge Graphs)

Advanced SPARQL

Fariz Darari, Ph.D.

SPARQL Graph Patterns

SPARQL is based around graph pattern matching:

- Basic Graph Patterns (BGPs), where a set of triple patterns must match.
- Group Graph Pattern, where a set of graph patterns must all match.
- Optional Graph patterns, where additional patterns may extend the solution.
- Alternative Graph Pattern, where two or more possible patterns are tried.
- Patterns on Named Graphs, where patterns are matched against named graphs.

Basic Graph Pattern

Basic graph patterns are sets of triple patterns.

SPARQL graph pattern matching is defined in terms of combining the results from matching basic graph patterns.

A sequence of triple patterns, possibly with filters, comprises a single basic graph pattern.

SPARQL evaluates basic graph patterns using subgraph matching, which is defined for simple entailment.

Group Graph Pattern

In a SPARQL query string, a group graph pattern is delimited with braces: {}.

For example, this query's query pattern is a group graph pattern of one basic graph pattern.

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  {
          ?x foaf:name ?name .
          ?x foaf:mbox ?mbox .
       }
```

Empty Group Pattern

The group pattern { } matches any graph (incl. the empty graph) with one solution that does not bind any variables.

For example, the following

```
SELECT ?x
WHERE { }
```

matches one solution where var x is not bound.

Scope of Filters

A constraint, expressed by the keyword FILTER, is a restriction on solutions over the whole group in which the filter appears. The following patterns all have the same solutions:

```
{   ?x foaf:name ?name .
    ?x foaf:mbox ?mbox .
    FILTER regex(?name, "Smith")
}
```

```
{   FILTER regex(?name, "Smith")
    ?x foaf:name ?name .
    ?x foaf:mbox ?mbox .
}
```

```
{   ?x foaf:name ?name .
    FILTER regex(?name, "Smith")
    ?x foaf:mbox ?mbox .
}
```

Optional Patterns

```
@prefix foaf:       <http://xmlns.com/foaf/0.1/> .
@prefix rdf:        <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

_:a  rdf:type        foaf:Person .
_:a  foaf:name       "Alice" .
_:a  foaf:mbox       <mailto:alice@example.com> .
_:a  foaf:mbox       <mailto:alice@work.example> .

_:b  rdf:type        foaf:Person .
_:b  foaf:name       "Bob" .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  { ?x foaf:name  ?name .
         OPTIONAL { ?x  foaf:mbox  ?mbox }
       }
```

Optional Patterns (cont.)

name	mbox
"Alice"	<mailto:alice@example.com>
"Alice"	<mailto:alice@work.example>
"Bob"	

Compiled by Fariz Darari

Alternative Patterns

```
@prefix dc10:  <http://purl.org/dc/elements/1.0/> .
@prefix dc11:  <http://purl.org/dc/elements/1.1/> .

_:a  dc10:title     "SPARQL Query Language Tutorial" .
_:a  dc10:creator   "Alice" .

_:b  dc11:title     "SPARQL Protocol Tutorial" .
_:b  dc11:creator   "Bob" .

_:c  dc10:title     "SPARQL" .
_:c  dc11:title     "SPARQL (updated)" .
```

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>

SELECT ?title
WHERE  { { ?book dc10:title  ?title } UNION { ?book dc11:title  ?title } }
```

Alternative Patterns (cont.)

title
"SPARQL Protocol Tutorial"
"SPARQL"
"SPARQL (updated)"
"SPARQL Query Language Tutorial"

Compiled by Fariz Darari

Patterns on Named Graphs

When querying a collection of graphs, the GRAPH keyword is used to match patterns against named graphs.

GRAPH can provide an IRI to select one graph or use a variable which will range over the IRI of all the named graphs in the query's RDF dataset.

The use of GRAPH changes the active graph for matching graph patterns within that part of the query.

Outside the use of GRAPH, matching is done using the default graph.

Patterns on Named Graphs (cont.)

```
# Named graph: http://example.org/foaf/aliceFoaf
@prefix  foaf:     <http://xmlns.com/foaf/0.1/> .
@prefix  rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix  rdfs:     <http://www.w3.org/2000/01/rdf-schema#> .

_:a  foaf:name     "Alice" .
_:a  foaf:mbox     <mailto:alice@work.example> .
_:a  foaf:knows    _:b .

_:b  foaf:name     "Bob" .
_:b  foaf:mbox     <mailto:bob@work.example> .
_:b  foaf:nick     "Bobby" .
_:b  rdfs:seeAlso  <http://example.org/foaf/bobFoaf> .

<http://example.org/foaf/bobFoaf>
     rdf:type      foaf:PersonalProfileDocument .
```

```
# Named graph: http://example.org/foaf/bobFoaf
@prefix  foaf:     <http://xmlns.com/foaf/0.1/> .
@prefix  rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix  rdfs:     <http://www.w3.org/2000/01/rdf-schema#> .

_:z  foaf:mbox     <mailto:bob@work.example> .
_:z  rdfs:seeAlso  <http://example.org/foaf/bobFoaf> .
_:z  foaf:nick     "Robert" .

<http://example.org/foaf/bobFoaf>
     rdf:type      foaf:PersonalProfileDocument .
```

Patterns on Named Graphs (cont.)

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?src ?bobNick
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE
  {
    GRAPH ?src
    { ?x foaf:mbox <mailto:bob@work.example> .
      ?x foaf:nick ?bobNick
    }
  }
```

src	bobNick
<http://example.org/foaf/aliceFoaf>	"Bobby"
<http://example.org/foaf/bobFoaf>	"Robert"

Credits:
- https://www.w3.org/TR/sparql11-query/

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia

CSCE604131 | Jejaring Semantik (Knowledge Graphs)

Wikidata & DBpedia

Fariz Darari, Ph.D.

Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

Fadirra Talk S

Article Talk Read Edit View history More

Malang

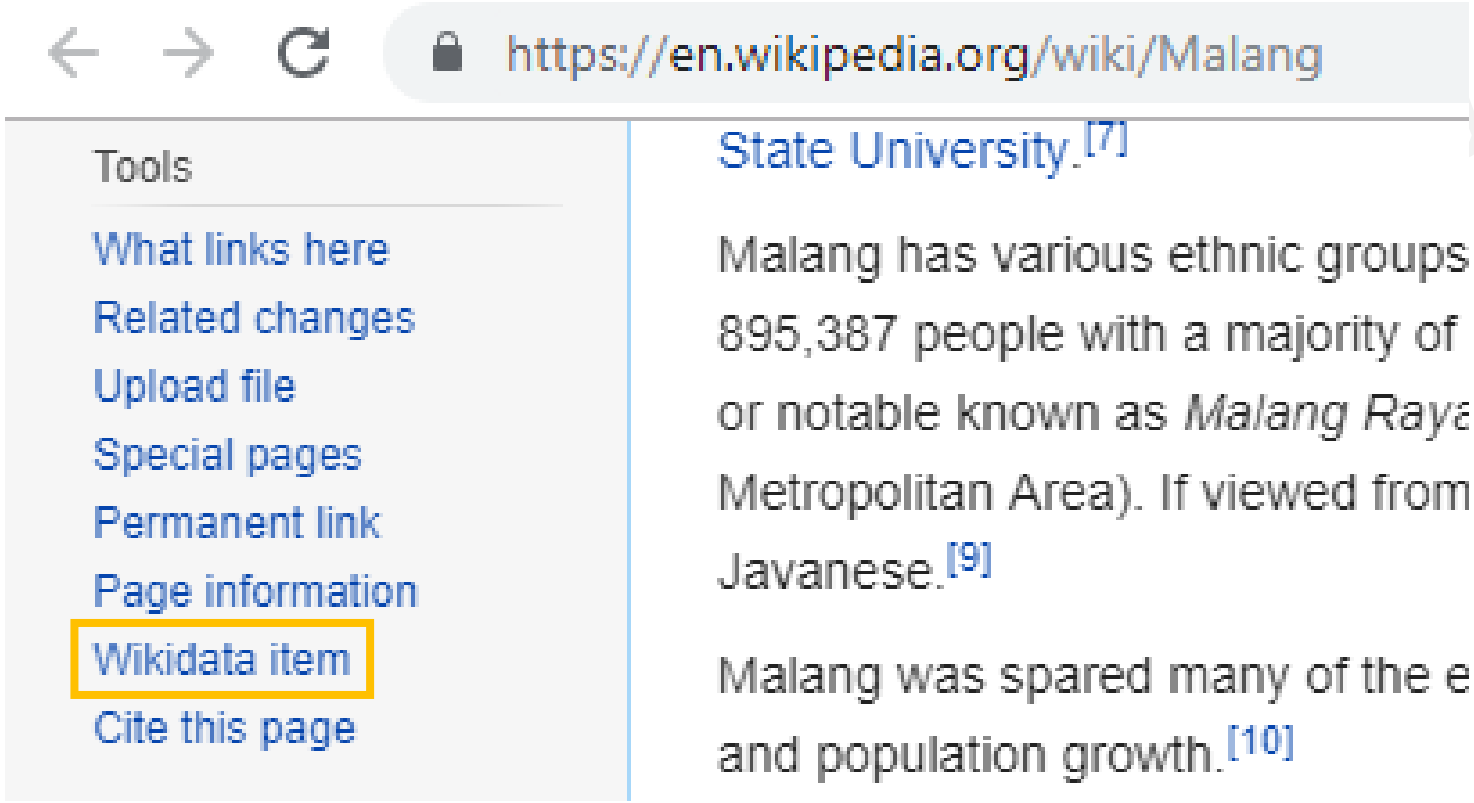
From Wikipedia, the free encyclopedia

This article is about the city. For other uses, see Malang (disambiguation).

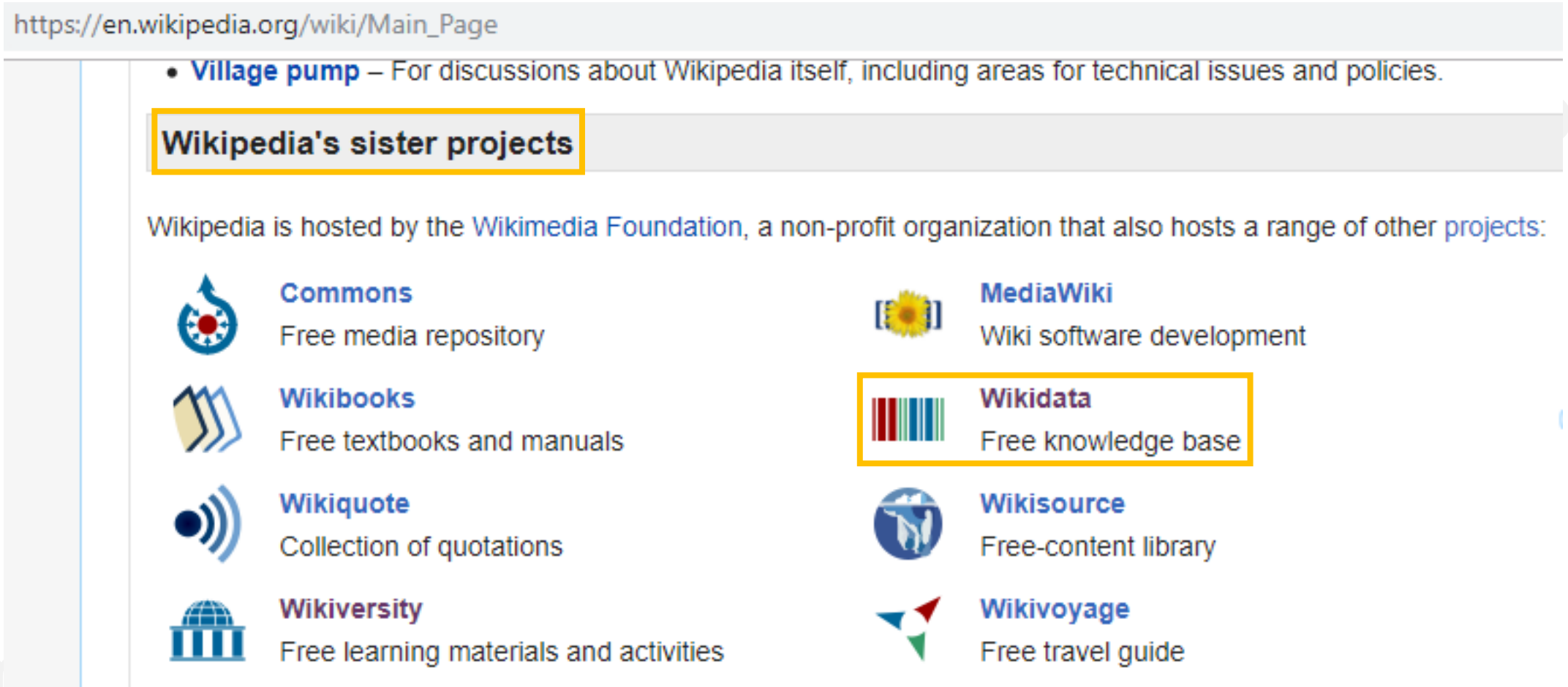
Malang (/.mɒˈlɒŋ-/; Javanese: ꦩꦭꦁ) is the second largest city in the Indonesian province of East Java. It has a history dating back to the age of Singhasari Kingdom. As the second most populous city in the province, the 2016 census recorded 887,443 people in the city.[3] Its metro area is home to 3,663,691 inhabitants spread across two cities and 22 districts (21 in Malang Regency and one in Pasuruan Regency).[4]

The city is well known for its mild climate. During the period of Dutch colonization, it was a popular destination for European residents. Until now, Malang still holds its position a popular destination for international tourists.[5] Malang keeps various historical relics. This city keeps relics of the Kingdom of Kanjuruhan period until the Dutch period.[6] Dutch heritage in general in the form of ancient buildings such as the Kayutangan church and Ijen cathedral which has gothic architecture. Malang also held various events to preserve its cultural heritage, one of them is the Malang Tempo Doeloe Festival. Malang also has a lot of historical heritage which has become a landmark like Tugu Malang (*Alun-alun Bundar*). Malang is also well known because it is labeled as an educational city. This city has one of the best universities in Indonesia such as Brawijaya University and Malang State University.[7]

Wikipedia (cont.)



Wikipedia (cont.)



Wikidata

Item | Discussion | Read | View history

Main page
Community portal
Project chat
Create a new Item
Create a new Lexeme
Recent changes
Random item
Query Service
Nearby
Help
Donate

Malang (Q11445)

city in East Java Province, Indonesia

▾ In more languages Configure

Language	Label	Description	Also known as
English	Malang	city in East Java Province, Indonesia	
Indonesian	Malang	kotamadya di provinsi Jawa Timur, Indonesia	Kota Malang
Javanese	Kutha Malang	kutha ing provinsi Jawa Wétan, Indonésia	
Sundanese	Kota Malang	No description defined	

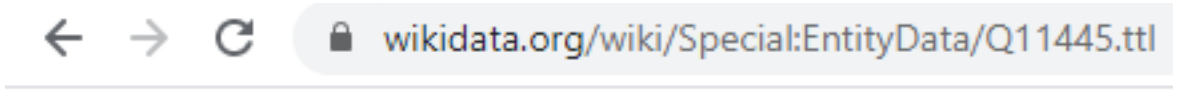
All entered languages

Wikidata (cont.)

Statements

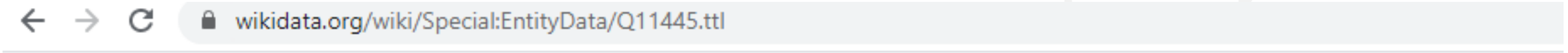
instance of	city of Indonesia 0 references	edit + add reference
	city 0 references	edit + add reference
		+ add value
image	 Skyline Malang Barat.jpg 729 × 123; 151 KB 1 reference	edit
		+ add value
country	Indonesia 1 reference	edit

Wikidata's RDF




```
wd:Q11445 rdfs:label "Malang"@en ;
        skos:prefLabel "Malang"@en ;
        schema:name "Malang"@en ;
        rdfs:label "Malang"@fr ;
        skos:prefLabel "Malang"@fr ;
        schema:name "Malang"@fr ;
        rdfs:label "Malang"@nb ;
        skos:prefLabel "Malang"@nb ;
        schema:name "Malang"@nb ;
        rdfs:label "Malang"@af ;
        skos:prefLabel "Malang"@af ;
        schema:name "Malang"@af ;
        rdfs:label "مالانغ"@ar ;
        skos:prefLabel "مالانغ"@ar ;
        schema:name "مالانغ"@ar ;
        rdfs:label "Маланг"@bg ;
        skos:prefLabel "Маланг"@bg ;
        schema:name "Маланг"@bg ;
        rdfs:label "Malang"@de ;
        skos:prefLabel "Malang"@de ;
        schema:name "Malang"@de ;
```

Wikidata's RDF (cont.)

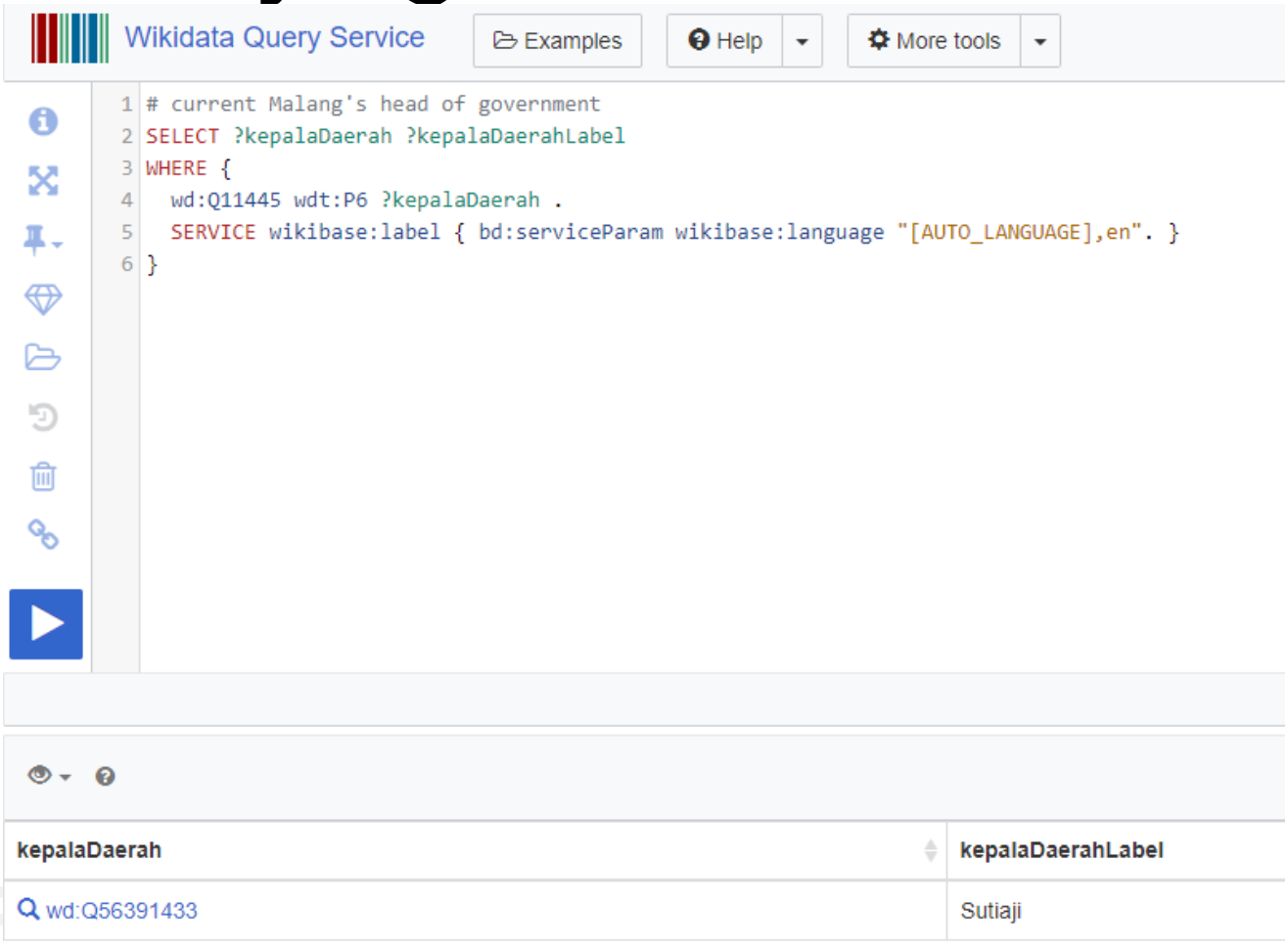



```
wdt:P373 "Malang" ;
wdt:P158 <http://commons.wikimedia.org/wiki/Special:FilePath/Seal%20of%20Malang%20City%20%28Logo%20Kota%20Malang%29.svg> ;
wdt:P17 wd:Q252 ;
wdt:P625 "Point(112.62 -7.98)"^^geo:wktLiteral ;
wdt:P131 wd:Q3586 ;
wdt:P910 wd:Q7830151 ;
wdt:P31 wd:Q3199141,
        wd:Q1549591 ;
```

. . . . .

Compiled by Fariz Darari

SPARQL Querying over Wikidata



Wikidata Key Features

- It is like Wikipedia but for data!
- In other words, it's the knowledge graph version of Wikipedia
- It is crowdsourced, anyone/anything can add data
- It is free, and can be used commercially
- As of 2021, it's got 1.2 billion facts about 92 million subjects! (Wikipedia "only" has 6.3 million subjects!)

Compiled by Fariz Darari

DBpedia

DBpedia Browse using Formats

About: Telkom Indonesia

An Entity of Type : Public company, from Named Graph : http://dbpedia.org, within Data Space :

PT Telekomunikasi Indonesia, Tbk, commonly abbreviated as Te telecommunications services company in Indonesia. Telkom is a multiple exchanges. Since this privatization in 1995, Telkom Indo customers at end of December 2011 increased by 7.8% from 20 telecommunication service provider by subscribers.

Property	Value
dbo:abstract	▪ PT Telekomunikasi Indonesia, Tbk, commonly abl services company in Indonesia. Telkom is a semi- Indonesia is the telecommunication business unit company of the Telkom Group, which is engaged property and finance services. Since 2008, Telkor organization and human resources, as well as the 1995, Telkom Indonesia has a combined total of a from 2010, making the company the nation's large
dbo:assets	▪ 1.1364E10
dbo:equity	▪ 5.24E9

DBpedia Key Features

DBpedia key features:

- It extracts data from Wikipedia infoboxes (summary box on top right corner).
- It is free, and can be used commercially
- It's got 9.5 **BILLION** facts about >6 million subjects!
- It conforms to the W3C's KG standards

DBpedia Indonesia

Content-Length: 263403

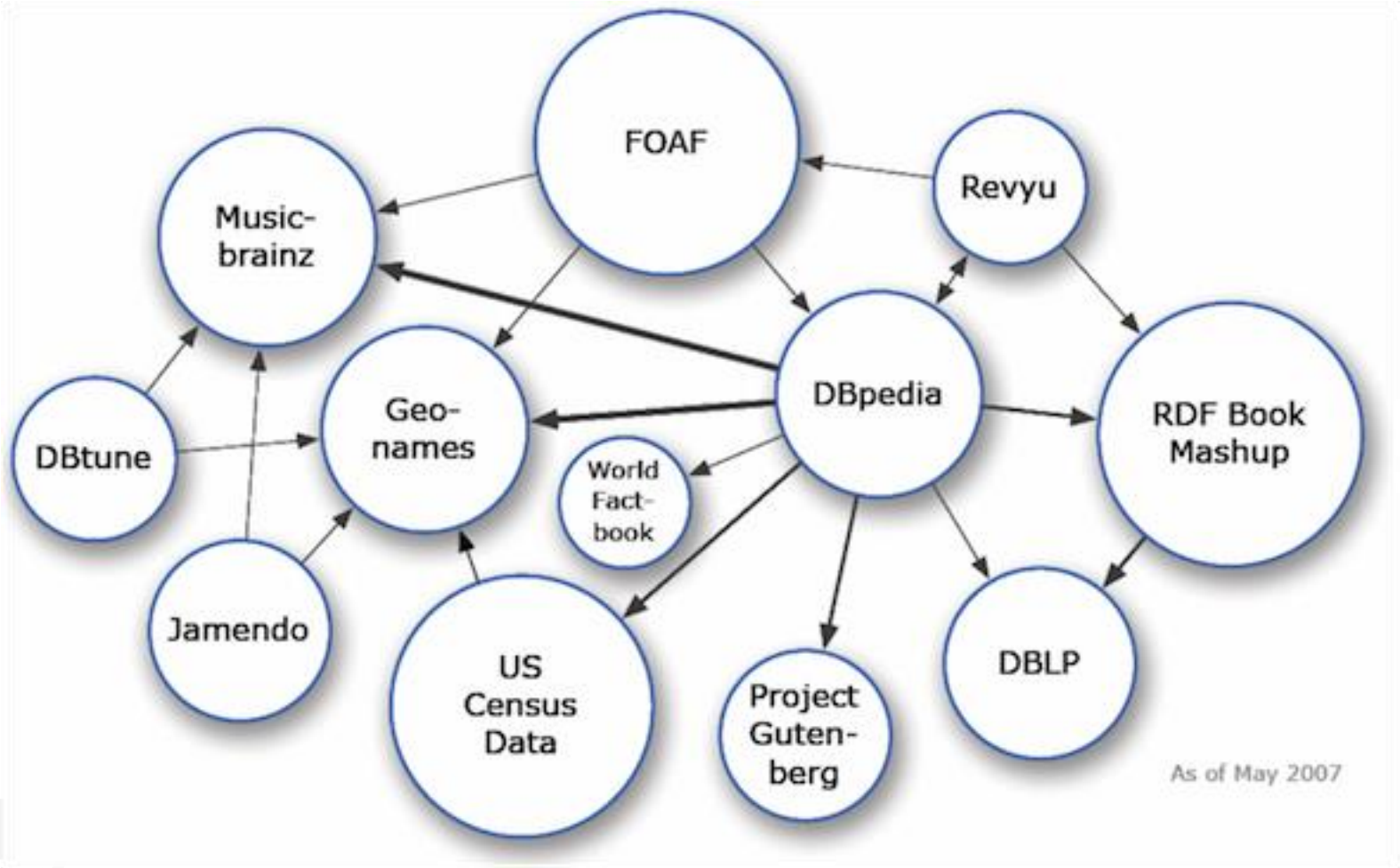
About: Universitas Indonesia

An Entity of Type : Perguruan tinggi negeri, from Named Graph : http://id.dbpedia.org, within Data Space : id.dbpedia.org

Universitas Indonesia, disingkat UI, adalah sebuah perguruan tinggi di Indonesia. Kampus utamanya terletak di bagian Utara dari Depok, Jawa Barat, dan kampus utama lainnya terdapat di daerah Salemba di Jakarta Pusat. UI secara umum dianggap sebagai salah satu dari tiga perguruan tinggi papan atas di Indonesia bersama dengan Universitas Gadjah Mada dan Institut Teknologi Bandung.

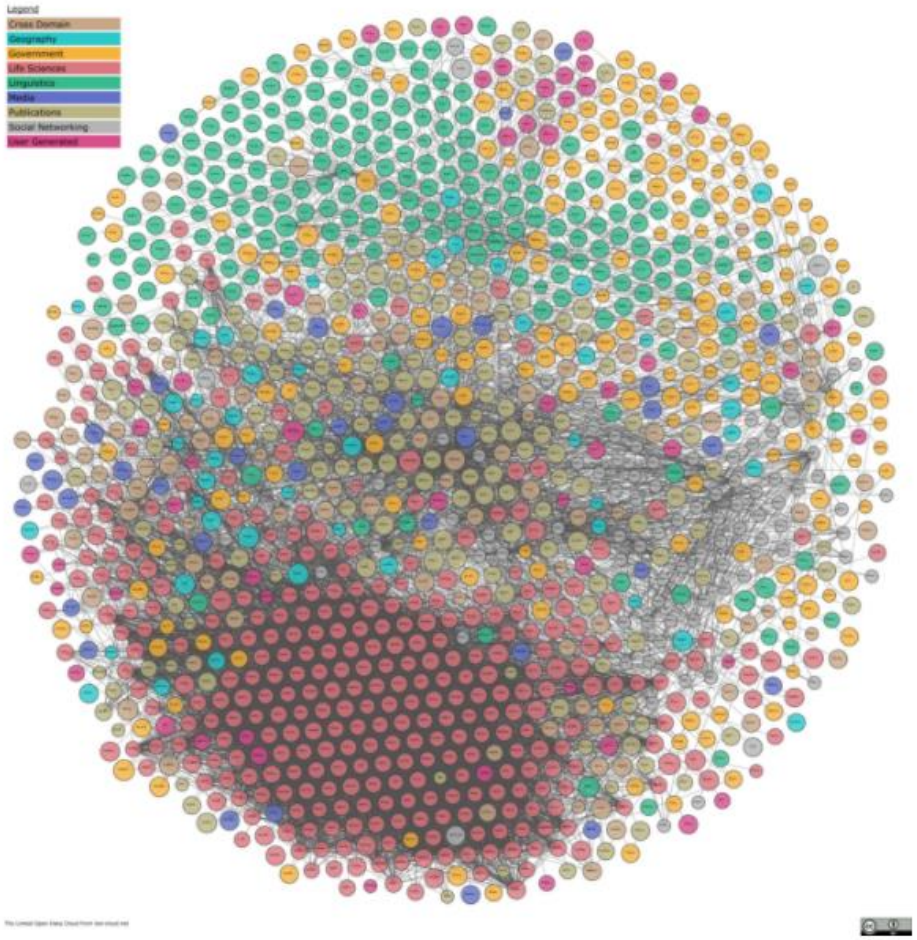
Property	Value
dbpedia-owl:abstract	▪ Universitas Indonesia, disingkat UI, adalah sebuah perguruan tinggi di Indonesia. Kampus uta terletak di bagian Utara dari Depok, Jawa Barat, dan kampus utama lainnya terdapat di daeral Jakarta Pusat. UI secara umum dianggap sebagai salah satu dari tiga perguruan tinggi papan Indonesia bersama dengan Universitas Gadjah Mada dan Institut Teknologi Bandung. ▪ Universitas Indonesia, disingkat UI, adalah sebuah perguruan tinggi di Indonesia. Kampus uta terletak di bagian Utara dari Depok, Jawa Barat, dan kampus utama lainnya terdapat di daeral Jakarta Pusat. UI secara umum dianggap sebagai salah satu dari tiga perguruan tinggi papan Indonesia bersama dengan Universitas Gadjah Mada dan Institut Teknologi Bandung.
dbpedia-owl:affiliation	▪ dbpedia-id:Association_of_Pacific_Rim_Universities ▪ dbpedia-id:Association_of_Pacific_Rim_Universities ▪ dbpedia-id:Association_of_Southeast_Asian_Institutions_of_Higher_Learning ▪ dbpedia-id:Association_of_Southeast_Asian_Institutions_of_Higher_Learning ▪ dbpedia-id:Jaringan_Universitas_ASEAN ▪ dbpedia-id:Jaringan_Universitas_ASEAN
dbpedia-owl:campus	▪ dbpedia-id:Area_urban ▪ dbpedia-id:Area_urban
dbpedia-owl:country	▪ dbpedia-id:Indonesia ▪ dbpedia-id:Indonesia
dbpedia-owl:locationCity	▪ dbpedia-id:Kota_Depok ▪ dbpedia-id:Kota_Depok

KGs from time to time: 2007



http://lod-cloud.net/

KGs from time to time: 2021



http://lod-cloud.net/

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia

CSCE604131 | Jejaring Semantik (Knowledge Graphs)

OWL

Fariz Darari, Ph.D.

OWL: The Schema Part (of KGs)

Short for: Web Ontology Language (WOL? No, it is OWL!)

Key features:
- Vocabulary description
For example: To say that prop:knows can be used to link between two things
- Reasoning: You can conclude new things based on existing facts!

For example: owl subClassOf bird + bird subClassOf animal
Now, if Bobi is an owl, do you think Bobi is an animal?

Yes!

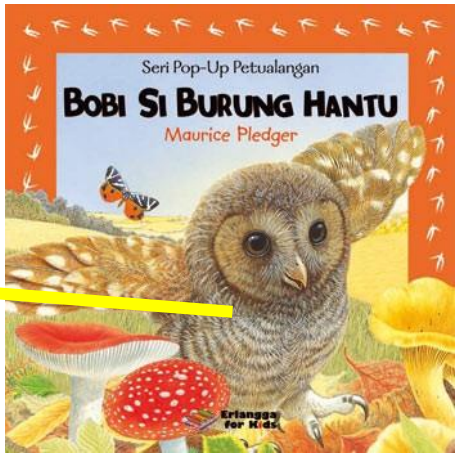


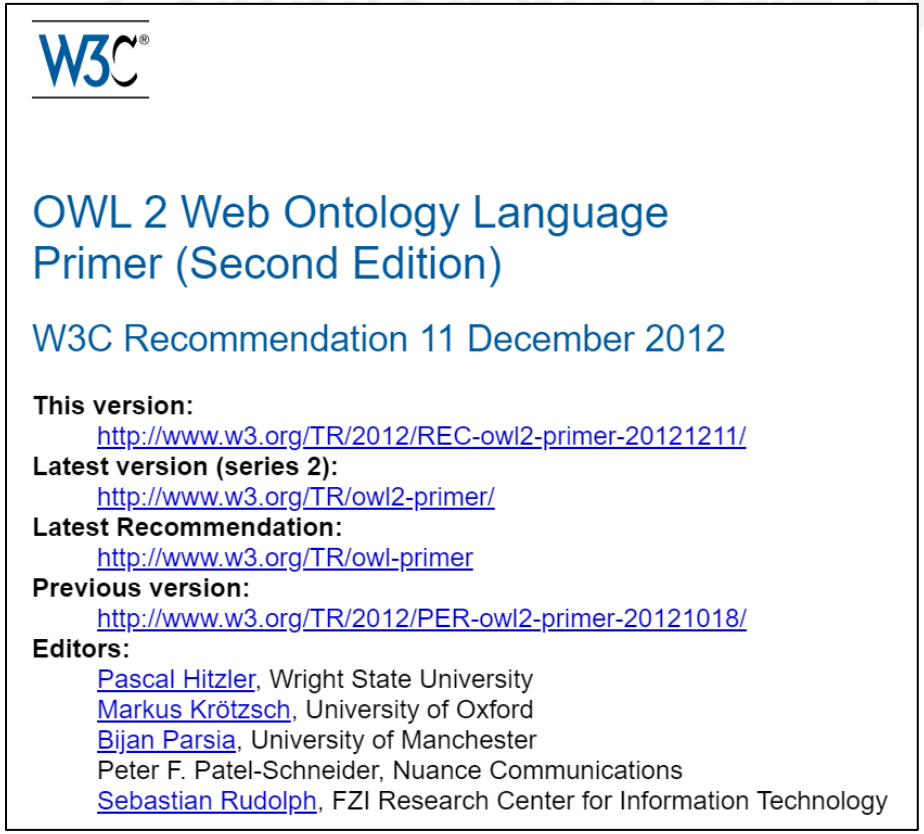
Image: http://www.bukabuku.com/browses/product/9789797811211/pop-upbobi-si-burung-hantu.html

OWL Characteristics

- OWL is a language to create ontologies
- OWL is more expressive/heavyweight compared to RDFS
- Ontology: Explicit specification of a conceptualization
- So ontology specifies concepts, relations, and other characterizations pertaining to domain modeling
- Such a specification can be called **semantics**

OWL Characteristics (cont.)

- OWL enables inferencing over data (+ schema)
- Example inferences: Consistency checking and hierarchy reasoning
- OWL 2 is standardized in 2012
- Specification by W3C can be found at: https://www.w3.org/TR/owl-primer/



W3C

OWL 2 Web Ontology Language Primer (Second Edition)

W3C Recommendation 11 December 2012

This version:
http://www.w3.org/TR/2012/REC-owl2-primer-20121211/
Latest version (series 2):
http://www.w3.org/TR/owl2-primer/
Latest Recommendation:
http://www.w3.org/TR/owl-primer
Previous version:
http://www.w3.org/TR/2012/PER-owl2-primer-20121018/
Editors:
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 Technology

Basic OWL Features

- Class and Instance
  ```
  :Mary rdf:type :Woman .
  ```

- Class Hierarchy
  ```
  :Woman rdfs:subClassOf :Person .
  ```

- Class Disjointness
  ```
  :Woman owl:disjointWith :Man.
  ```

Basic OWL Features (cont.)

- Object Property
```
:John :hasWife :Mary .
```


- Datatype Property
```
:John :hasAge "18"^^xsd:integer .
```

- Hierarchy Property
```
:hasWife rdfs:subPropertyOf :hasSpouse .
```

- Instance Equality and Difference
```
:John owl:differentFrom :Bill .
:James owl:sameAs :Jim .
```

Complex OWL Features

- Complex classes
  Mother is equivalent to the intersection between Woman and Parent
- Property restrictions
  Class Parent is a class from instances relating to a Person through the property hasChild
- Property cardinality restrictions
  Human has at most two parents
- Inverse properties
  hasParent is an inverse from hasChild
- Property chains
  Two hasParent sequence make up property hasGrandparent
- Keys
  Two instances are the same if having the same SSNs

List of Ontologies

- Schema.org: https://schema.org/
- FOAF (Friend-of-a-Friend): http://xmlns.com/foaf/spec/
- vCard: http://www.w3.org/2006/vcard/ns#
- Dublin Core: http://purl.org/dc/terms/
- PROV: http://www.w3.org/ns/prov#

Compiled by Fariz Darari

Ontology Development 101

1. Determine the domain and scope of the ontology
2. Consider reusing existing ontologies
3. Enumerate important terms in the ontology
4. Define the classes and the class hierarchy
5. Define the properties of classes—slots
6. Define the facets of the slots
7. Create instances

https://protege.stanford.edu/publications/ontology_development/ontology101.pdf

Credits:

- W3C specifications
- Semantic Web & Knowledge Graph literature

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia

CSCE604131 | Jejaring Semantik (Knowledge Graphs)

SHACL

Fariz Darari, Ph.D.

SHACL

- SHACL: SHApe Constraint Language
- SHACL is a language to define shape and constraint rules for RDF data
- SHACL is a W3C Recommendation since July 2017
- SHACL is useful for describing and validating RDF data
- SHACL specification is available at: https://www.w3.org/TR/shacl/

Compiled by Fariz Darari

Shapes Constraint Language (SHACL)

W3C Recommendation 20 July 2017

This version:

https://www.w3.org/TR/2017/REC-shacl-20170720/

Latest published version:

https://www.w3.org/TR/shacl/

Latest editor's draft:

https://w3c.github.io/data-shapes/shacl/

Implementation report:

https://w3c.github.io/data-shapes/data-shapes-test-suite/

Previous version:

https://www.w3.org/TR/2017/PR-shacl-20170608/

Editors:

Holger Knublauch, TopQuadrant, Inc.

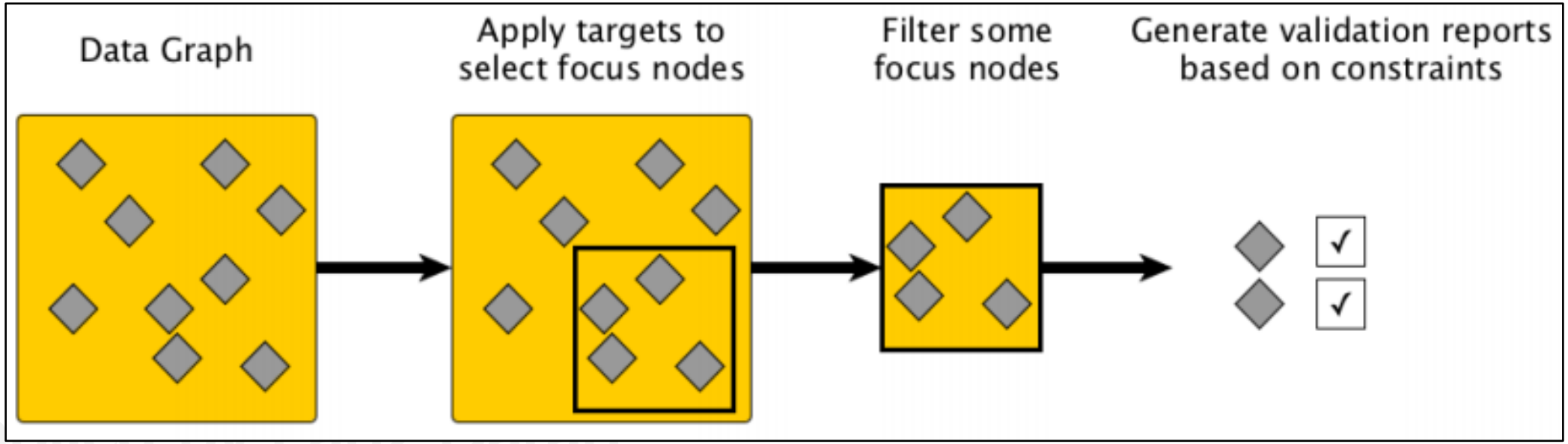
Dimitris Kontokostas, University of Leipzig

SHACL Terms

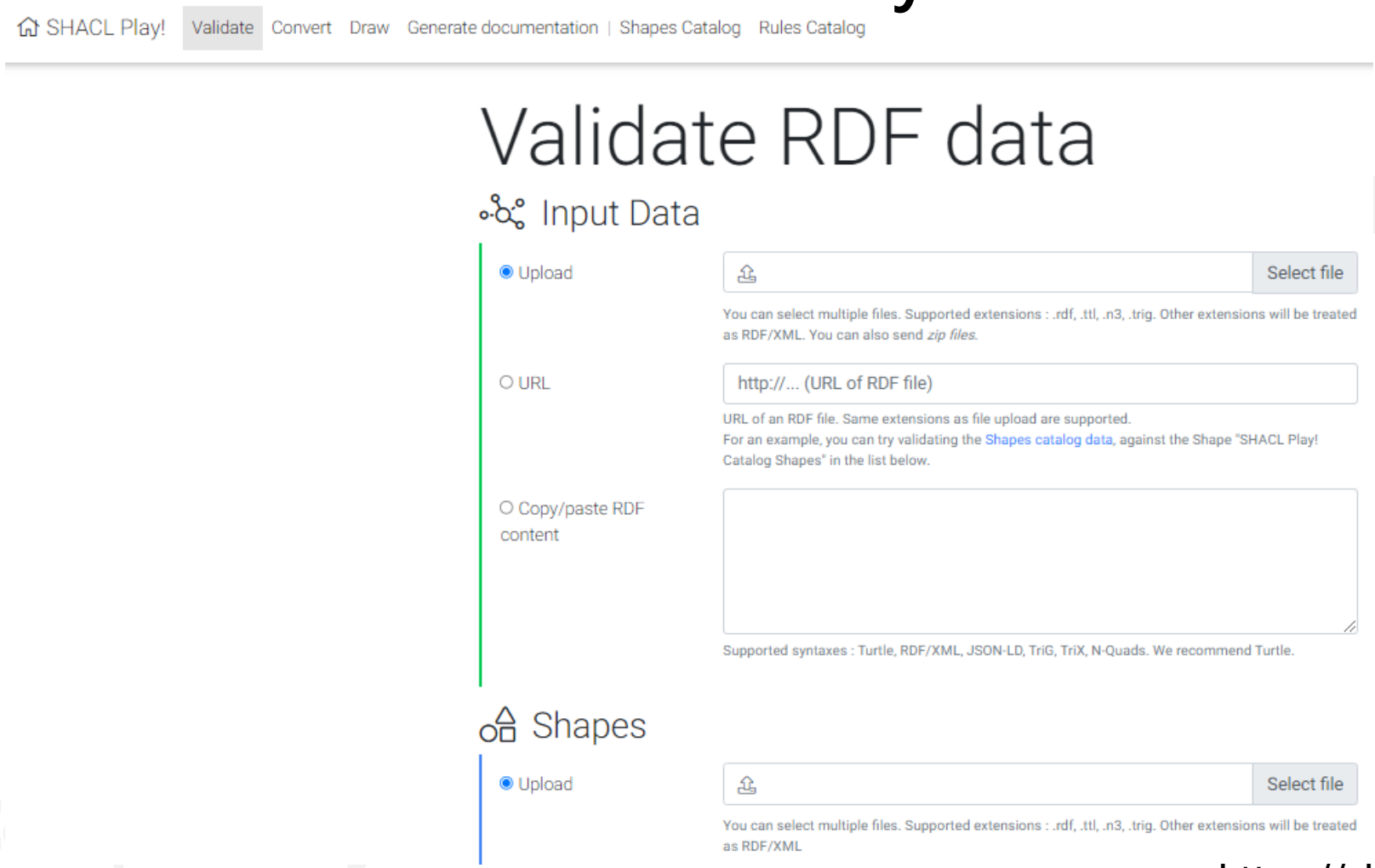
- SHACL is for validating RDF KGs against a set of conditions.
- These conditions are called shapes.
- So, in SHACL there are two types of graphs:
  - > **Shapes graphs:** to validate
  - > **Data graphs:** to be validated
- SHACL usages: validation, user interface building, code generation and data integration.

SHACL Shape

- A SHACL shape has two main components: target and constraint
- Target specifies which part of data has to conform to a shape
- Constraint determines how a target can be validated



SHACL Tool: SHACL Play!



https://shacl-play.sparna.fr/play/validate

SHACL Shape: Color of Among Us characters is Pink or Purple

```
@prefix ex: <http://example.org/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .

ex:MyShape
  a sh:NodeShape ;
  sh:targetClass ex:AmongUsCharacter ;
  sh:property [
    sh:path ex:color ;
    sh:in ( ex:Pink ex:Purple )
  ] .
```

SHACL Shape: Color of Among Us characters is Pink or Purple - Erroneous Data

```
@prefix ex: <http://example.org/> .

ex:Tom a ex:AmongUsCharacter ;
       ex:color ex:Green .
```

Validation results of 1 shapes

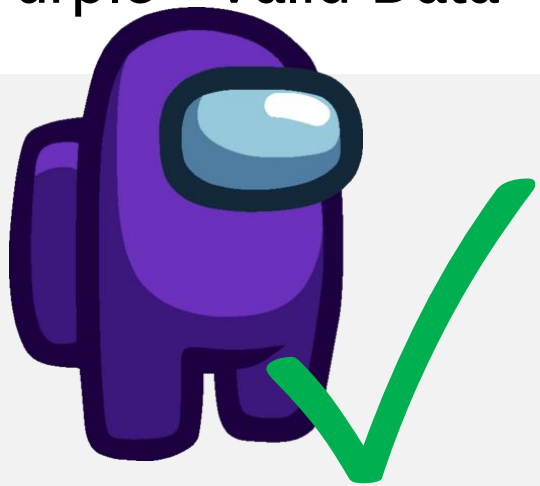
Download validation report in CSV Turtle RDF/XML

1 Violation

Violation 1 *"Not a value from the sh:in enumeration"* (ex:color) see details

SHACL Shape: Color of Among Us characters is Pink or Purple - Valid Data

```
@prefix ex: <http://example.org/> .

ex:Tom a ex:AmongUsCharacter ;
       ex:color ex:Purple .
```



Validation results of 1 shapes

Download validation report in CSV Turtle RDF/XML

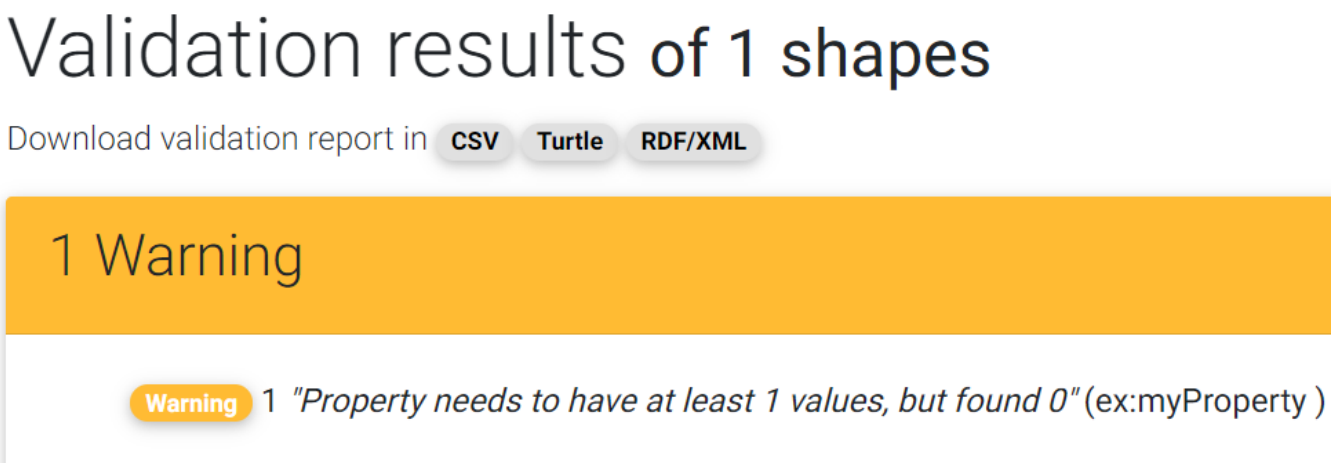
Valid Data is conformant !

SHACL Shape: Property cardinality and value (string) length

```
@prefix ex: <http://example.org/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:MyShape
        a sh:NodeShape ;
        sh:targetNode ex:MyInstance ;
        sh:property [
                sh:path ex:myProperty ;
                sh:minCount 1 ;
                sh:datatype xsd:string ;
                sh:severity sh:Warning ; # warning-level constraint
        ] ;
        sh:property [
                # the default severity here is sh:Violation
                sh:path ex:myProperty ;
                sh:maxLength 10 ;
                sh:message "Too many characters"@en ;
        ] .
```

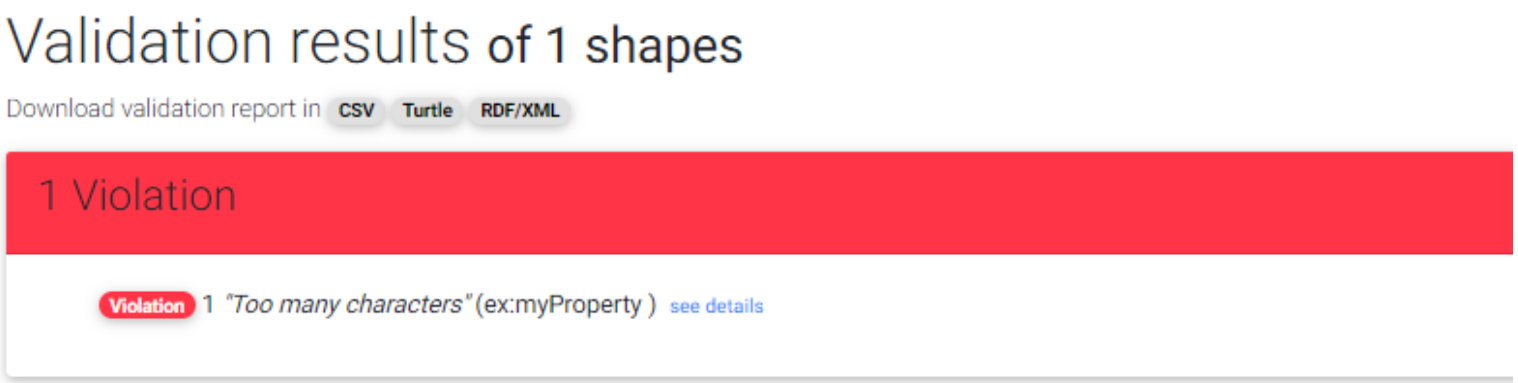
SHACL Shape: Property cardinality and value (string) length - Erroneous Data

```
@prefix ex: <http://example.org/> .

ex:MyInstance
     ex:someProperty "morethan10chars" .
```



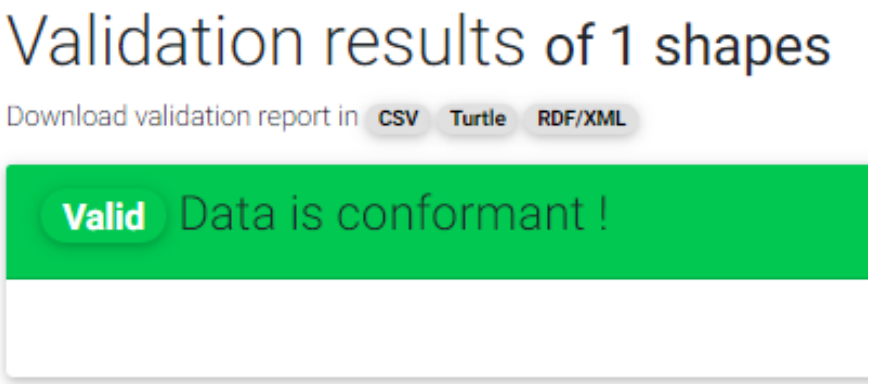
SHACL Shape: Property cardinality and value (string) length - Erroneous Data

```
@prefix ex: <http://example.org/> .

ex:MyInstance
     ex:myProperty "morethan10chars" .
```



SHACL Shape: Property cardinality and value (string) length - Valid Data

```
@prefix ex: <http://example.org/> .

ex:MyInstance
     ex:myProperty "bob" .
```



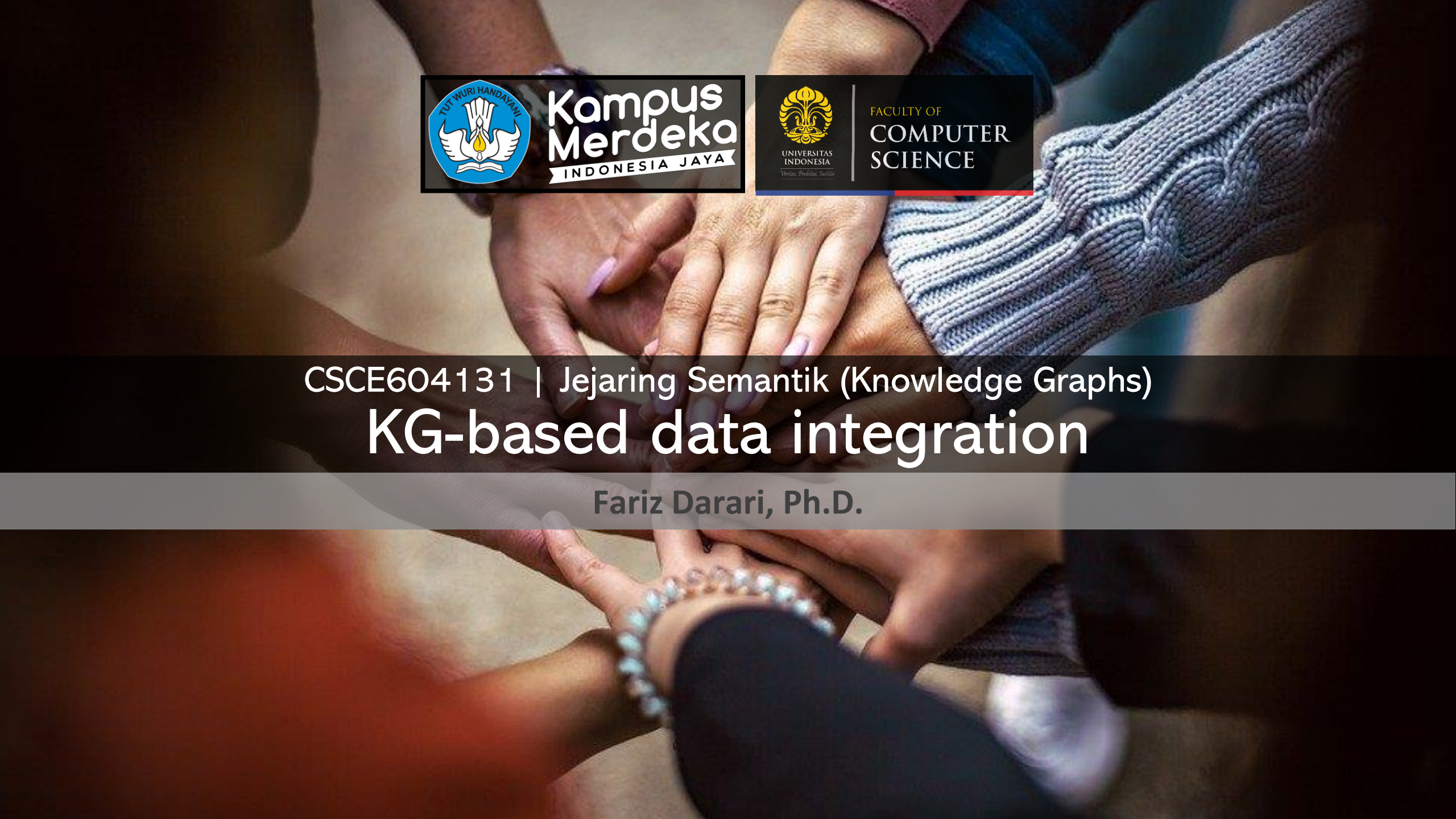
SHACL Core Constraint Components

Operation	*Parameters*
Cardinality constraints	sh:minCount, sh:maxCount
Value types	sh:class, sh:datatype, sh:nodeKind sh:in, sh:hasValue
Value range constraints	sh:minInclusive, sh:maxInclusive sh:minExclusive, sh:maxExclusive
String based constraints	sh:minLength, sh:maxLength sh:length sh:pattern
Language based	sh:uniqueLang, sh:languageIn
Logical constraints	sh:and, sh:or, sh:xone, sh:not
Shape-based constraints	sh:node, sh:property sh:qualifiedValueShape, sh:qualifiedValueShapesDisjoint sh:qualifiedMinCount sh:qualifiedMaxCount
Closed shapes	sh:closed, sh:ignoredProperties
Property pair constraints	sh:equals, sh:disjoint sh:lesThan, sh:lessThanOrEquals
Non-validating constraints	sh:name, sh:description, sh:order, sh:group

Credits:

- https://book.validatingrdf.com/bookHtml011.html
- https://www.ida.liu.se/~robke04/SHACLTutorial/Introduction%20to%20SHACL.pdf
- https://www.w3.org/TR/shacl/
- https://among-us.fandom.com/

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia



CSCE604131 | Jejaring Semantik (Knowledge Graphs)

KG-based data integration

Fariz Darari, Ph.D.

Data integration

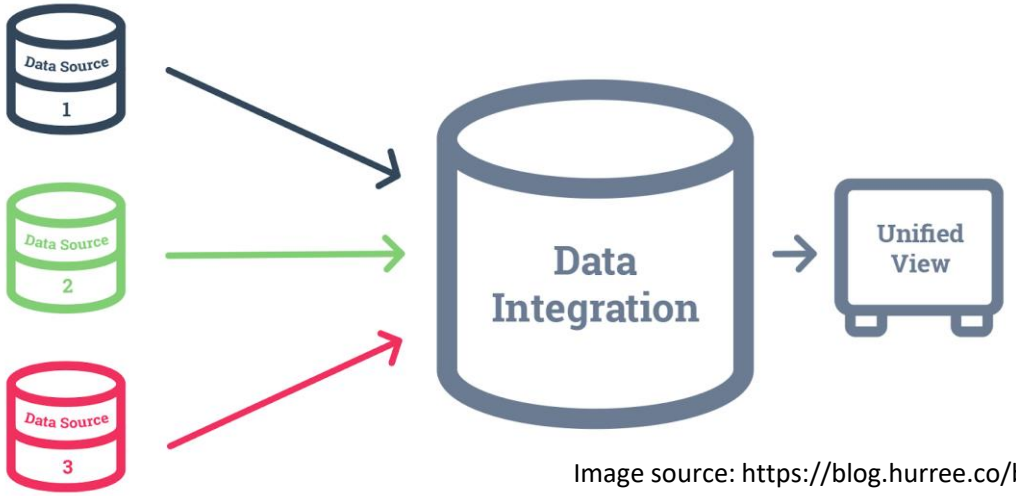


Image source: https://blog.hurree.co/blog/data-integration-in-marketing

- Ideal: All datasets use one schema only
- Reality: Datasets are created independently, hence (potentially) different schemas
- The goal of data integration: Tie together different sources, different source schemas, under a common, unifying schema

R2RML

- R2RML: RDB to RDF Mapping Language
- A language for expressing customized mappings from relational databases to RDF datasets
- R2RML provides the ability to view existing relational data in the RDF data model, expressed in a structure and target vocabulary of the mapping author's choice
- W3C Recommendation since 2012: https://www.w3.org/TR/r2rml/

Compiled by Fariz Darari

R2RML Mapping Implementations

- A virtual SPARQL endpoint over mapped relational data
- Generate RDF dumps
- Offer a Linked Data interface

Direct Mapping

- A strategy for mapping relational data to RDF
- Direct mapping defines a simple transformation, providing a basis for defining and comparing more intricate transformations
- Can be used to materialize RDF graphs or define virtual graphs
- Difference to R2RML is that R2RML is more customized

Direct Mapping Example: SQL DDL

```
CREATE TABLE "Addresses" (
        "ID" INT, PRIMARY KEY("ID"),
        "city" CHAR(10),
        "state" CHAR(2)
)

CREATE TABLE "People" (
        "ID" INT, PRIMARY KEY("ID"),
        "fname" CHAR(10),
        "addr" INT,
        FOREIGN KEY("addr") REFERENCES "Addresses"("ID")
)

INSERT INTO "Addresses" ("ID", "city", "state") VALUES (18, 'Cambridge', 'MA')
INSERT INTO "People" ("ID", "fname", "addr") VALUES (7, 'Bob', 18)
INSERT INTO "People" ("ID", "fname", "addr") VALUES (8, 'Sue', NULL)
```

People

PK		→ *Address(ID)*
ID	**fname**	**addr**
7	Bob	18
8	Sue	**NULL**

Addresses

PK		
ID	**city**	**state**
18	Cambridge	MA

Direct Mapping Example: Generated RDF

```
@base <http://foo.example/DB/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .


<People/ID=7> rdf:type <People> .
<People/ID=7> <People#ID> 7 .
<People/ID=7> <People#fname> "Bob" .
<People/ID=7> <People#addr> 18 .
<People/ID=7> <People#ref-addr> <Addresses/ID=18> .
<People/ID=8> rdf:type <People> .
<People/ID=8> <People#ID> 8 .
<People/ID=8> <People#fname> "Sue" .

<Addresses/ID=18> rdf:type <Addresses> .
<Addresses/ID=18> <Addresses#ID> 18 .
<Addresses/ID=18> <Addresses#city> "Cambridge" .
<Addresses/ID=18> <Addresses#state> "MA" .
```

People

PK		→ *Address(ID)*
ID	**fname**	**addr**
7	Bob	18
8	Sue	**NULL**

Addresses

PK		
ID	**city**	**state**
18	Cambridge	MA

R2RML: Overview

An R2RML mapping refers to logical tables to retrieve data from the input database.
A logical table can be one of the following:

- A base table,
- a view, or
- a valid SQL query (called an “R2RML view” because it emulates a SQL view without modifying the database).

R2RML: Overview

Each logical table is mapped to RDF using a **triples map**. The triples map is a rule that maps each row in the logical table to a number of RDF triples. The rule has two main parts:

1. A **subject map** that generates the subject of all RDF triples that will be generated from a logical table row. The subjects often are IRIs that are generated from the primary key column(s) of the table.
2. Multiple **predicate-object maps** that in turn consist of predicate maps and object maps (or referencing object maps).

R2RML: Overview

Triples are produced by combining the subject map with a predicate map and object map, and applying these three to each logical table row. For example, the complete rule for generating a set of triples might be:

- Subjects: A template `http://data.example.com/employee/{empno}` is used to generate subject IRIs from the `empno` column.
- Predicates: The constant vocabulary IRI `ex:name` is used.
- Objects: The value of the `ename` column is used to produce an RDF literal.

R2RML: Example Input DB

EMP

EMPNO INTEGER PRIMARY KEY	ENAME VARCHAR(100)	JOB VARCHAR(20)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

DEPT

DEPTNO INTEGER PRIMARY KEY	DNAME VARCHAR(30)	LOC VARCHAR(100)
10	APPSERVER	NEW YORK

R2RML: Example Desired RDF Output

Example output data

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department <http://data.example.com/department/10>.

<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
<http://data.example.com/department/10> ex:staff 1.
```

R2RML: Mapping a Simple Table

Example R2RML mapping

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "EMP" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{EMPNO}";
        rr:class ex:Employee;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "ENAME" ];
    ].
```

EMP

EMPNO INTEGER PRIMARY KEY	ENAME VARCHAR(100)	JOB VARCHAR(20)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

DEPT

DEPTNO INTEGER PRIMARY KEY	DNAME VARCHAR(30)	LOC VARCHAR(100)
10	APPSERVER	NEW YORK

R2RML: Mapping a Simple Table - Output

EMP

EMPNO INTEGER PRIMARY KEY	ENAME VARCHAR(100)	JOB VARCHAR(20)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

DEPT

DEPTNO INTEGER PRIMARY KEY	DNAME VARCHAR(30)	LOC VARCHAR(100)
10	APPSERVER	NEW YORK

Example output data

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
```

R2RML: Computing a Property with an R2RML View

Example SQL query

```
<#DeptTableView> rr:sqlQuery """
SELECT DEPTNO,
       DNAME,
       LOC,
       (SELECT COUNT(*) FROM EMP WHERE EMP.DEPTNO=DEPT.DEPTNO) AS STAFF
FROM DEPT;
""".
```

R2RML: Computing a Property with an R2RML View

Example R2RML mapping

```
<#TriplesMap2>
    rr:logicalTable <#DeptTableView>;
    rr:subjectMap [
        rr:template "http://data.example.com/department/{DEPTNO}";
        rr:class ex:Department;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "DNAME" ];
    ];
    rr:predicateObjectMap [
        rr:predicate ex:location;
        rr:objectMap [ rr:column "LOC" ];
    ];
    rr:predicateObjectMap [
        rr:predicate ex:staff;
        rr:objectMap [ rr:column "STAFF" ];
    ].
```

R2RML: Computing a Property with an R2RML View - Data

EMP

EMPNO INTEGER PRIMARY KEY	ENAME VARCHAR(100)	JOB VARCHAR(20)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

DEPT

DEPTNO INTEGER PRIMARY KEY	DNAME VARCHAR(30)	LOC VARCHAR(100)
10	APPSERVER	NEW YORK

Example output data

```
<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
<http://data.example.com/department/10> ex:staff 1.
```

R2RML: Linking Two Tables

Example R2RML mapping

```
<#TriplesMap1>
    rr:predicateObjectMap [
        rr:predicate ex:department;
        rr:objectMap [
            rr:parentTriplesMap <#TriplesMap2>;
            rr:joinCondition [
                rr:child "DEPTNO";
                rr:parent "DEPTNO";
            ];
        ];
    ].
```

R2RML: Linking Two Tables - Data

EMP

EMPNO INTEGER PRIMARY KEY	ENAME VARCHAR(100)	JOB VARCHAR(20)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

DEPT

DEPTNO INTEGER PRIMARY KEY	DNAME VARCHAR(30)	LOC VARCHAR(100)
10	APPSERVER	NEW YORK

Example output data

```
<http://data.example.com/employee/7369> ex:department <http://data.example.com/department/10>.
```

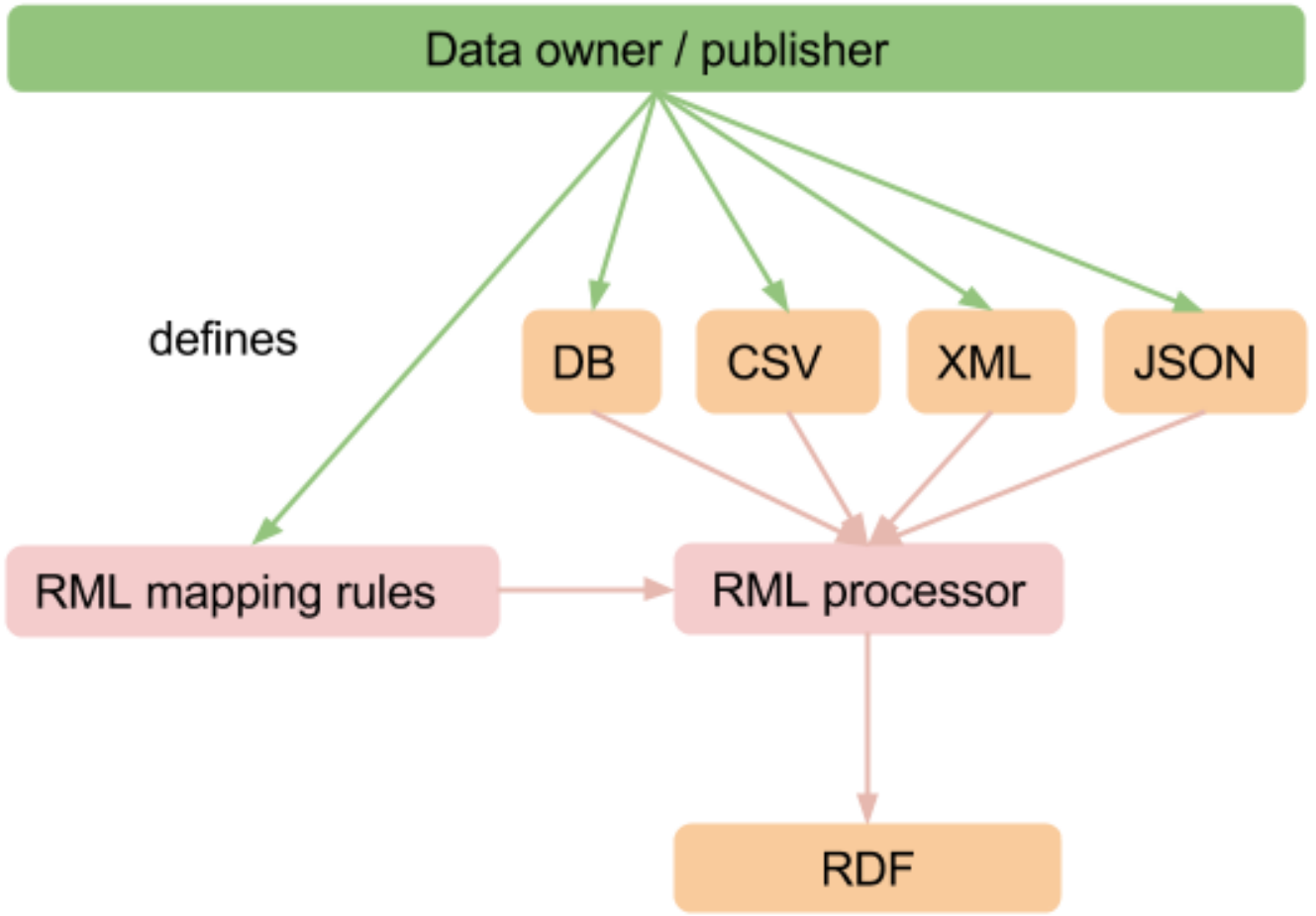
RML

- RML: RDF Mapping Language
- A generic mapping language based on and extending R2RML
- Express customized mapping rules from heterogeneous data structures and serializations to the RDF data model
- RML specification is available: https://rml.io/specs/rml/

Compiled by Fariz Darari

RML: Features

- Generate knowledge graphs
- Create declarative rules
- Use any type of semi-structured data
- Guarantee high-quality KGs



Credits:

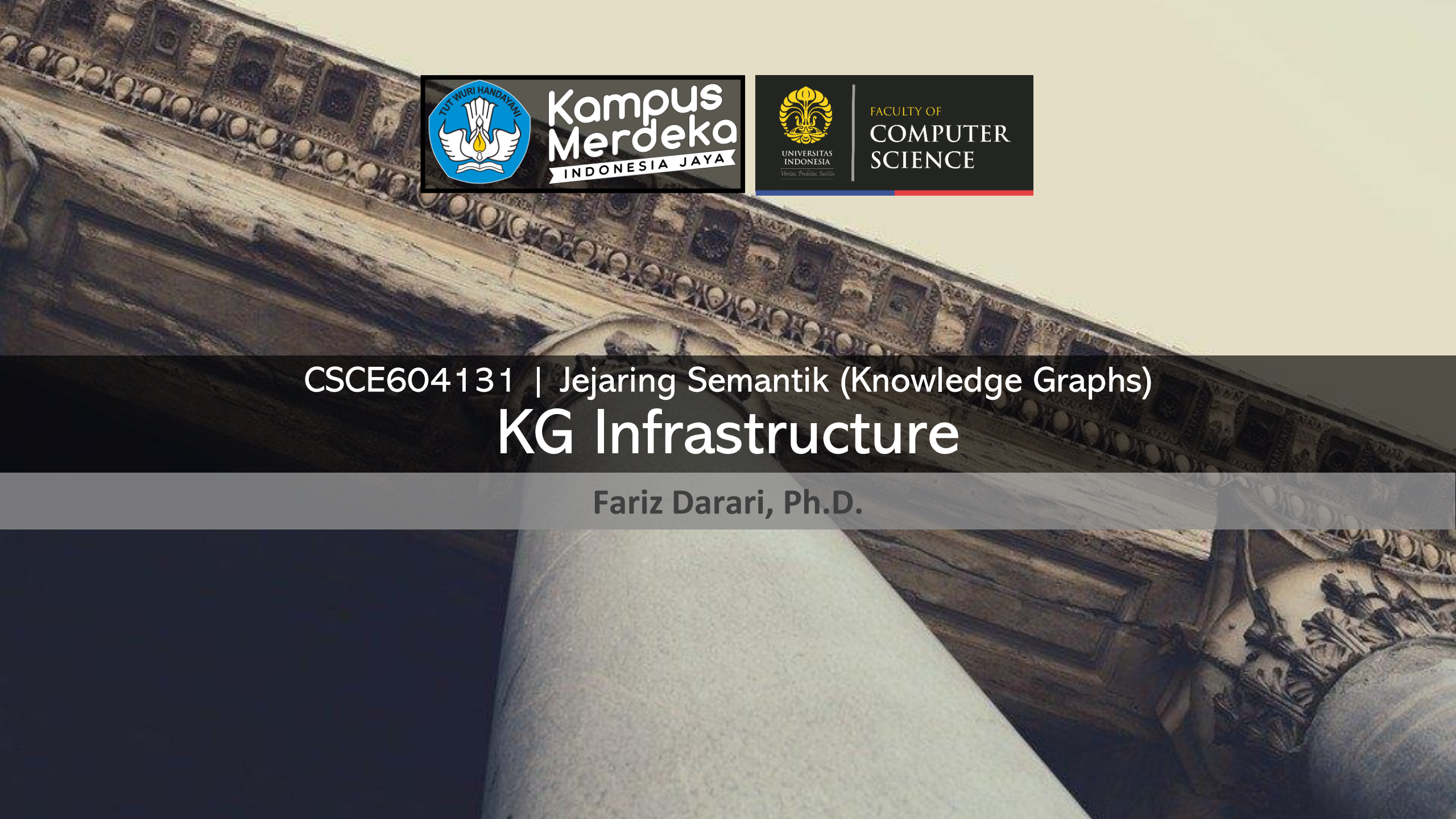
https://www.w3.org/TR/r2rml/

https://www.w3.org/TR/rdb-direct-mapping/

https://research.cs.wisc.edu/dibook/

https://rml.io/specs/rml/

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia

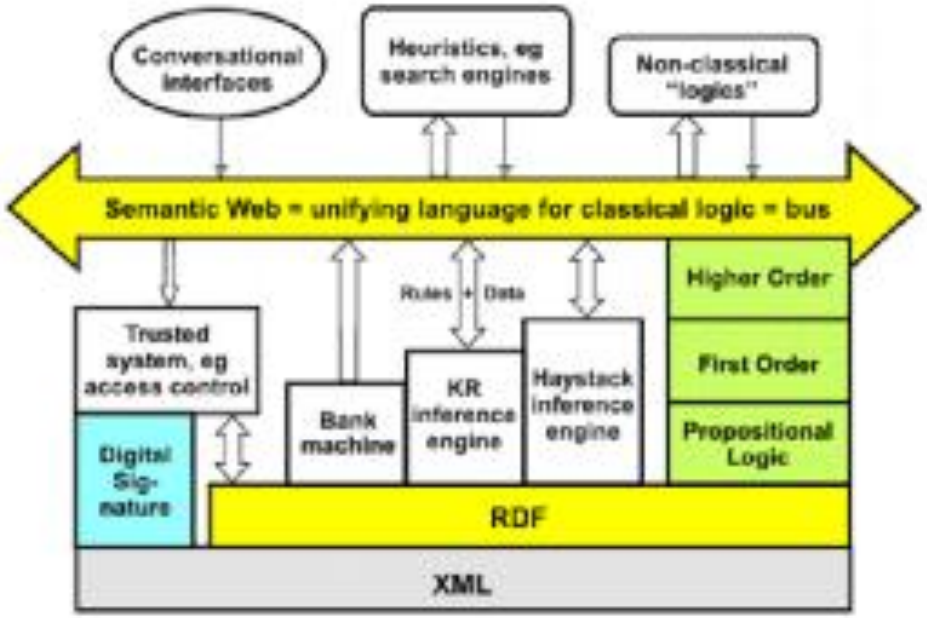


CSCE604131 | Jejaring Semantik (Knowledge Graphs)

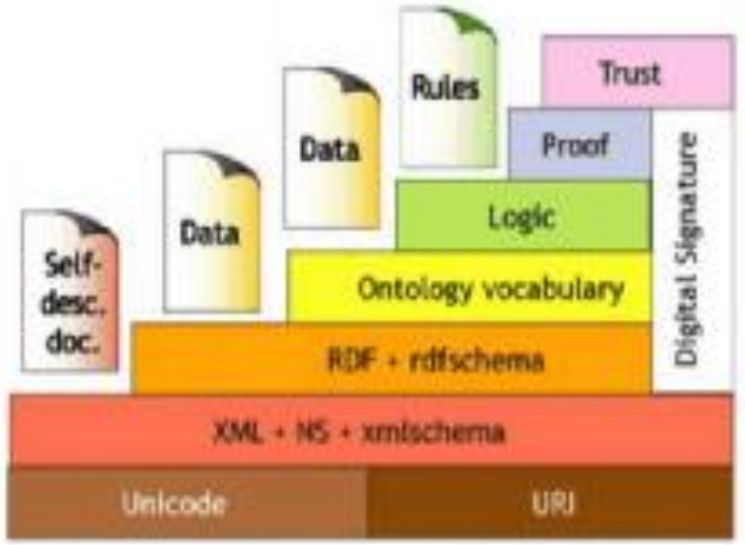
KG Infrastructure

Fariz Darari, Ph.D.

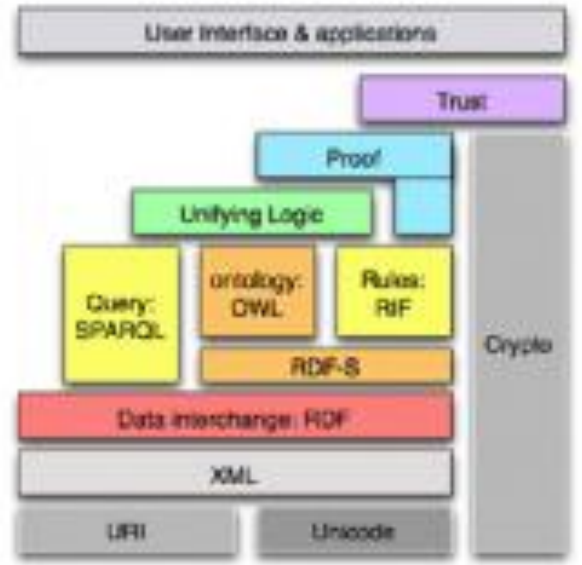
KG Stack from 2000-2005



2000



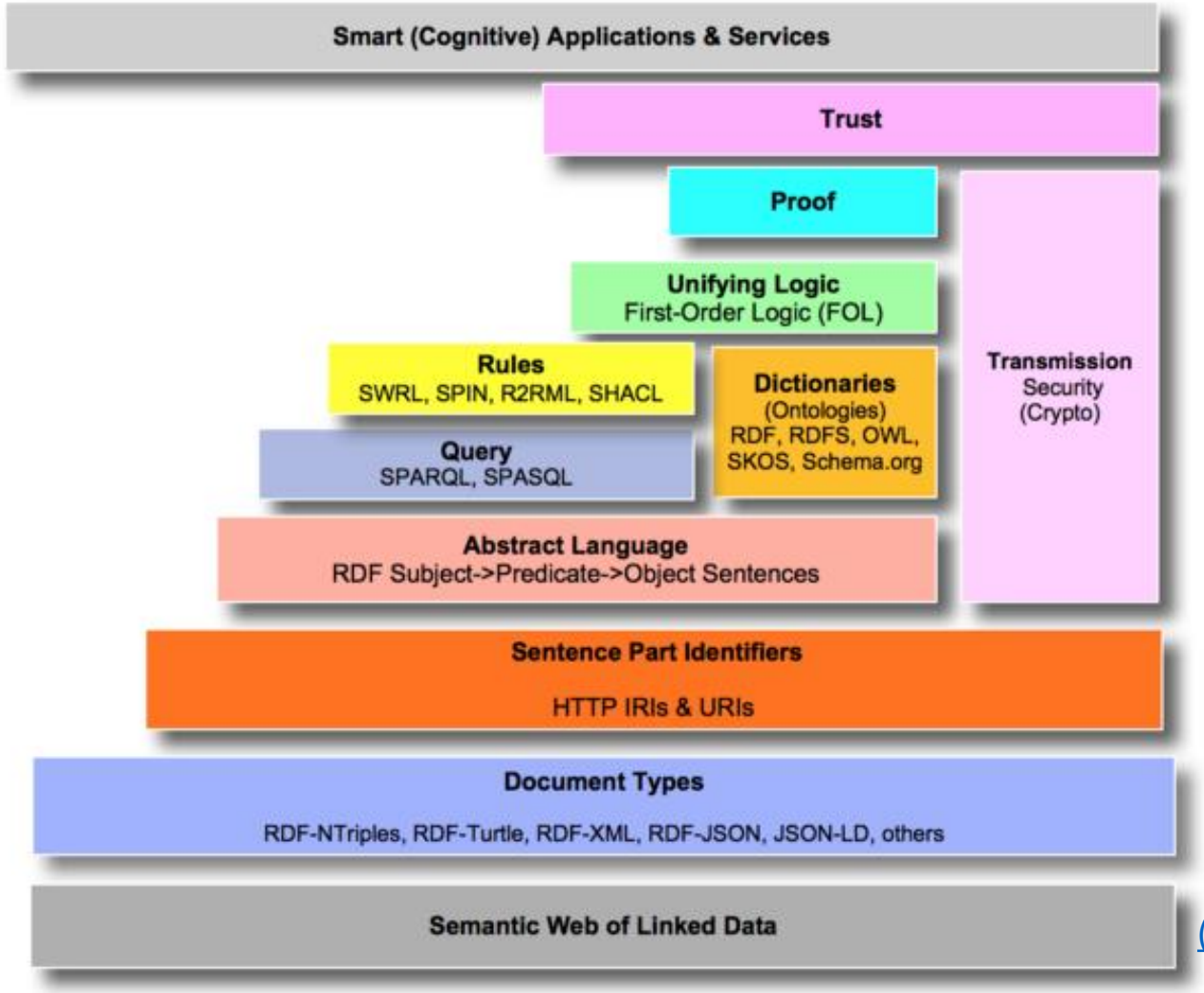
2001



2005

Source: W3C history

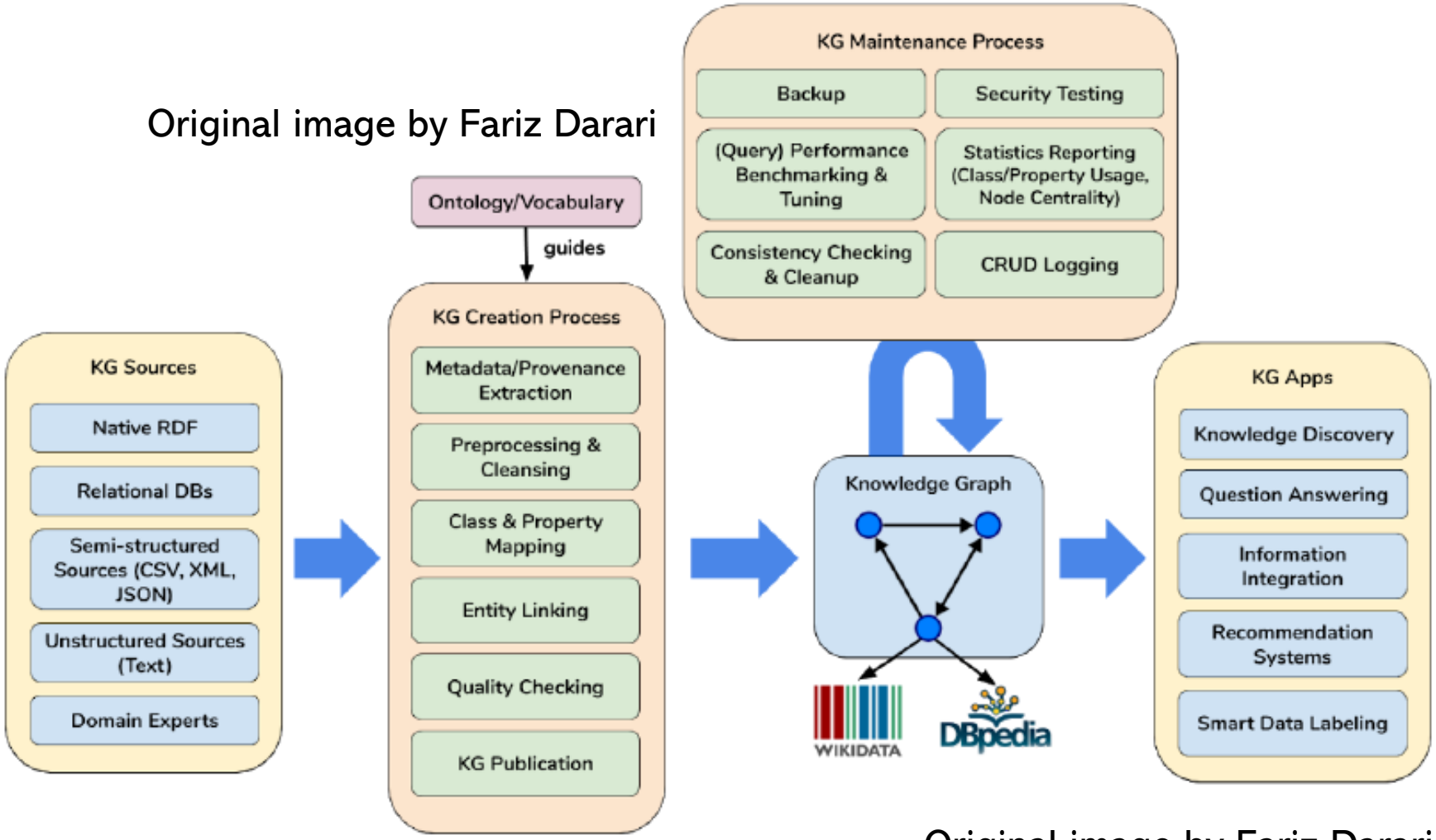
Most Recent KG Stack (2017)



(Kingsley Uyi Idehen, 2017)

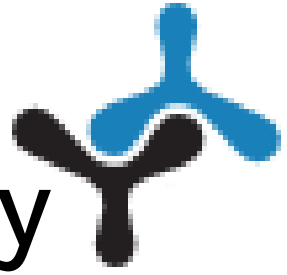
KG Workflow

Original image by Fariz Darari



Original image by Fariz Darari

Apache Jena Library

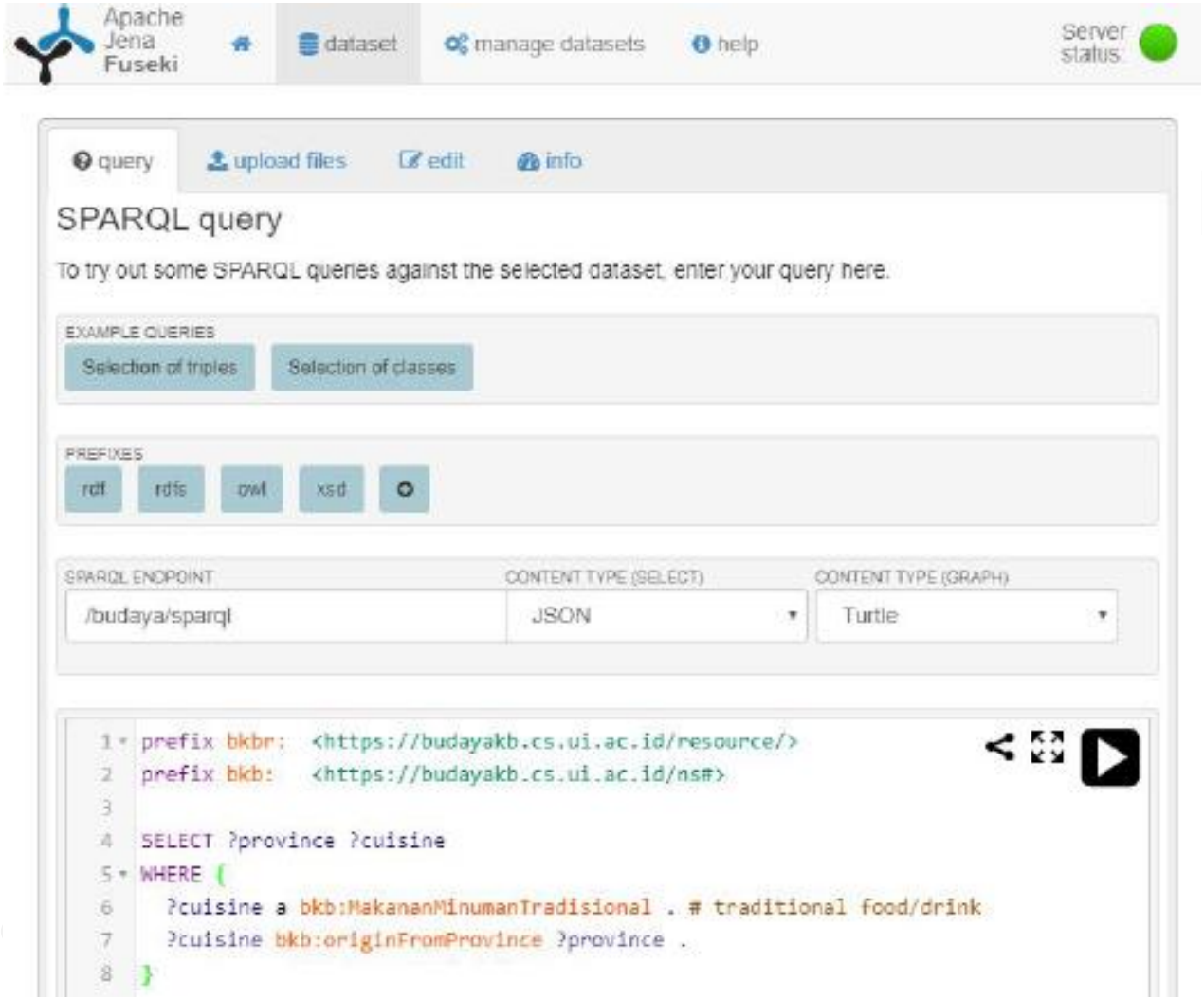


- Apache Jena (or Jena in short) is a free and open source Java framework for building semantic web and Linked Data applications.
- The framework is composed of different APIs interacting together to process RDF data:
  - > RDF API
  - > ARQ (SPARQL)
  - > TDB triple store
  - > Fuseki SPARQL server
  - > Ontology & inference API
- Link: https://jena.apache.org/

Apache Jena Fuseki

- Apache Jena Fuseki is a SPARQL server, part of offering from the Apache Jena library, though it can be used separately.
- As a SPARQL server, its main function is to act as the protocol engine for SPARQL querying, update, and graph store.
- That is, Fuseki acts as a SPARQL endpoint to whatever RDF data behind it.
- It can run with RDF data stored in memory or together with a persistent RDF store.
- For the latter, Jena Fuseki can work well together with Jena TDB storage.

Apache Jena Fuseki



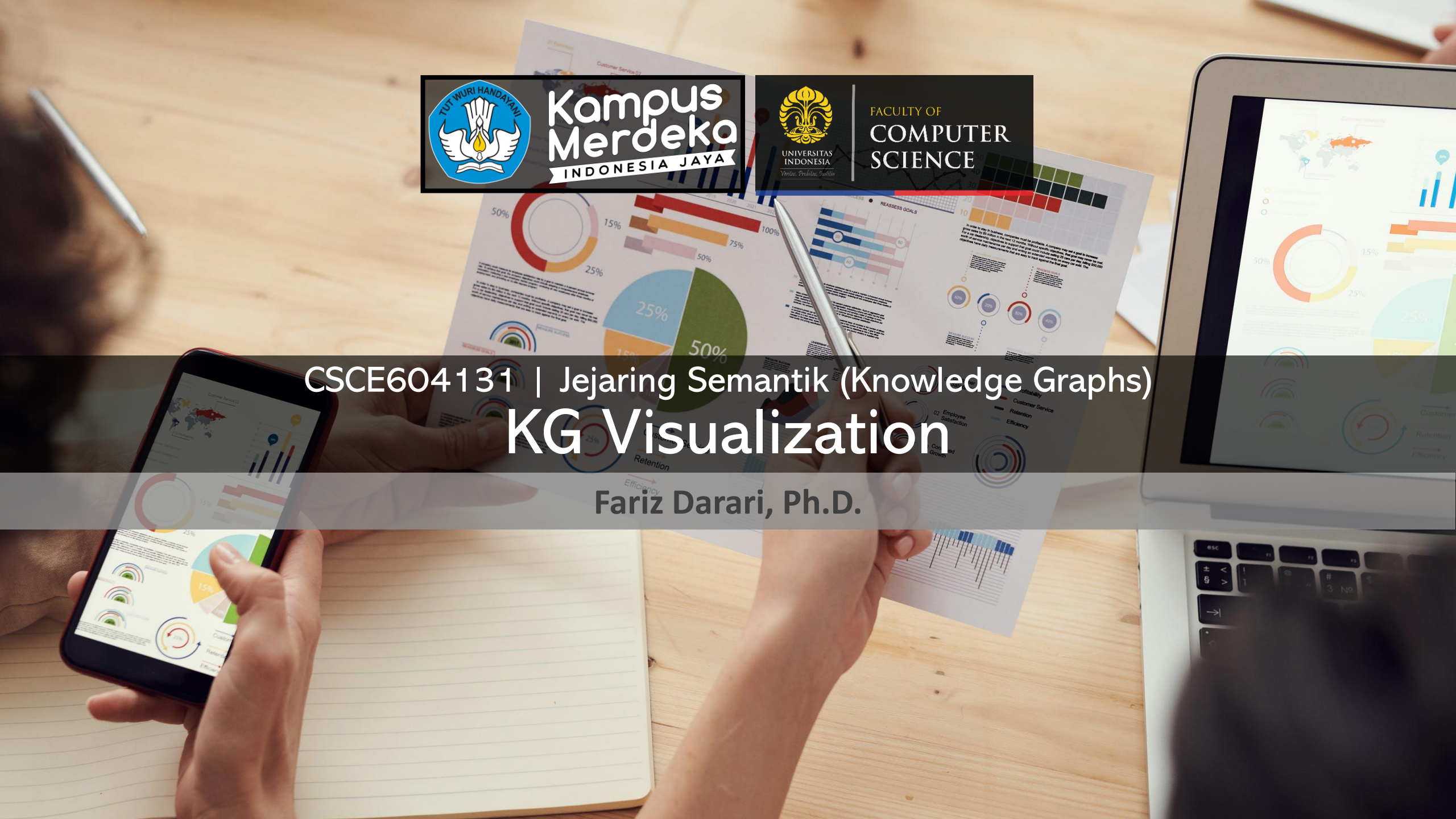
Apache Jena TDB

- Apache Jena TDB is the RDF storage and query component of the Apache Jena library.
- As such, it supports the whole range of Jena APIs.
- Typically, one uses Jena TDB programmatically through Jena library.
- Jena TDB is tied to a single JVM, that is, it cannot be directly accessed by more than one JVM at a time.
- To enable sharing across multiple applications, it must be used in conjunction with Fuseki.

Eclipse RDF4J

- Eclipse RDF4J is a Java framework for processing and handling RDF data.
- This includes creating, parsing, scalable storage, reasoning and querying with RDF and Linked Data.
- It offers an easy-to-use API that can be connected to all leading RDF database solutions.
- It allows you to connect with SPARQL endpoints and create applications that leverage the power of linked data and Semantic Web.
- Link: https://rdf4j.org/

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia



CSCE604131 | Jejaring Semantik (Knowledge Graphs)

KG Visualization

Fariz Darari, Ph.D.

Table: COVID-19 Victims

dateOfDeath	name	citizenship	profession
July 28, 2021	Porfirio Armando Betancourt	Honduras	association football player
July 26, 2021	René Juárez Cisneros	Mexico	economist
July 26, 2021	Natty Hollmann	Argentina	model
July 25, 2021	Hasip Kalimuddin Syam	Indonesia	politician
July 25, 2021	Roberto Valcárcel	Bolivia	painter
July 23, 2021	Mahmoud Khedri	Algeria	politician
July 23, 2021	Rodolfo Peña Flores	Costa Rica	administrator
July 23, 2021	Jimmy Demianus Ijie	Indonesia	politician
July 22, 2021	Ian Palmer	South Africa	association football player
July 20, 2021	Theo Jubitana	Suriname	village head

Wikidata Query Service

Link

Image Grid: COVID-19 Vaccines

commons:2020 12 27 mas vacunados en la residencia mixta 1.jpg
COVID-19 vaccine

commons:Moderna COVID-19 vaccine.jpg
mRNA-1273 vaccine

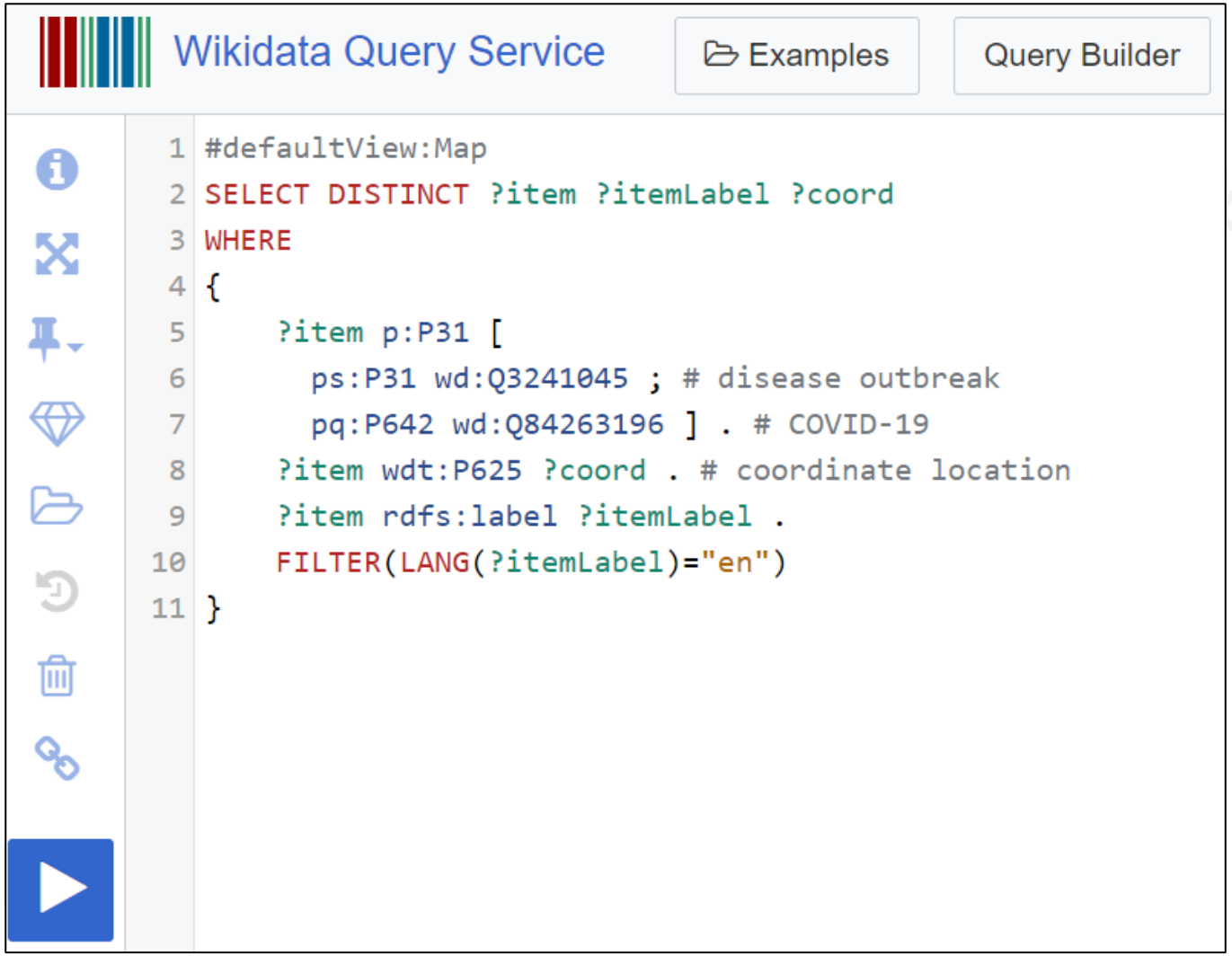
Wikidata Query Service

Link

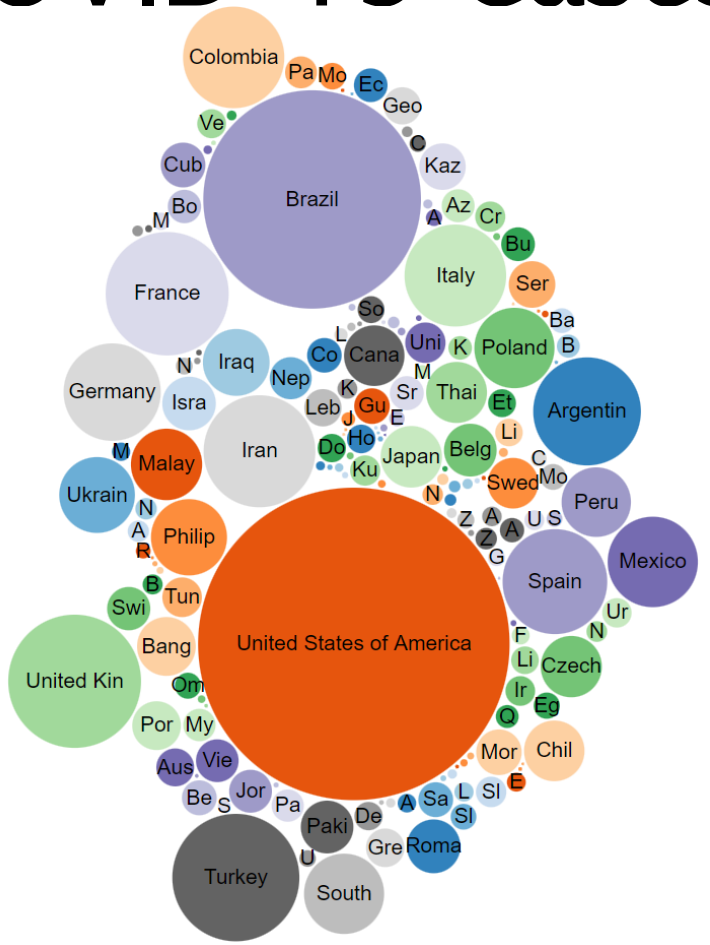
Map: COVID-19 Outbreaks

Link

Map: COVID-19 Outbreaks

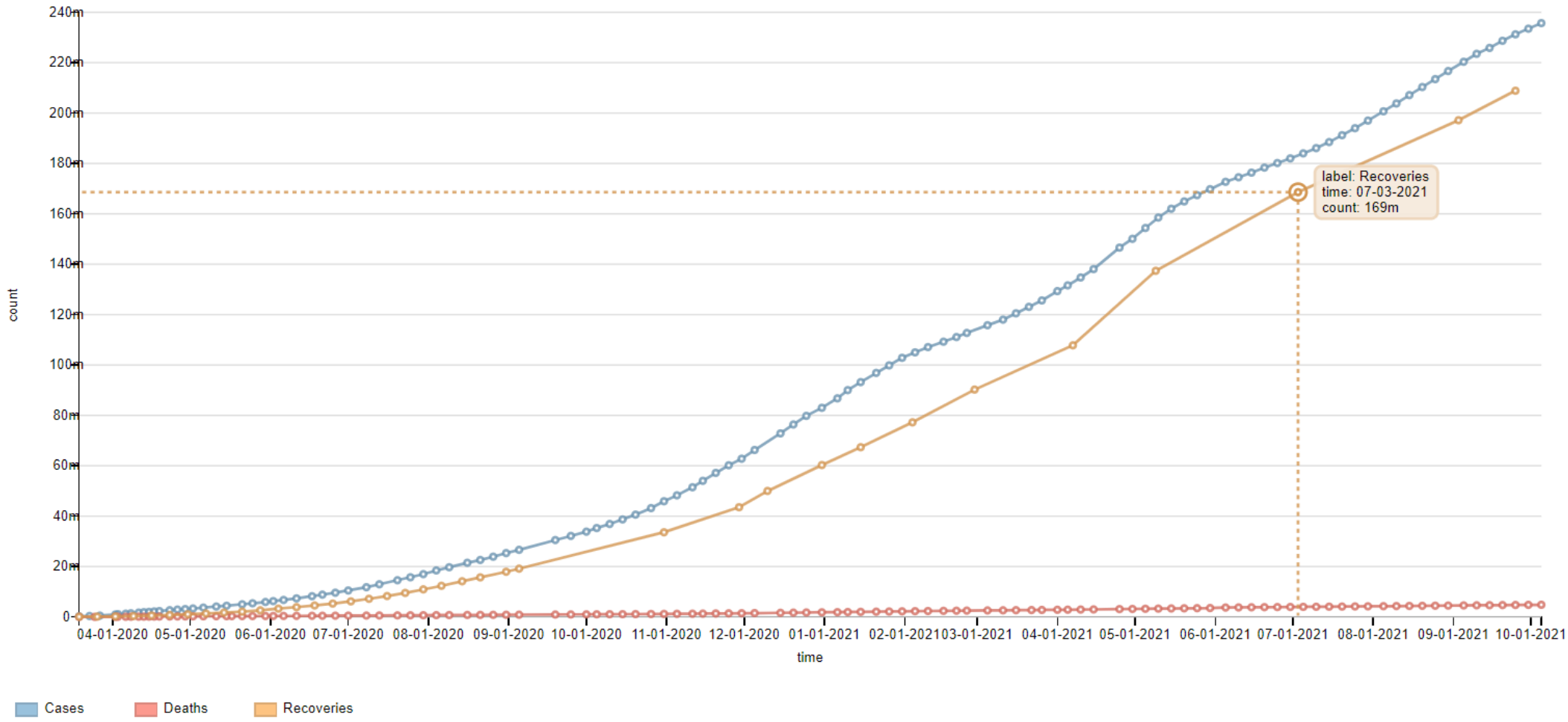


Bubble Plot: COVID-19 Cases



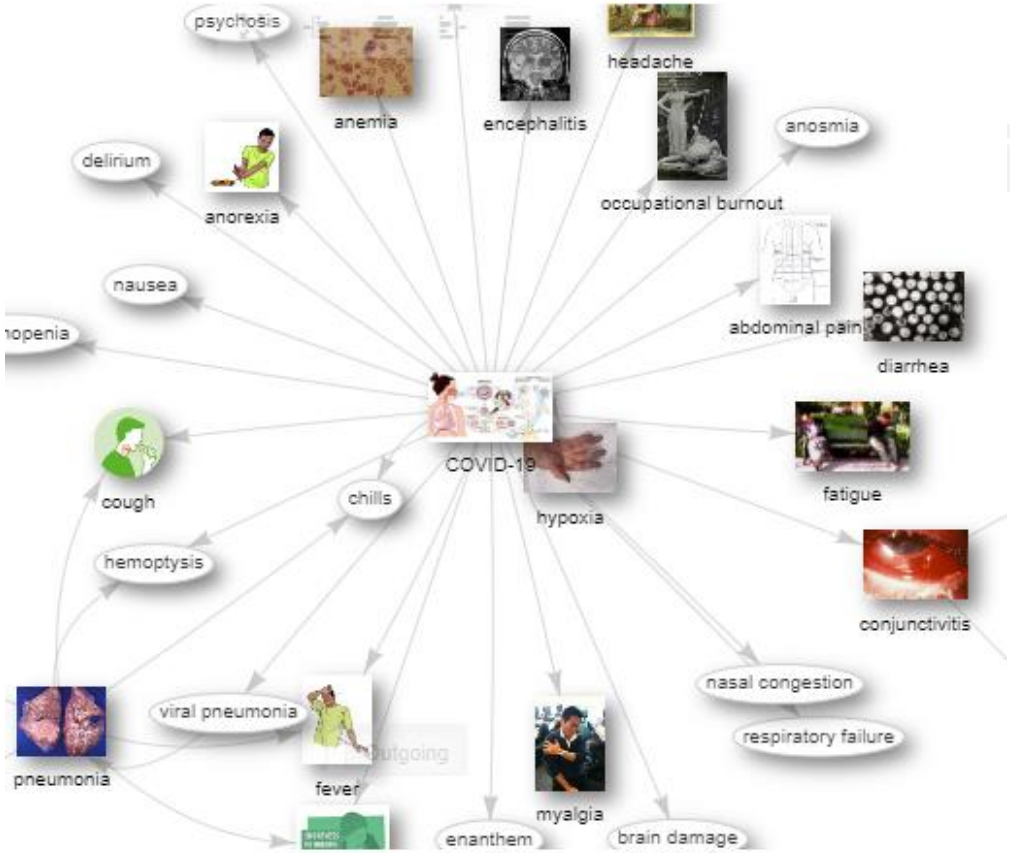
Link: https://w.wiki/4CRg

Line Charts: COVID-19 Growths



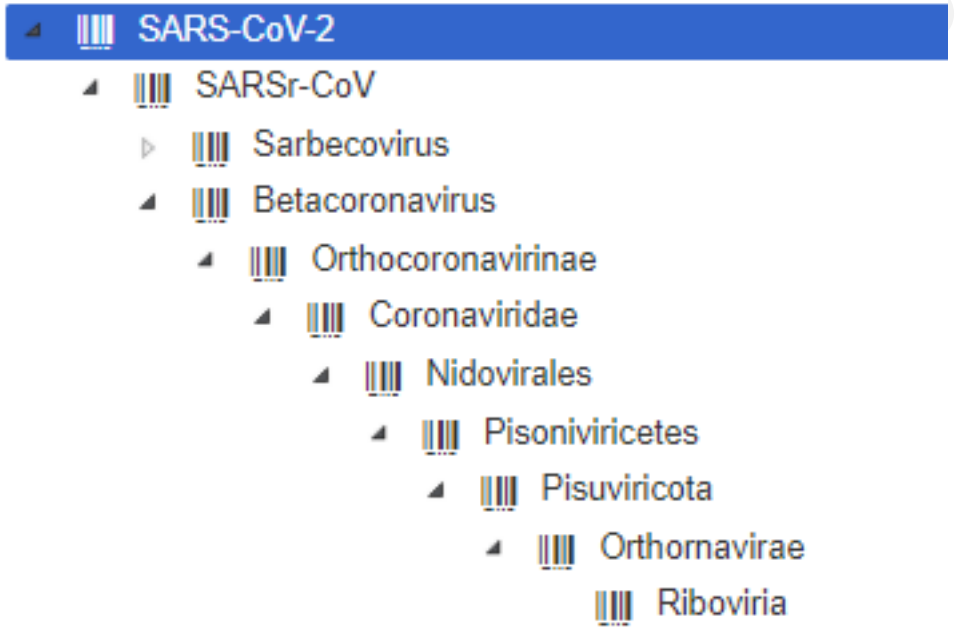
Link: https://w.wiki/gN7

Graph: COVID-19 Symptoms



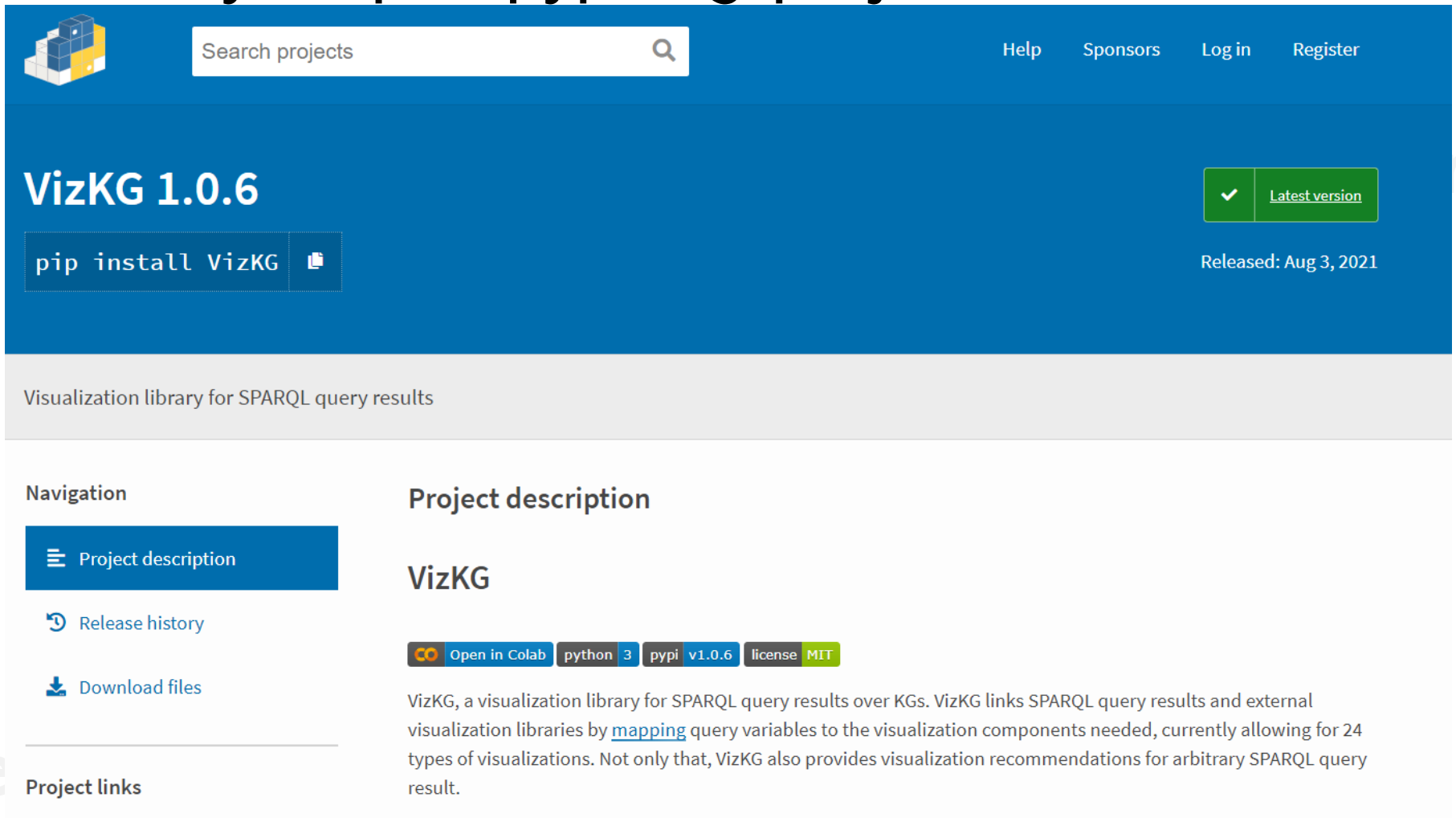
[Link]

Tree: COVID-19 Taxonomy



Link

VizKG Library: https://pypi.org/project/VizKG/



VizKG Library: https://pypi.org/project/VizKG/

Demo

Source Code

Statistics

GitHub statistics:

Stars: 4

Forks: 1

Open issues/PRs: 0

View statistics for this project via Libraries.io, or by using our public dataset on Google BigQuery

Installation

Use the package manager pip to install VizKG.

```
pip install VizKG
```

Usage

```
# Import the library
import VizKG.visualize as vkg
```

VizKG Library: https://pypi.org/project/VizKG/

Supported Visualizations

- Table
- ImageGrid
- Map
- Tree
- Timeline
- Dimensions
- Graph
- WordCloud
- Tree Map
- SunBurst Chart
- Line Chart
- Bar Chart
- Area Chart
- StackedArea Chart
- Histogram
- Density Plot
- Box Plot
- Violin Plot
- Bubble Chart
- Scatter Chart
- HeatMap
- Radar Chart

VizKG Library: https://pypi.org/project/VizKG/

VizKG: A Framework for Visualizing SPARQL Query Results over Knowledge Graphs

Hana Raissya[1], Fariz Darari (✉)[1,2], and Fajar J. Ekaputra[3]

[1] Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia
[2] Tokopedia-UI AI Center of Excellence, Jakarta, Indonesia
{hana.raissya, fariz}@ui.ac.id
[3] Institute of Information Systems Engineering, TU Wien, Vienna, Austria
fajar.ekaputra@tuwien.ac.at

Abstract. Despite the rise of the knowledge graph (KG) popularity, understanding SPARQL query results from a KG can be challenging for users. The use of data visualization tools, e.g., Wikidata Query Service and YASGUI, can help address this challenge. However, existing tools are either focused just on a specific KG or only provided as a web interface. This paper proposes VizKG, a framework that provides a wide range of visualizations for SPARQL query results over KGs. VizKG aims to assist users in extracting patterns and insights from data in KGs, and hence supporting further KG analysis. VizKG features a wrapper that links SPARQL query results and external visualization libraries by mapping query result variables to the required visualization components, currently allowing for 24 types of visualizations. Not only that, VizKG also includes visualization recommendations for arbitrary SPARQL query results as well as extension mechanisms for additional visualization types. In our evaluation, the visualization recommendation feature of VizKG achieves an accuracy of 87.8%. To demonstrate the usefulness of VizKG in practical settings, this paper also reports on use case evaluation over various domains and KGs. A Python-based, Jupyter Notebook friendly implementation of VizKG is openly available at https://pypi.org/project/VizKG/.

Keywords: Visualization · Knowledge Graphs · SPARQL · Insights

VOILA 2021 Topics Submission Committees

12:00 - 12:05 Opening and Introduction

12:05 - 13:25 Session I: Linked data - SPARQL

12:05 - 12:20 A Survey on User Interaction with Linked Data
by Mariana Aguiar, Sérgio Nunes and Bruno Giesteira

12:20 - 12:35 Fast Approximate Autocompletion for SPARQL Query Builders
by Gabriel de la Parra and Aidan Hogan

12:35 - 12:50 Displaying triple provenance with extensions of Fresnel
by Lloyd Rutledge, Pascal Mellema, Tije Pietersma and Stef Joosten

12:50 - 13:05 Visual Presentation of SPARQL Queries in ViziQuer
by Kārlis Čerāns, Julija Ovcinnikova, Mikus Grasmanis, Lelde Lace and Aiga Romane

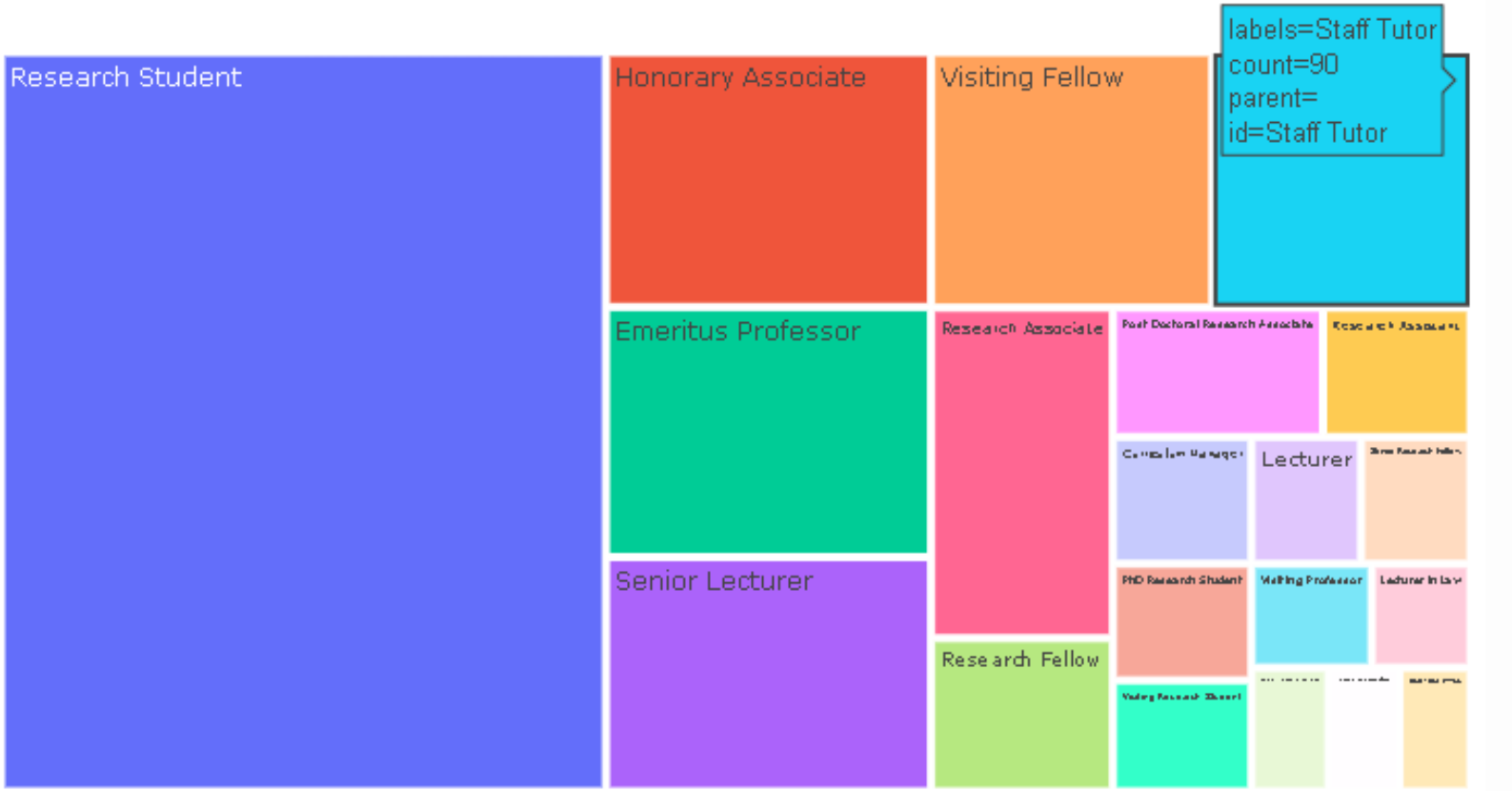
13:05 - 13:15 VizKG: A Framework for Visualizing SPARQL Query Results over Knowledge Graphs
by Hana Raissya, Fariz Darari and Fajar J. Ekaputra

13:15 - 13:25 Discussion

http://voila2021.visualdataweb.org/

VizKG Map: Temple (= candi) location in Indonesia

TreeMap: Number of University Employees by Job Title



Link

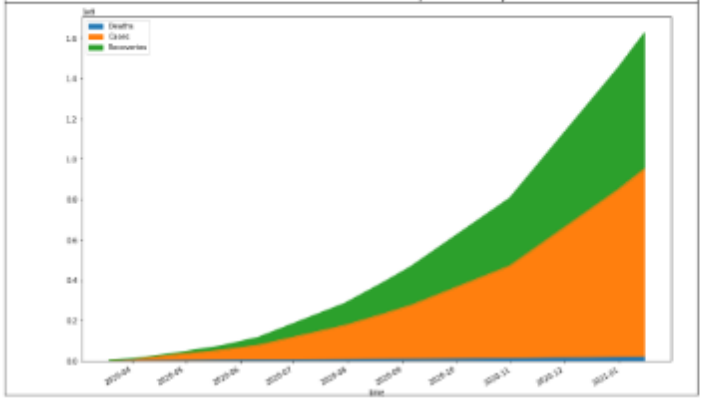
TreeMap: Number of University Employees by Job Title

OU_UK: Number of Employees by Job Title

```
sparql_query = """
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX schema: <http://schema.org/jobTitle>

SELECT DISTINCT ?jobTitle (COUNT(?jobTitle) as ?count) WHERE {?s a foaf:Person .
  ?s <http://schema.org/jobTitle> ?jobTitle .
  FILTER (lang(?jobTitle) != 'en')
}
GROUP BY ?jobTitle
HAVING (?count > 10)
"""
#to query another endpoint, change the URL for the service and the query
sparql_service_url = "https://data.open.ac.uk/sparql"
chart = vkg(sparql_query=sparql_query, sparql_service_url=sparql_service_url, chart='TreeMap')
chart.plot()
```

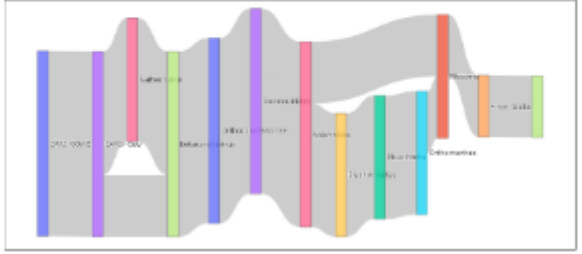
COVID-19

Wikidata: Growths of COVID-19 Cases, Deaths, and Recoveries



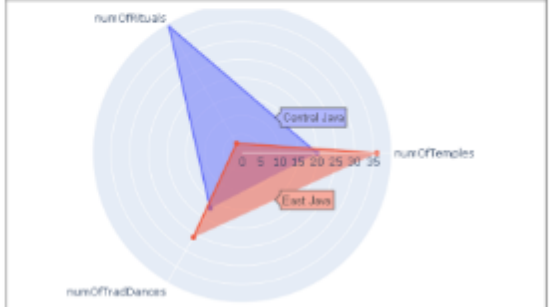
Wikidata: COVID-19 Vaccine Origins

Wikidata: COVID-19 Taxonomy

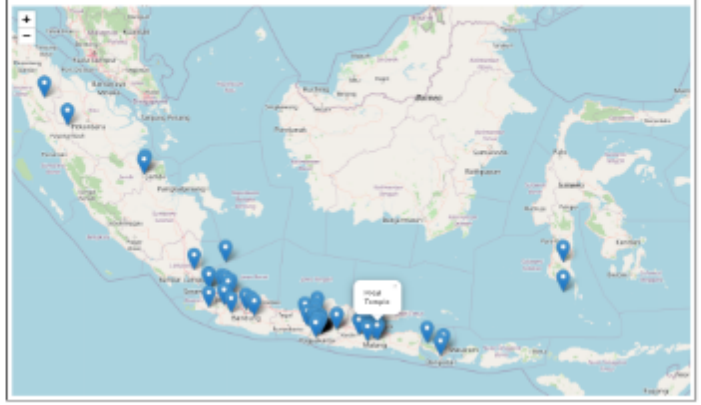


Cultural Heritage in Indonesia

Wikidata: Number of Temples, Rituals, and Traditional Dances in Central Java and East Java



DBpedia: Map of Temples in Indonesia

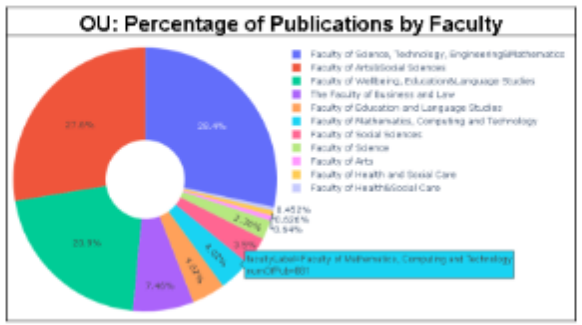


Budaya KB: Number of Temples in Indonesia

Higher Education

OU: Number of Employees by Job Title

OU: Percentage of Publications by Faculty



OU: Courses and Topics Offered

Link

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia

CSCE604131 | Jejaring Semantik (Knowledge Graphs)

KG Quality

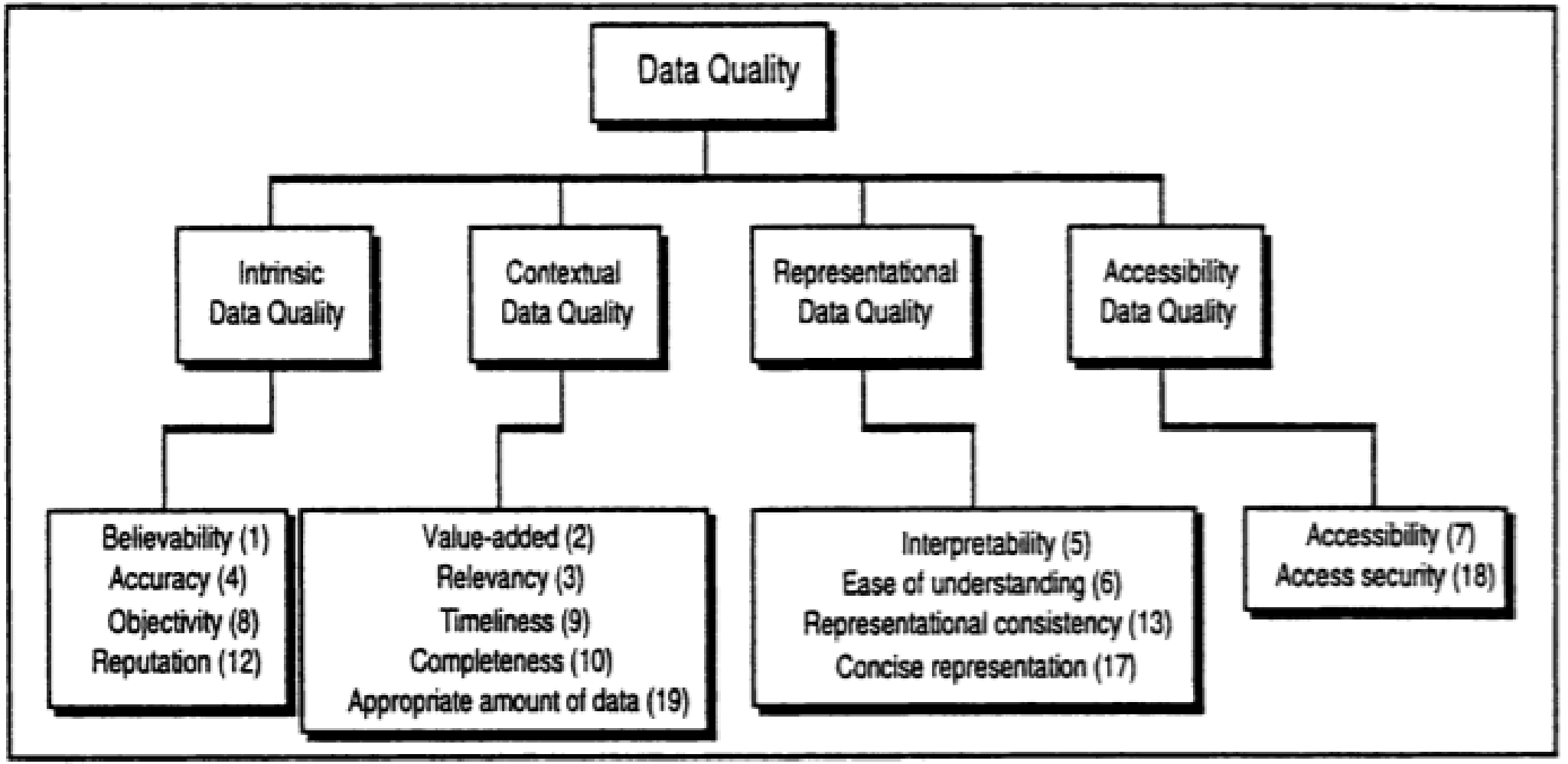
Fariz Darari, Ph.D.

Data Quality

Data quality is data that is fit for use by data consumers

https://www.datapine.com/blog/data-quality-management-and-metrics/

Data Quality Dimensions by (Wang & Strong, 1996)



KG Quality Dimensions (Zaveri et al., 2016)

- Accessibility dimensions
- Intrinsic dimensions
- Contextual dimensions
- Representational dimensions

Compiled by Fariz Darari

Accessibility Dimension

- **Availability**
  The extent to which data (or some portion of it) is present, obtainable, and ready for use
- **Licensing**
  Clarity of license and usage/sharing policy
- **Interlinking**
  Degree of interconnectedness of data from one source to another
- **Security**
  The extent to which data is protected against alteration and misuse
- **Performance**
  Efficiency of large-scale KG systems

Compiled by Fariz Darari

Intrinsic Dimension

- **Syntactic Validity**
  Degree to which an RDF document conforms to the specification of the serialization format
- **Semantic Accuracy**
  Degree to which data values correctly represent the real world facts
- **Consistency**
  Free of (logical/formal) contradictions with respect to particular knowledge representation and inference mechanisms
- **Conciseness**
  Minimization of redundancy of entities at the schema and the data level
- **Completeness**
  Degree to which all required information is present in a particular dataset

Compiled by Fariz Darari

Contextual Dimension

- **Relevancy**
  Provision of information which is in accordance with the task at hand and important to the user query
- **Trustworthiness**
  Degree to which the information is accepted to be correct, true, real and credible
- **Understandability**
  The ease with which data can be comprehended without ambiguity and be used by a human information consumer
- **Timeliness**
  How up-to-date data is relative to a specific task

Representational Dimension

- **Representational-Conciseness**
  The representation of the data, which is compact and well formatted
- **Interoperability**
  Degree to which the format and structure of the information conforms to previously returned information as well as data from other sources
- **Interpretability**
  Technical aspects of the data, that is, whether information is represented using an appropriate notation and whether the machine is able to process the data
- **Versatility**
  Availability of the data in different representations and in an internationalized way

Compiled by Fariz Darari

KG Quality Interrelations

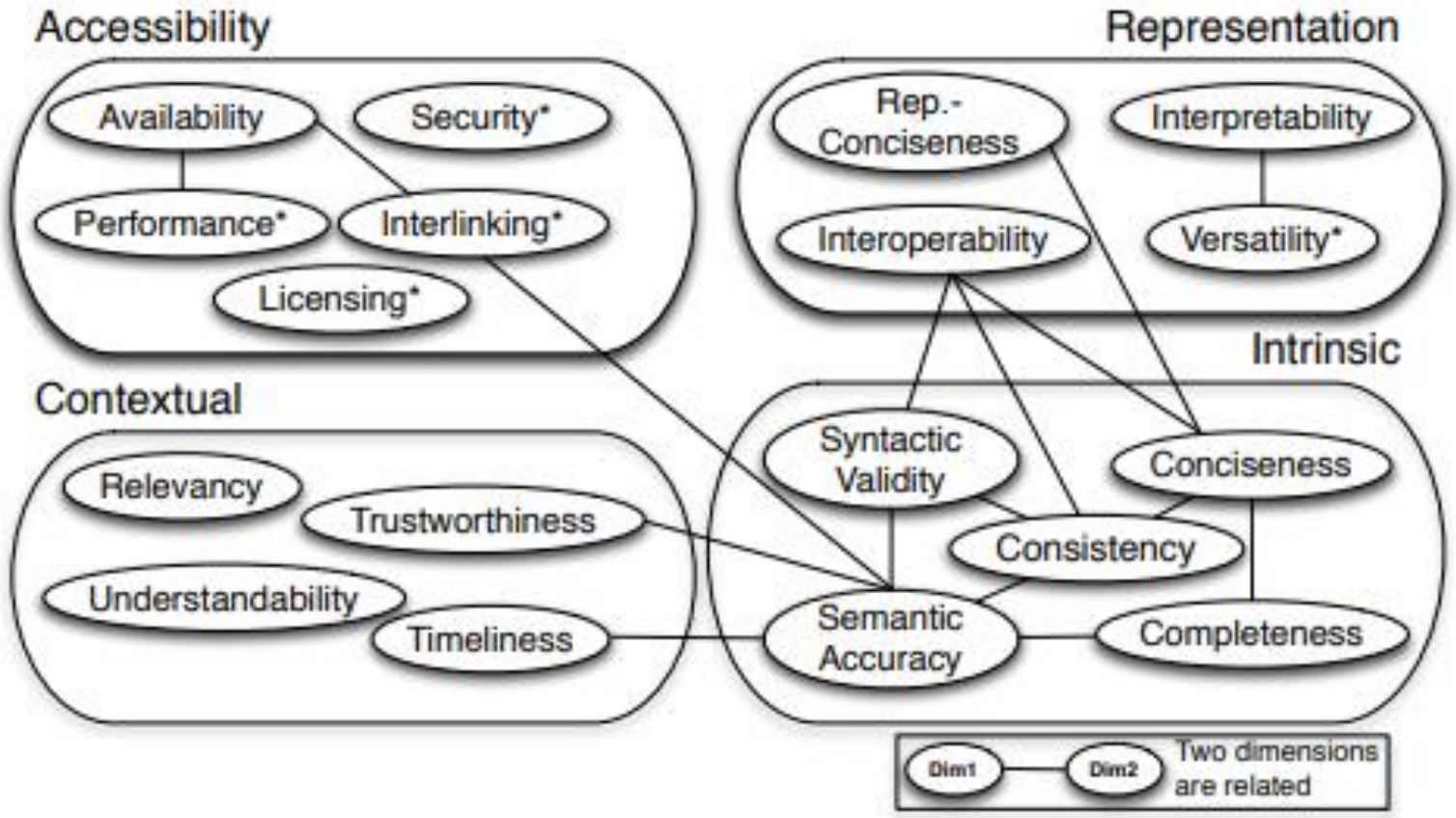


Fig. 2. Linked Data quality dimensions and the relations between them. The dimensions marked with '*' are specific for Linked Data.

(Zaveri et al., 2016)

Credits:

http://mitiq.mit.edu/Documents/Publications/TDQMpub/14_Beyond_Accuracy.pdf

https://www.datapine.com/blog/data-quality-management-and-metrics/

http://www.semantic-web-journal.net/system/files/swj773.pdf

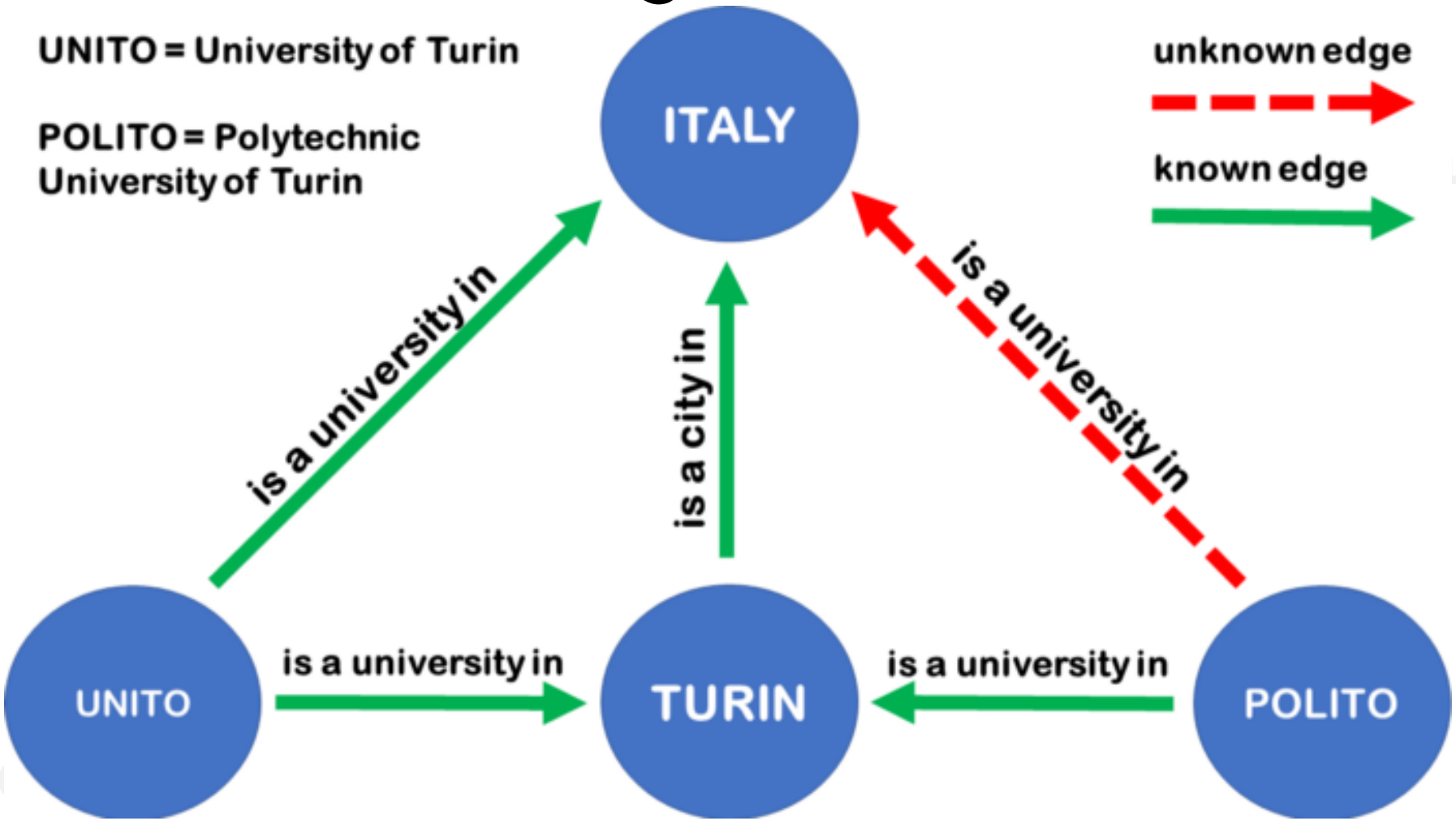
Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia

CSCE604131 | Jejaring Semantik (Knowledge Graphs)

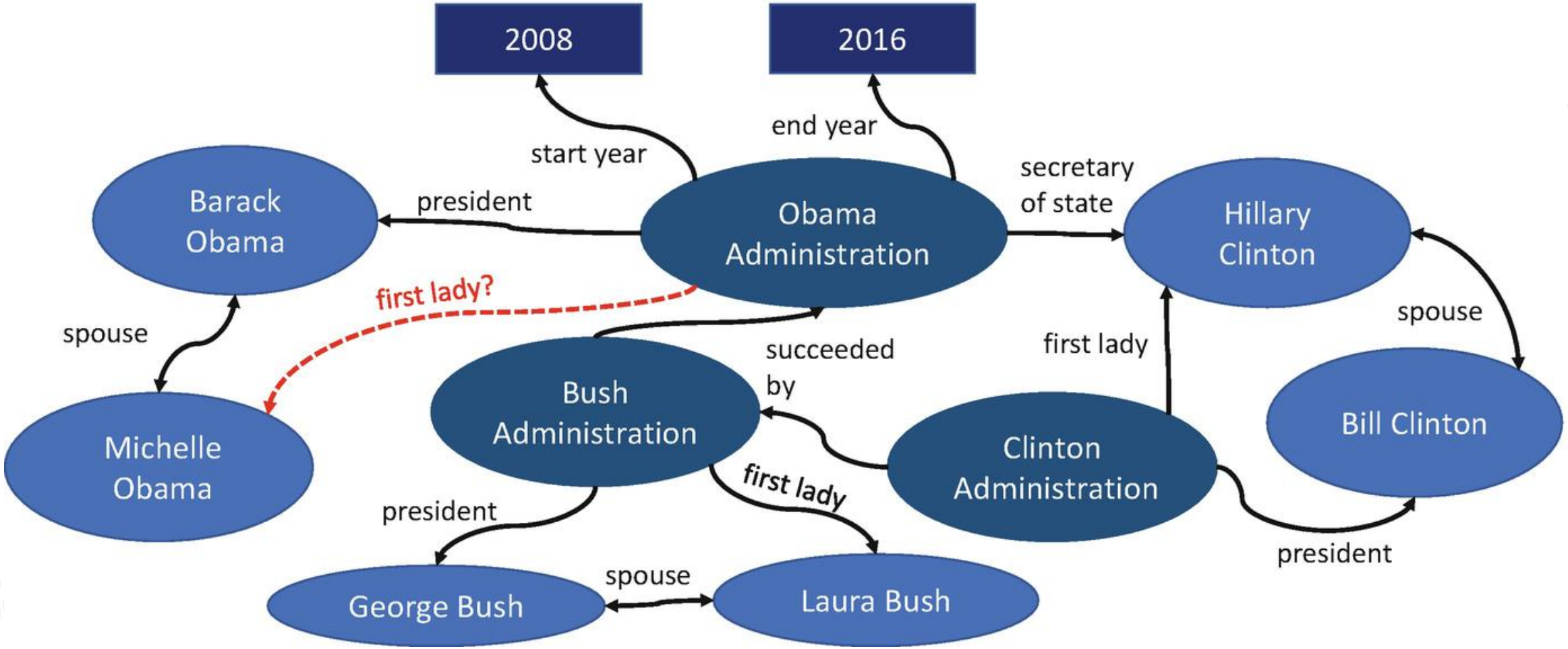
KG Embeddings

Fariz Darari, Ph.D.

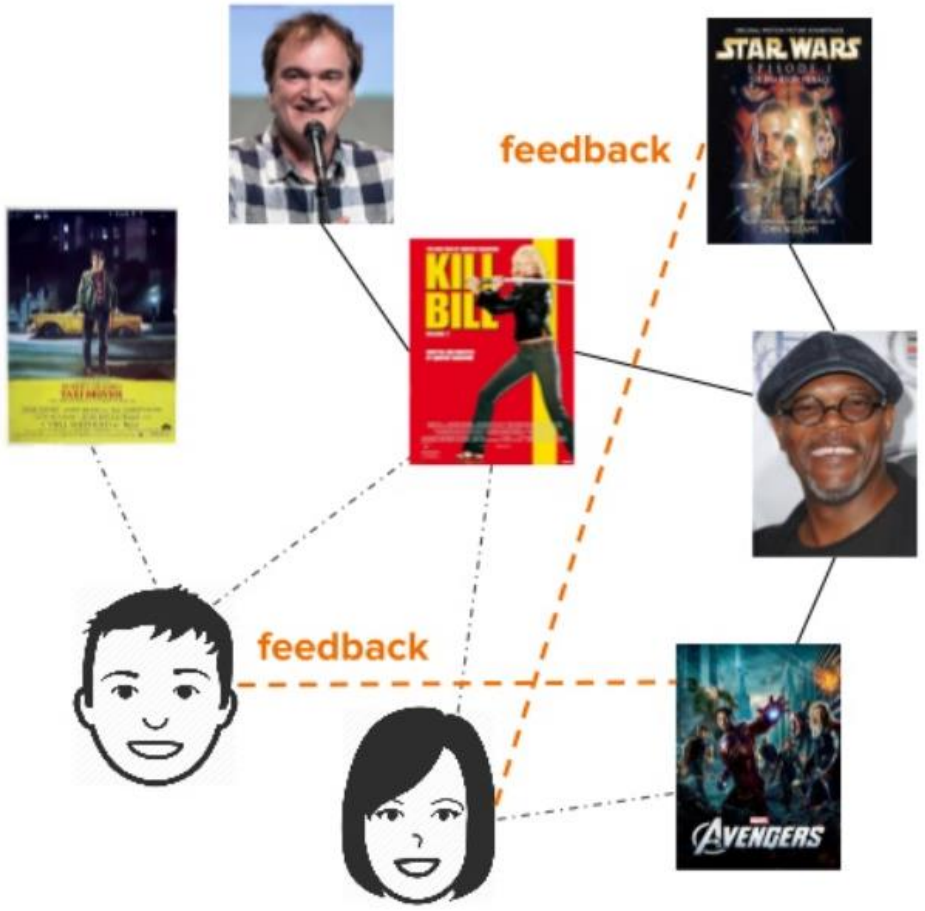
Motivation: Link/Edge Prediction



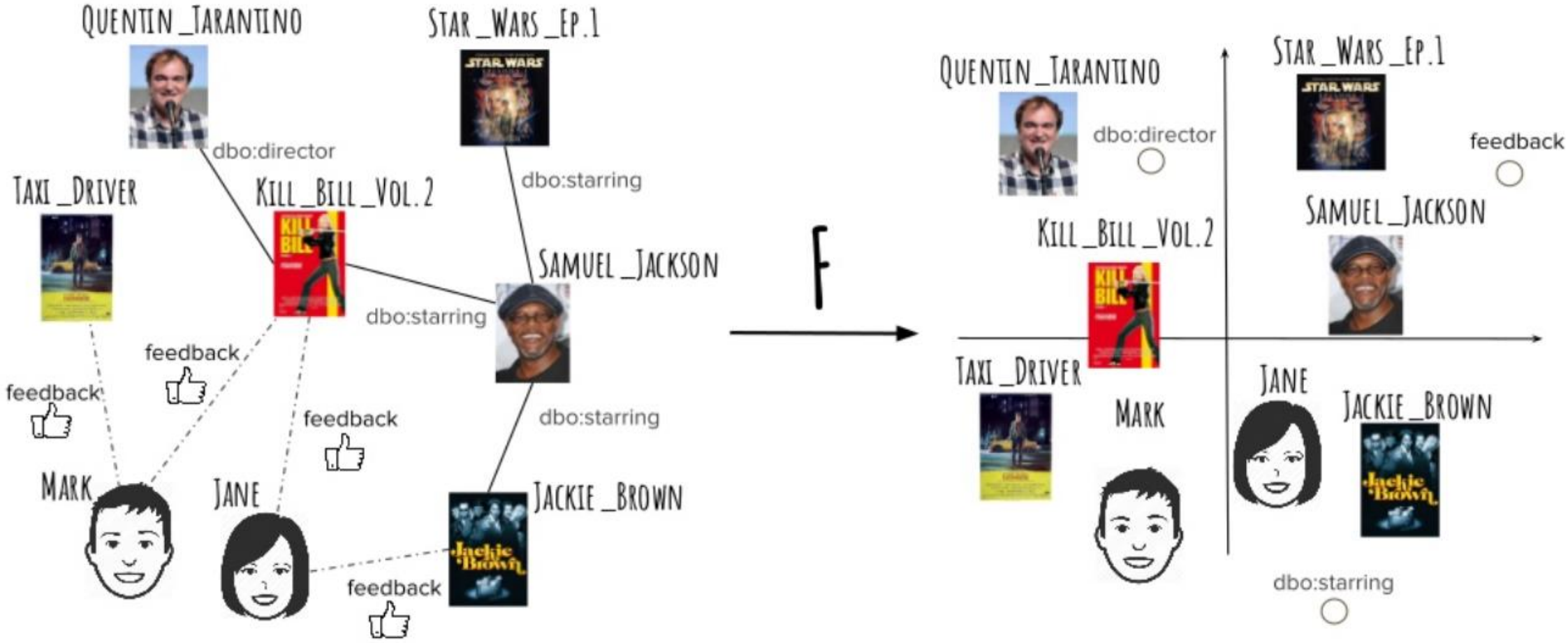
Motivation: Link/Edge Prediction



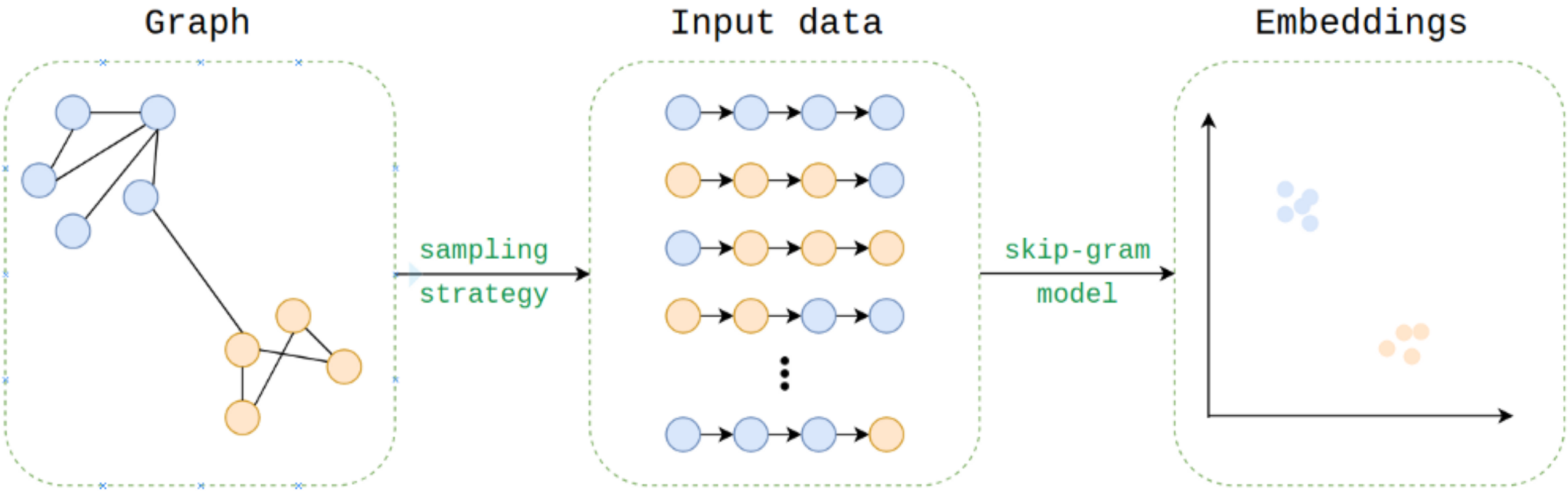
Motivation: KG Recommender System



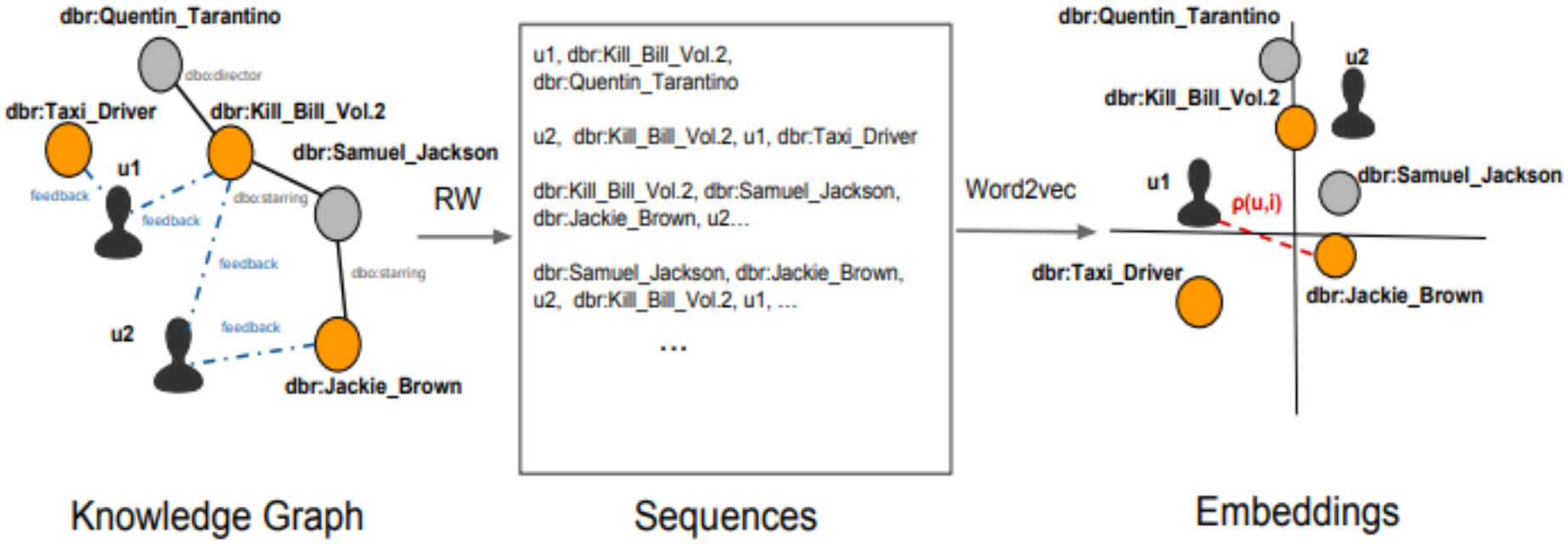
KG Embeddings: Idea



KG Embeddings: node2vec



KG Embeddings: node2vec for Item Recommendation

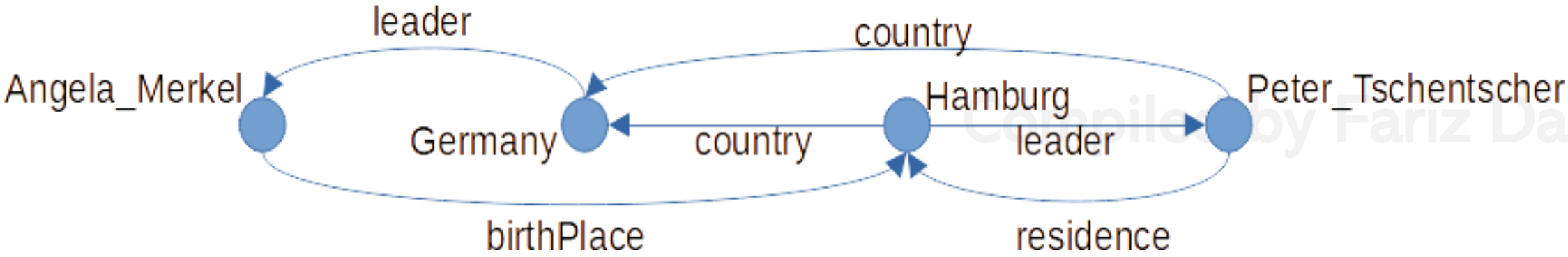


Source: https://2018.eswc-conferences.org/files/posters-demos/paper_265.pdf

KG Embeddings: rdf2vec

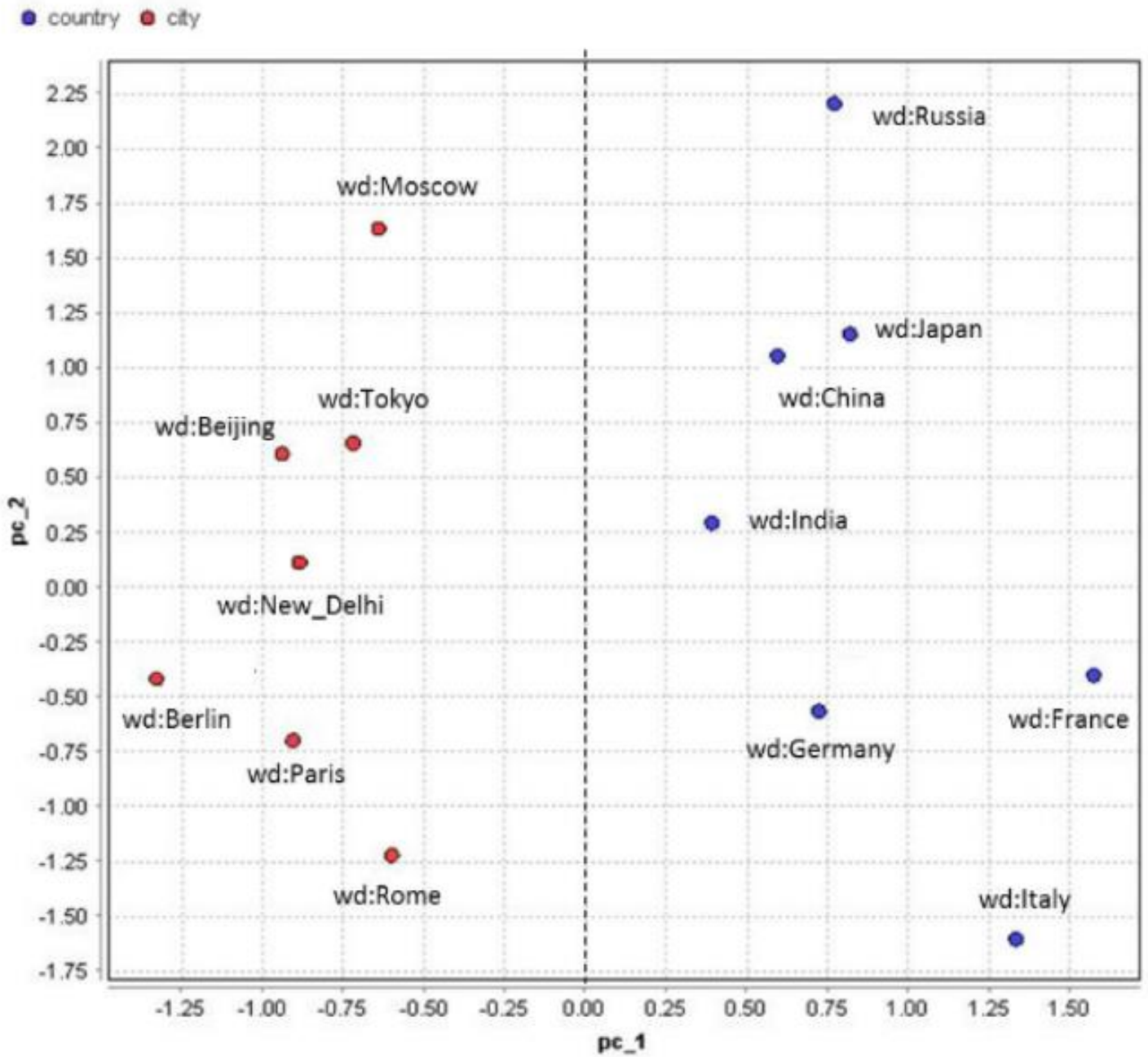
- RDF2vec is a tool for creating vector representations of RDF graphs.
- In essence, RDF2vec creates a numeric vector for each node in an RDF graph.
- RDF2vec was inspired by the word2vec approach.
- Random walks on the RDF graph are used to create sequences of RDF nodes, which are then used as input for the word2vec algorithm.
- Application scenarios: KGs as background knowledge in data mining tasks, or content-based recommender systems.
- Link: http://rdf2vec.org/

KG Embeddings: rdf2vec

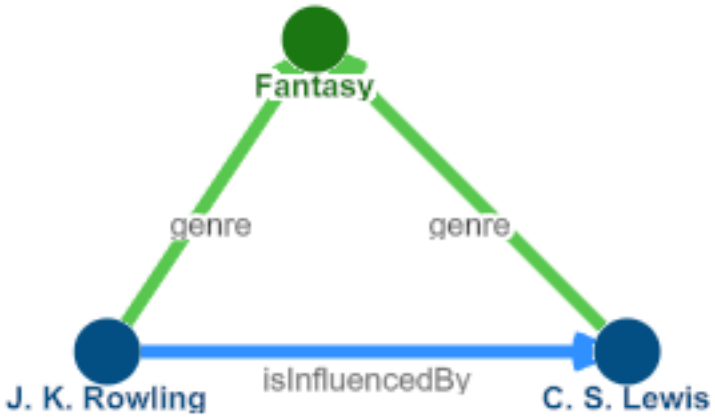


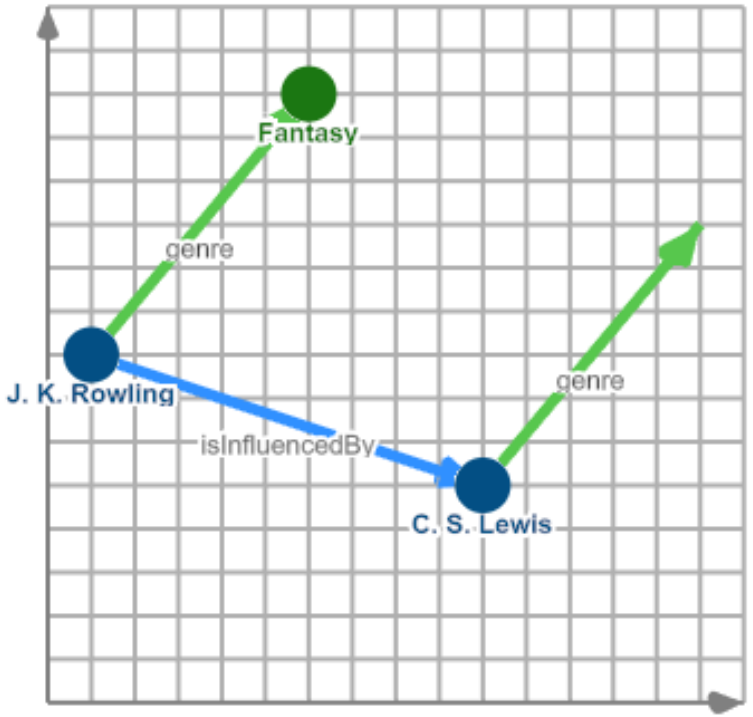
From this graph, a set of random walks that could be extracted may look as follows:

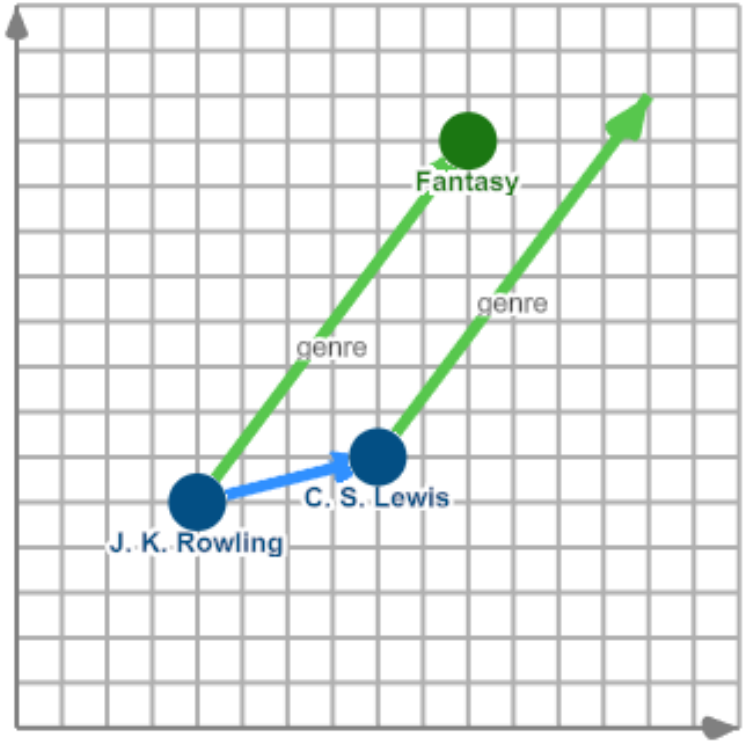
```
Hamburg -> country -> Germany           -> leader     -> Angela_Merkel
Germany -> leader  -> Angela_Merkel     -> birthPlace -> Hamburg
Hamburg -> leader  -> Peter_Tschentscher -> residence  -> Hamburg
```



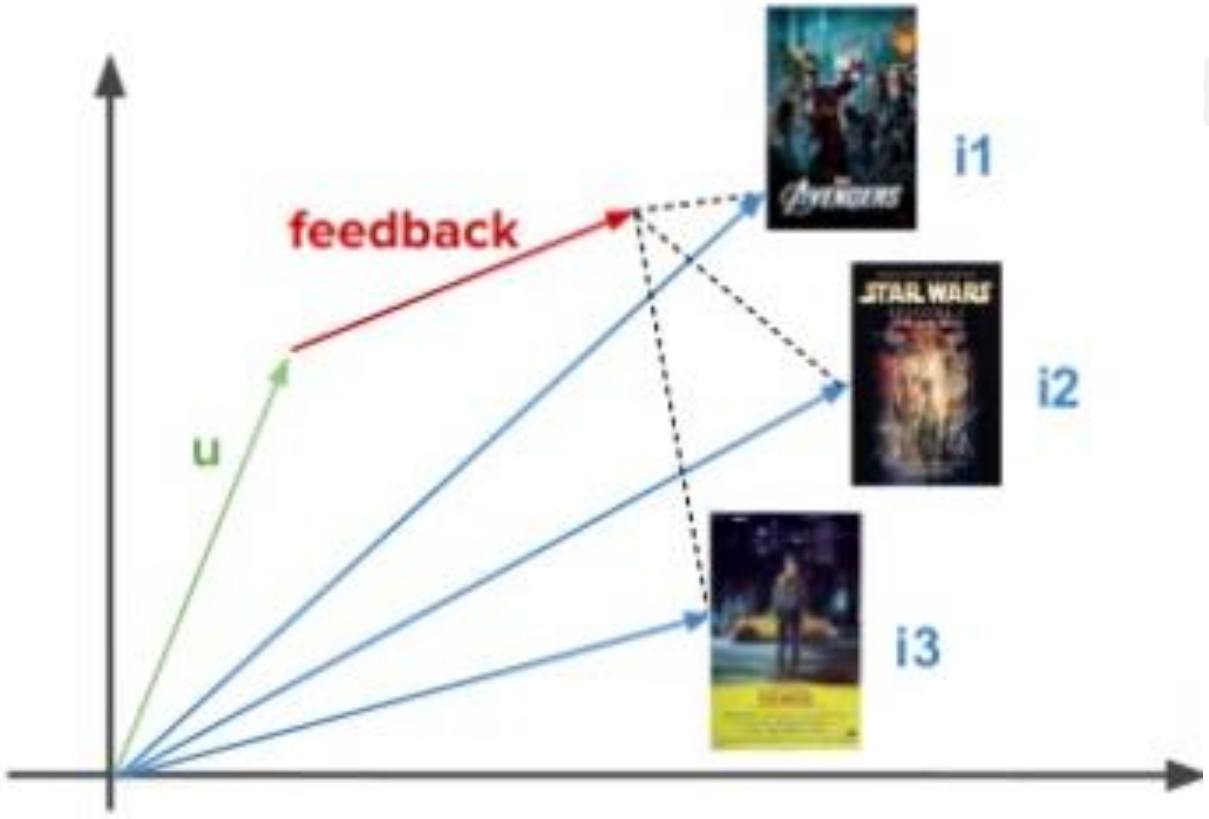
KG Embeddings: TransE







KG Embeddings: TransE for Recommending Movies



KG Embeddings Library: AmpliGraph

- AmpliGraph is a suite of neural machine learning models for relational learning, a branch of machine learning that deals with supervised learning on KGs.
- Use AmpliGraph if you need to:
  - Discover new knowledge from an existing knowledge graph.
  - Complete large knowledge graphs with missing statements.
  - Generate stand-alone knowledge graph embeddings.
  - Develop and evaluate a new relational model.
- Link: https://docs.ampligraph.org

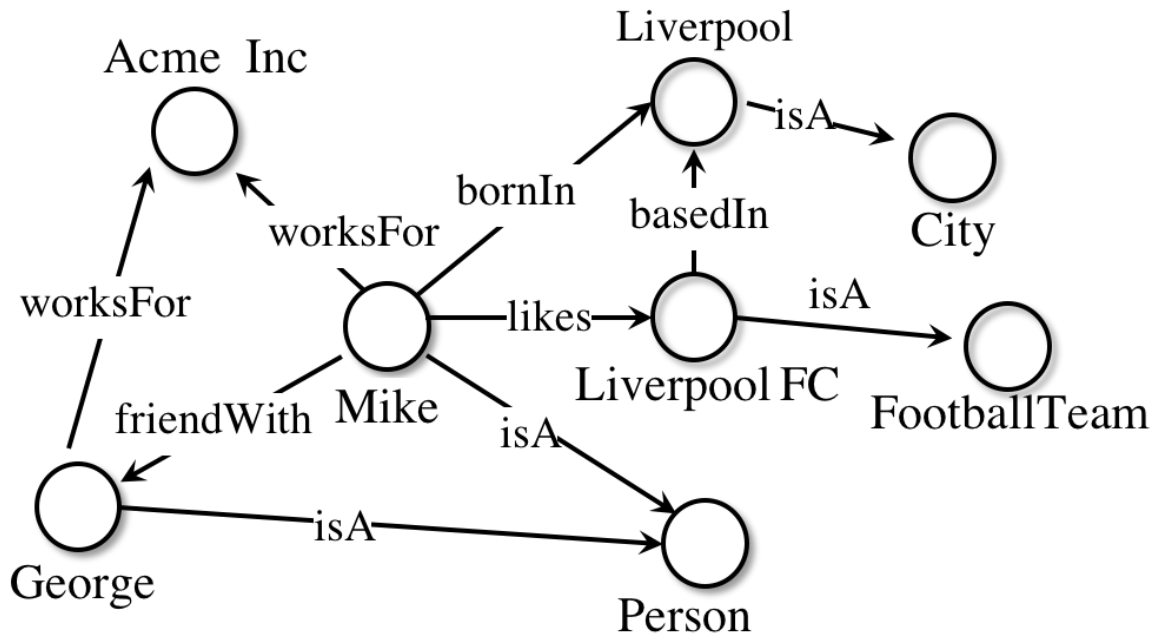
KG Embeddings Library: AmpliGraph Features

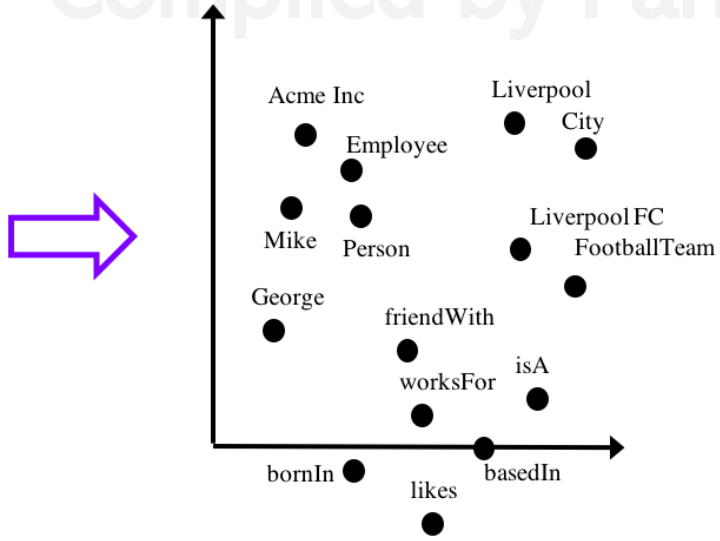
- Intuitive APIs: AmpliGraph APIs are designed to reduce the code amount required to learn models that predict links in knowledge graphs.
- GPU-Ready: AmpliGraph is based on TensorFlow, and it is designed to run seamlessly on CPU and GPU devices - to speed-up training.
- Extensible: Roll your own knowledge graph embeddings model by extending AmpliGraph base estimators.

KG Embeddings Library: AmpliGraph Modules

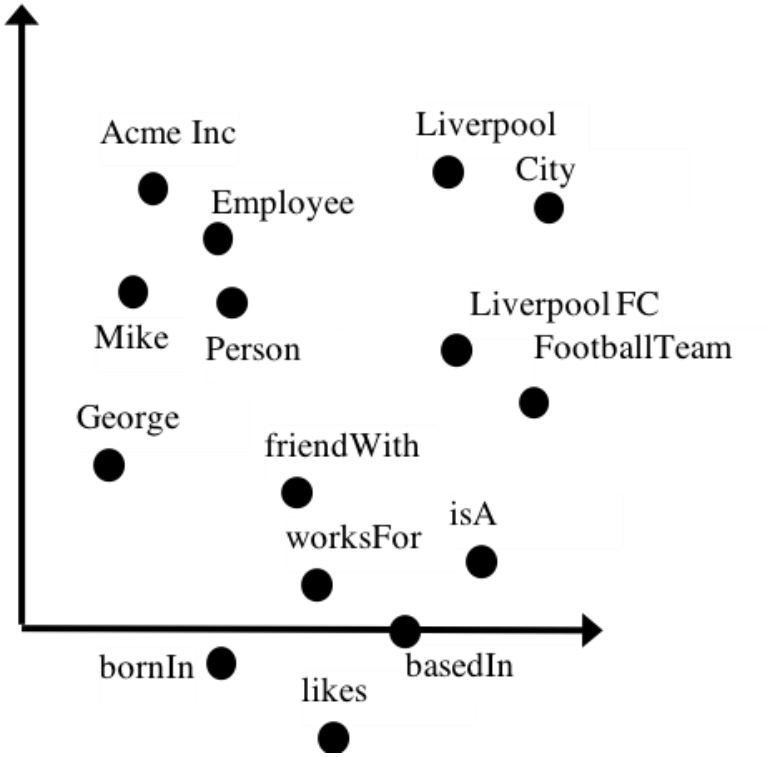
- Datasets: helper functions to load datasets (KGs).
- Models: knowledge graph embedding models. AmpliGraph contains TransE, DistMult, ComplEx, HoIE, ConvE, ConvKB (More to come!)
- Evaluation: metrics and evaluation protocols to assess the predictive power of the models.
- Discovery: High-level convenience APIs for knowledge discovery (discover new facts, cluster entities, predict near duplicates).

KG Embeddings Library: AmpliGraph Application





KG Embeddings Library: AmpliGraph Application



Credits:

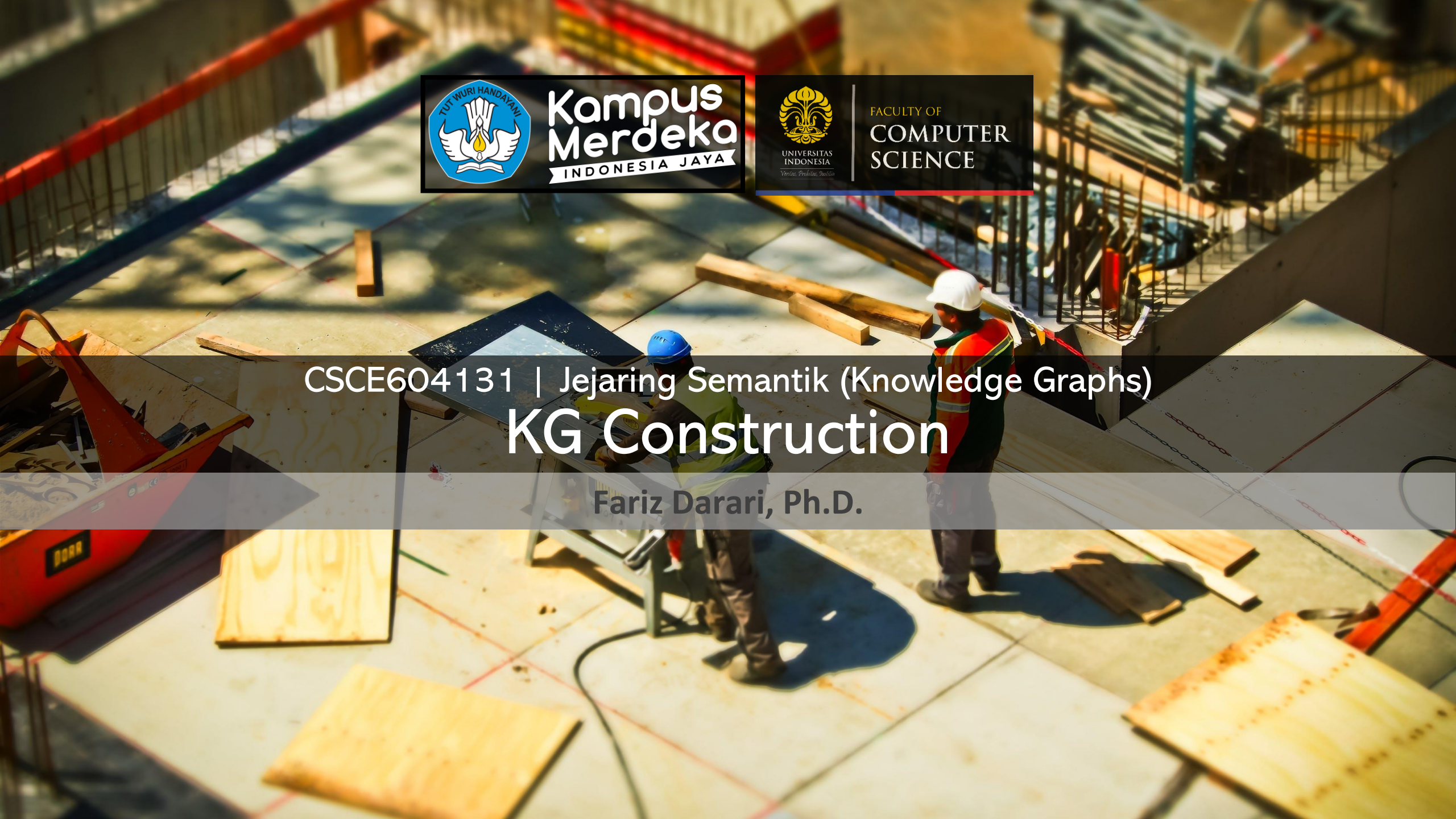
https://towardsdatascience.com/embedding-models-for-knowledge-graph-completion-a66d4c01d588

https://link.springer.com/chapter/10.1007/978-3-030-12375-8_4

https://www.slideshare.net/EnricoPalumbo2/an-empirical-comparison-of-knowledge-graph-embeddings-for-item-recommendation

https://towardsdatascience.com/node2vec-embeddings-for-graph-data-32a866340fef

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia

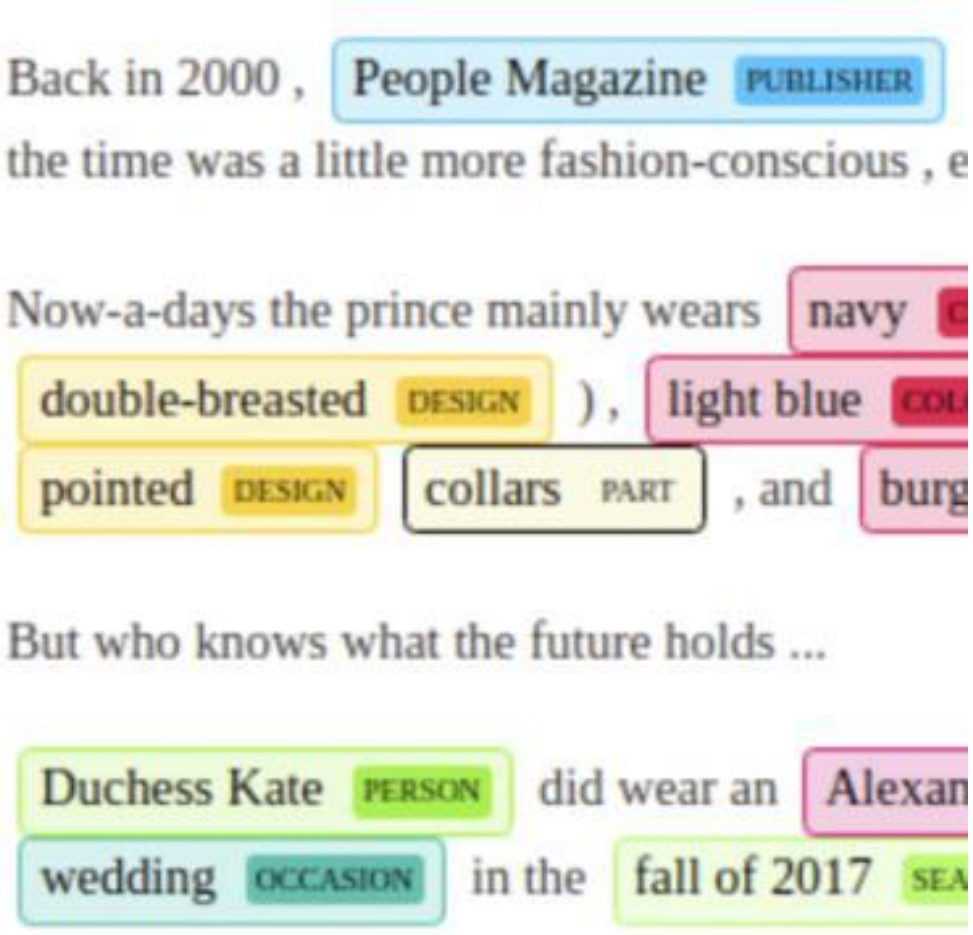


CSCE604131 | Jejaring Semantik (Knowledge Graphs)

KG Construction

Fariz Darari, Ph.D.

Named Entity Recognition



https://paperswithcode.com/task/named-entity-recognition-ner

- Named entity recognition (NER) is the task of tagging entities in text with their corresponding type.
- Approaches typically use BIO notation, which differentiates the beginning (B) and the inside (I) of entities. O is used for non-entity tokens.

(Named) Entity Linking

- Entity Linking (EL) is the task of recognizing (cf. Named Entity Recognition) and disambiguating (Named Entity Disambiguation) named entities to a knowledge graph (e.g., Wikidata, DBpedia, or YAGO).

- EL can be split into two classes of approaches:
  - End-to-End: processing a piece of text to extract the entities (i.e., Named Entity Recognition) and then disambiguate these extracted entities to the correct entry in a given KG (e.g., Wikidata, DBpedia, YAGO).
  - Disambiguation-Only: contrary to the first approach, this one directly takes gold standard named entities as input and only disambiguates them to the correct entry in a given KG.

Credits: http://nlpprogress.com/english/entity_linking.html

OpenTapioca$^{\beta}$

OpenTapioca annotates text with locations, organizations and people from Wikidata.
It is synchronous with Wikidata.
This is a prototype. See the GitHub project for more info.

Associated Press writer Julie Pace contributed from Washington.

Annotate

[Associated Press] writer [Julie Pace] contributed from [Washington].

Washington, D.C. (Q61)
capital city of the United States
Rank: 14.18, phrase: 9.86
Statements: 234, sitelinks: 266
Score: 0.27103144743233265

Relation Extraction

- Relation Extraction is the task of predicting attributes and relations for entities in a sentence.
- For example, given a sentence:

  "Barack Obama was born in Honolulu, Hawaii"

  A relation classifier aims at predicting the relation of "bornInCity".
- Relation Extraction is the key component for building relation knowledge graphs, and it is of crucial significance to NLP applications such as structured search, sentiment analysis, question answering, and summarization.

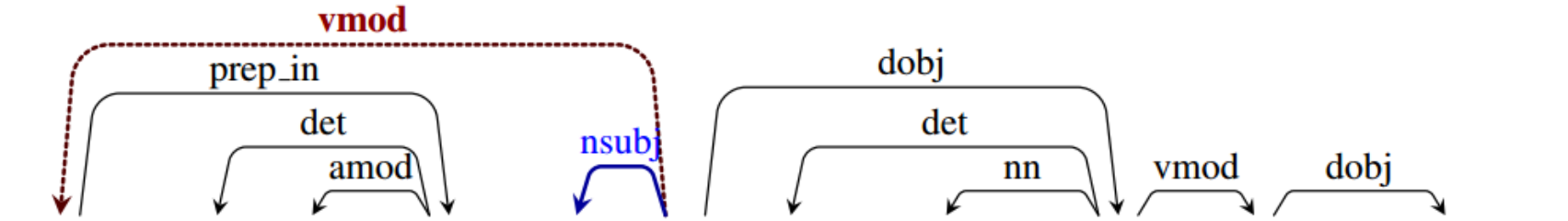
Credits: https://paperswithcode.com/task/relation-extraction/codeless

Open Information Extraction

- Open information extraction (open IE) refers to the extraction of relation tuples, typically binary relations, from plain text, such as (Mark Zuckerberg; founded; Facebook).
- The central difference from other information extraction is that the schema for these relations does not need to be specified in advance; typically the relation name is just the text linking two arguments.
- For example, “Barack Obama was born in Hawaii” would create a triple (Barack Obama; was born in; Hawaii), corresponding to the open domain relation: was-born-in(Barack-Obama, Hawaii).

Credits: https://nlp.stanford.edu/software/openie.html

Open Information Extraction



(input)

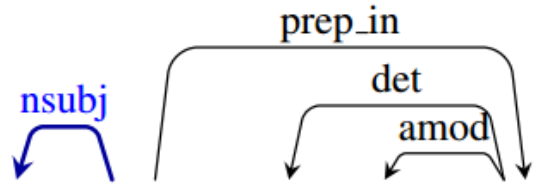
↓

she took the midnight train going anywhere
Born in a small town, she took the midnight train
Born in a town, she took the midnight train

she took the midnight train
she took midnight train
. . .

↓

(she; took; midnight train)



(extracted clause)

↓

she Born in small town
she Born in a town
she Born in town

↓

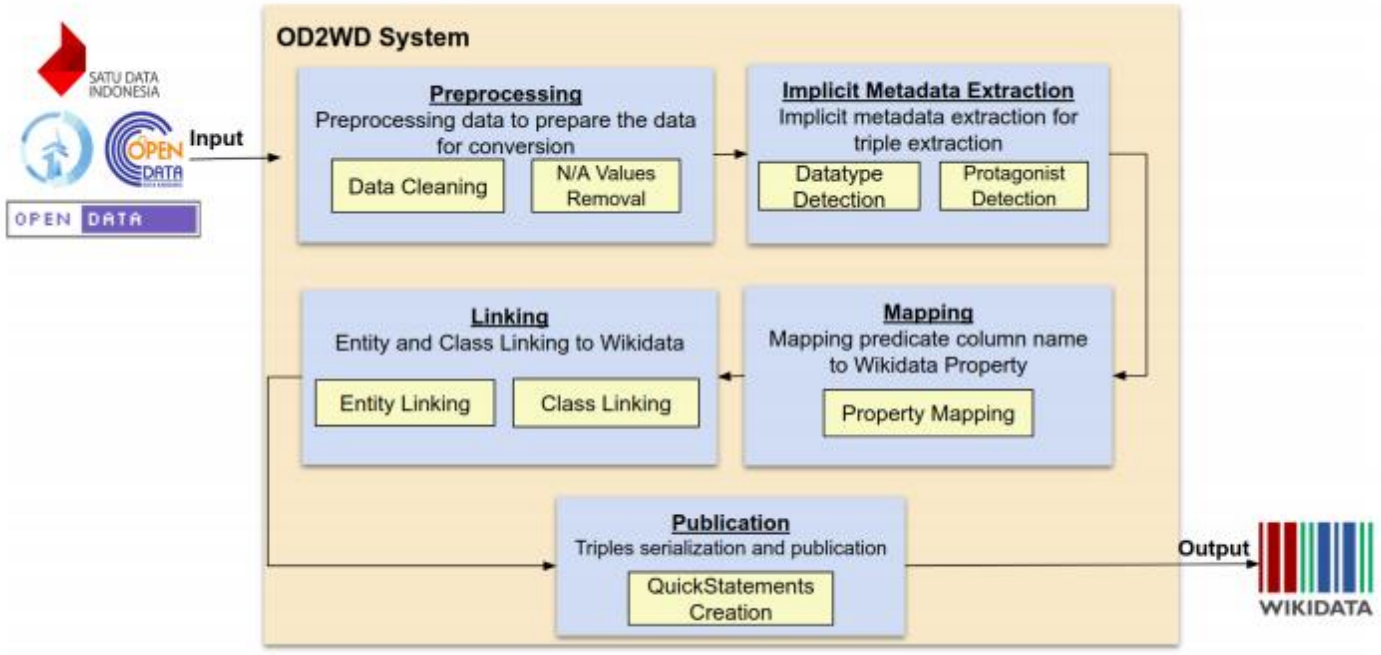
(she; born in; small town)
(she; born in; town)

NELL: Never Ending Language Learning

- Can computers learn to read? Since January 2010, a computer system called NELL (Never-Ending Language Learner) has been running continuously, attempting to perform two tasks each day:
  1) First, it attempts to "read," or extract facts from text found in hundreds of millions of web pages (e.g., playsInstrument(George_Harrison, guitar)).
  2) Second, it attempts to improve its reading competence, so that tomorrow it can extract more facts from the web, more accurately.
- So far, NELL has accumulated over 50 million candidate beliefs by reading the web, and it is considering these at different levels of confidence. NELL has high confidence in 2,810,379 of these beliefs — these are displayed at http://rtw.ml.cmu.edu/rtw/

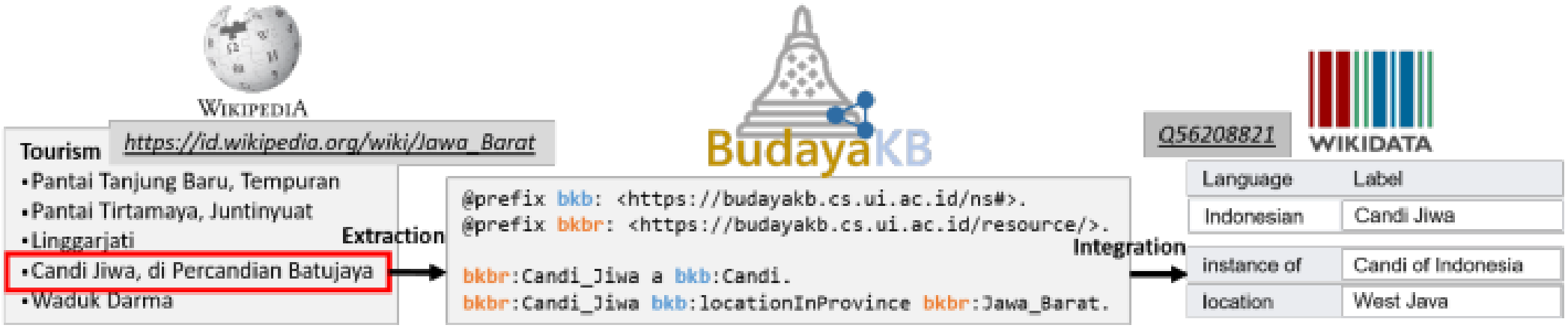
OD2WD: From Open Data to Wikidata

	A	B	C	D	E
1	Nama Sekolah	Alamat	Wilayah	Telepon	Jenis Sekolah
2	SMPN 3	JL. MANGGARAI UTARA IV/6 MANGGARAI	Jakarta Selatan	021 8303844	SMP
3	SMPN 11	JLN.KERINCI BLOK.E KEB.BARU JAK-SEL	Jakarta Selatan	021 7221665	SMP
4	SMPN 12	JLN.WIJAYA IX NO.50 KEB.BARU JAK-SEL	Jakarta Selatan	021 7208095	SMP
5	SMPN 13	JL.TIRTAYASA RAYA	Jakarta Selatan	021 7262939	SMP
6	SMPN 15	JL. PROF. DR. SOEPOMO, SH MENTENG DALAM	Jakarta Selatan	021 8312669	SMP
7	SMPN 16	JL.PALMERAH BARAT NO.59	Jakarta Selatan	021 5483415	SMP
8	SMPN 19	JLN.BUMI BLOK.E NO.21 KEB.BARU JAK-SEL	Jakarta Selatan	021 7250219	SMP
9	SMPN 29	JLN.BUMI BLOK.E	Jakarta Selatan	021 7247493	SMP
10	SMPN 31	JL.PENINGGARAN BARAT III	Jakarta Selatan	021 7239730	SMP



Credits: http://ceur-ws.org/Vol-2459/paper1.pdf

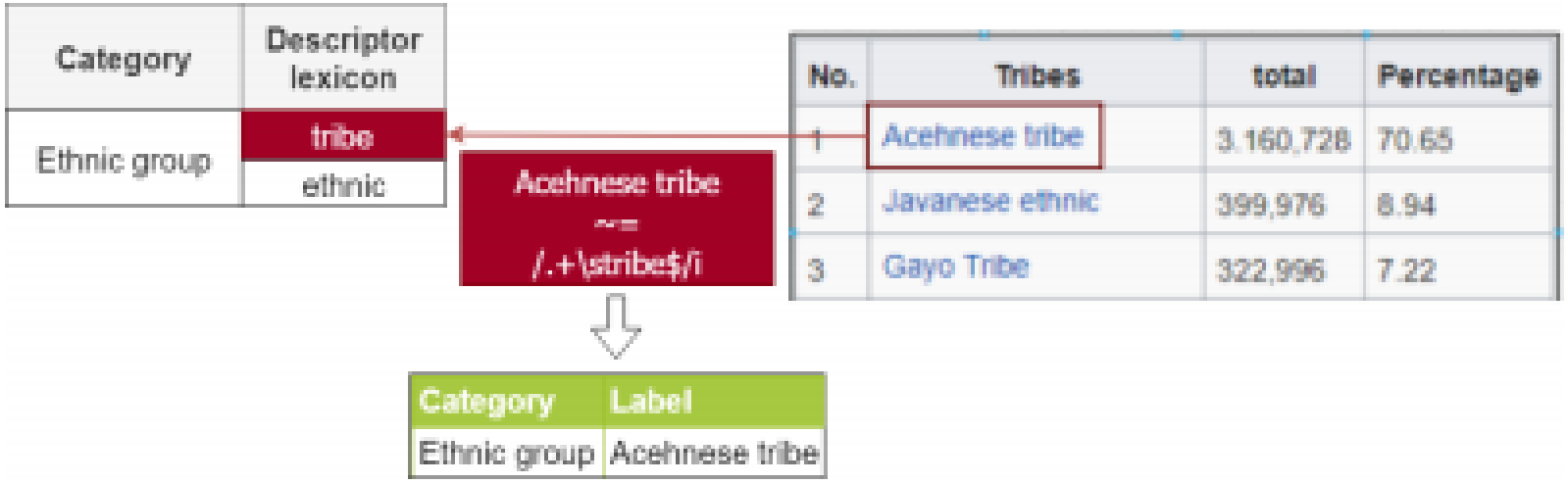
BudayaKB



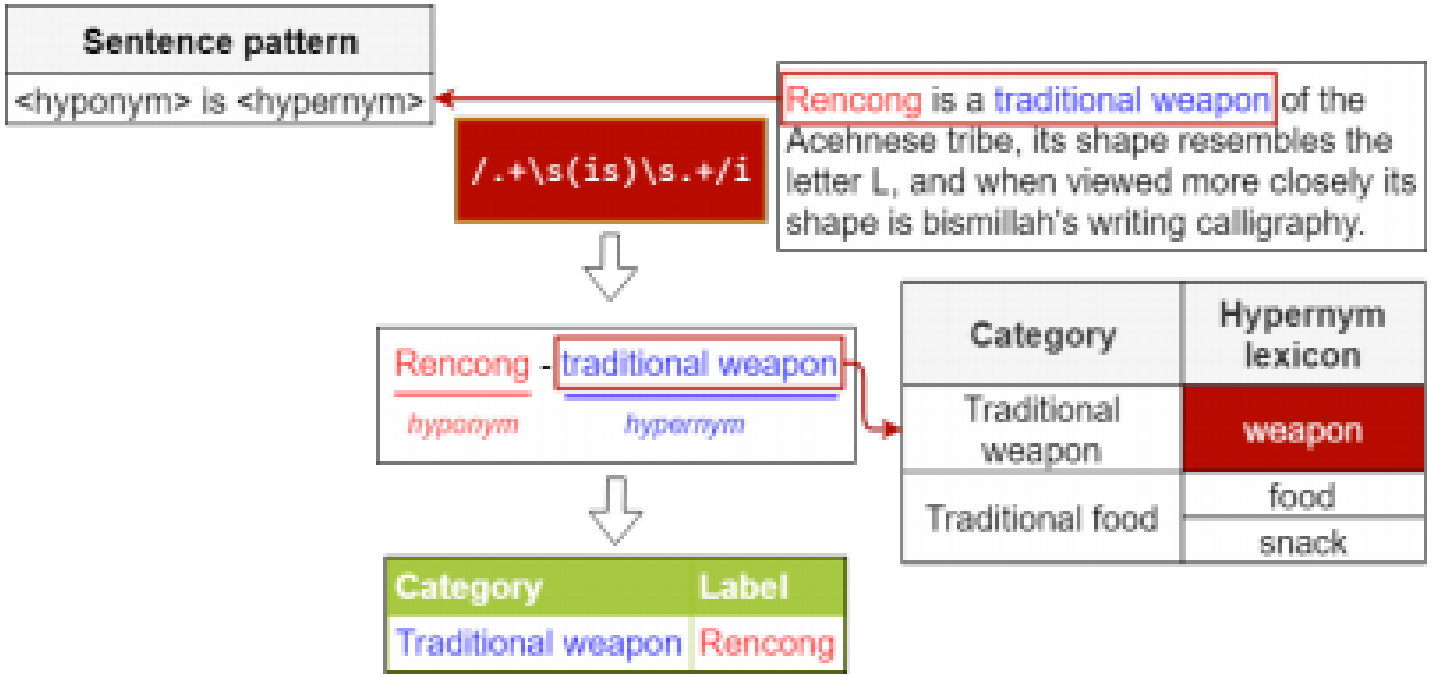
Credits: https://dl.acm.org/doi/10.1145/3326467.3326487

Compiled by Fariz Darari

BudayaKB



BudayaKB

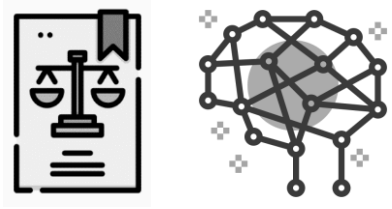


Lex2KG: From legal documents to knowledge graph - Background

- Laws are typically available in the PDF format.
- The advancement of digital and AI technologies calls for machine-readable laws in order to support complex, automated legal processing (= Legal 4.0).
- Knowledge graph provides structured descriptions about entities of interest and relationships among them.
- The European Legislation Identifier (ELI) is an initiative to achieve a common representation of legislation metadata across European Union (EU) countries, inspiring the development of the legal extension of Google's Schema.org.
- Link to download Lex2KG preprint: click here

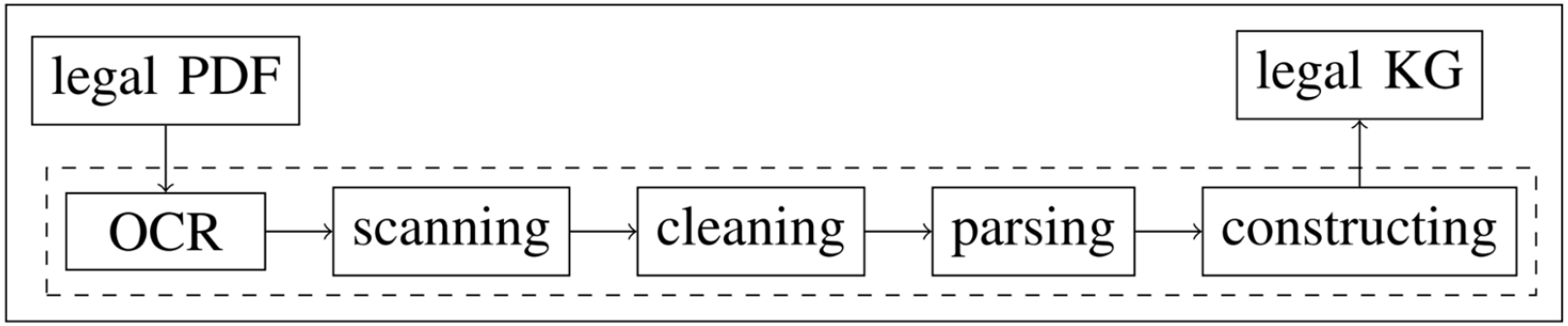
Lex2KG: From legal documents to knowledge graph

- A framework for automated conversion of legal documents to a legal KG.
- Aims to extract the metadata, structure, and textual content of legal documents, particularly laws, as well as relationships among legal resources, and present the extracted information as a (legal) KG.



Icons by: iconfinder

Lex2KG: Architecture



Compiled by Fariz Darari

Lex2KG: Schema

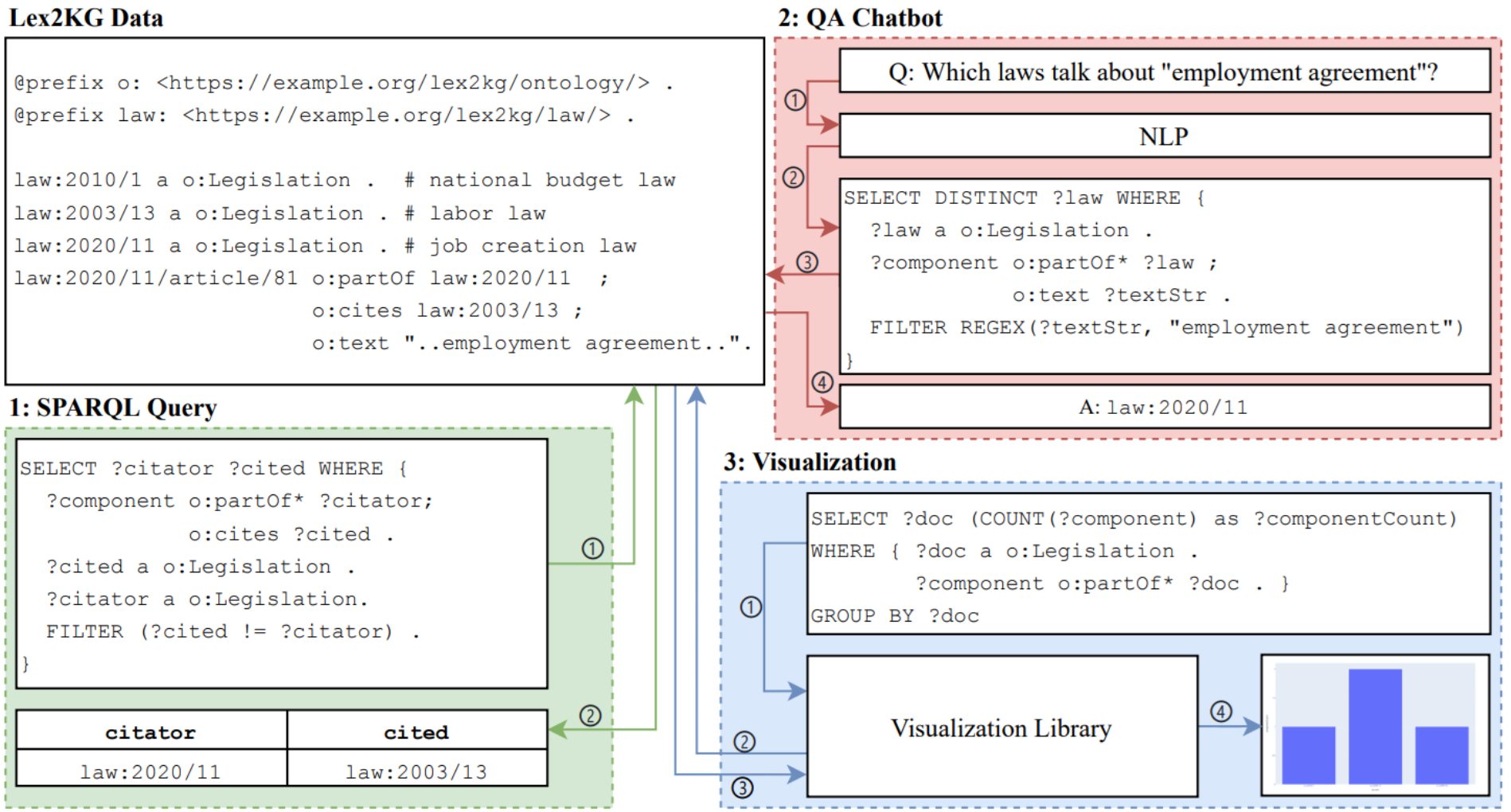
- Used as blueprint for KG construction
- Modeled using RDF Schema (RDFS), a W3C standard
- Facilitating the listing of the terms used in the legal KGs:
  - 15 classes (e.g., Legislation, Article)
  - 28 properties (e.g., partOf, cites, legislationType)
  - 6 (constant) individuals (e.g., Law – as a value of legislationType).
- Aligned to Schema.org to support interoperability

Lex2KG: Data

Legal KG contains the data of 784 Indonesian laws with a total size of 1,163,051 RDF triples successfully constructed from legal PDFs.

Information Type	#Triples
Metadata	6,869
Structure	714,888
Text	152,301
Citations in the same law	22,829
Citations between laws	1,401
Citations to non-existing resources	7,095
Amendments	1,252
Class assignments	256,416

Lex2KG: Use Case Evaluation

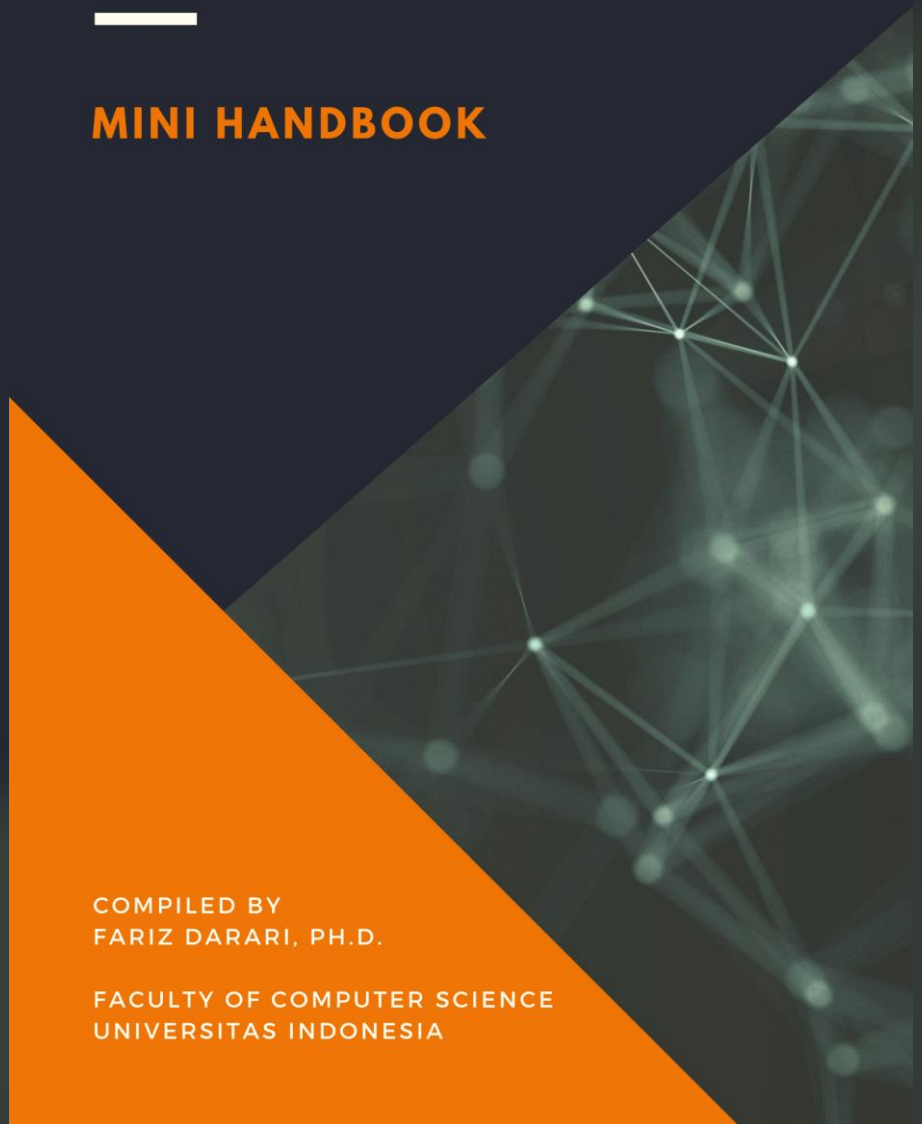


* For readability purposes, "\/" in URIs is written as simply "/". Also, data written in Indonesian is translated to English.

Supported by the Kampus Merdeka grant of
Ministry of Education, Culture, Research, and Technology of
Republic of Indonesia

KNOWLEDGE GRAPHS

MINI HANDBOOK



COMPILED BY
FARIZ DARARI, PH.D.

FACULTY OF COMPUTER SCIENCE
UNIVERSITAS INDONESIA