# Shaping other types of Knowledge Graphs

**Jose Emilio Labra Gayo**
WESO Research group
University of Oviedo, Spain

# Contents

- Introduction to Knowledge graphs
- Types of Knowledge Graphs:
  - RDF, Property graphs, Wikibase, RDF-Star
- Shaping RDF: ShEx & SHACL
- Shaping other types of Knowledge graphs:
  - Wikibase and Wikidata graphs
  - Property Graphs
  - RDF-1.2
- Applications:
  - Inferring shapes from data, Knowledge Graphs Subsets, etc.

# Shaping other types of knowledge graphs

We present some work on extending ShEx for:

- Wikidata and Wikibase graphs: WShEx
- RDF Star (RDF 1.2): ShEx-Star
- Property graphs: P-ShEx
- RDF with nodes as properties: ShEx-N

**Note**: This work is more theoretical and *work in progress*

We start reviewing ShEx from a more theoretical point of view

* Although we use ShEx in the presentation, similar extensions could be done using SHACL

# Conceptual model of RDF graphs

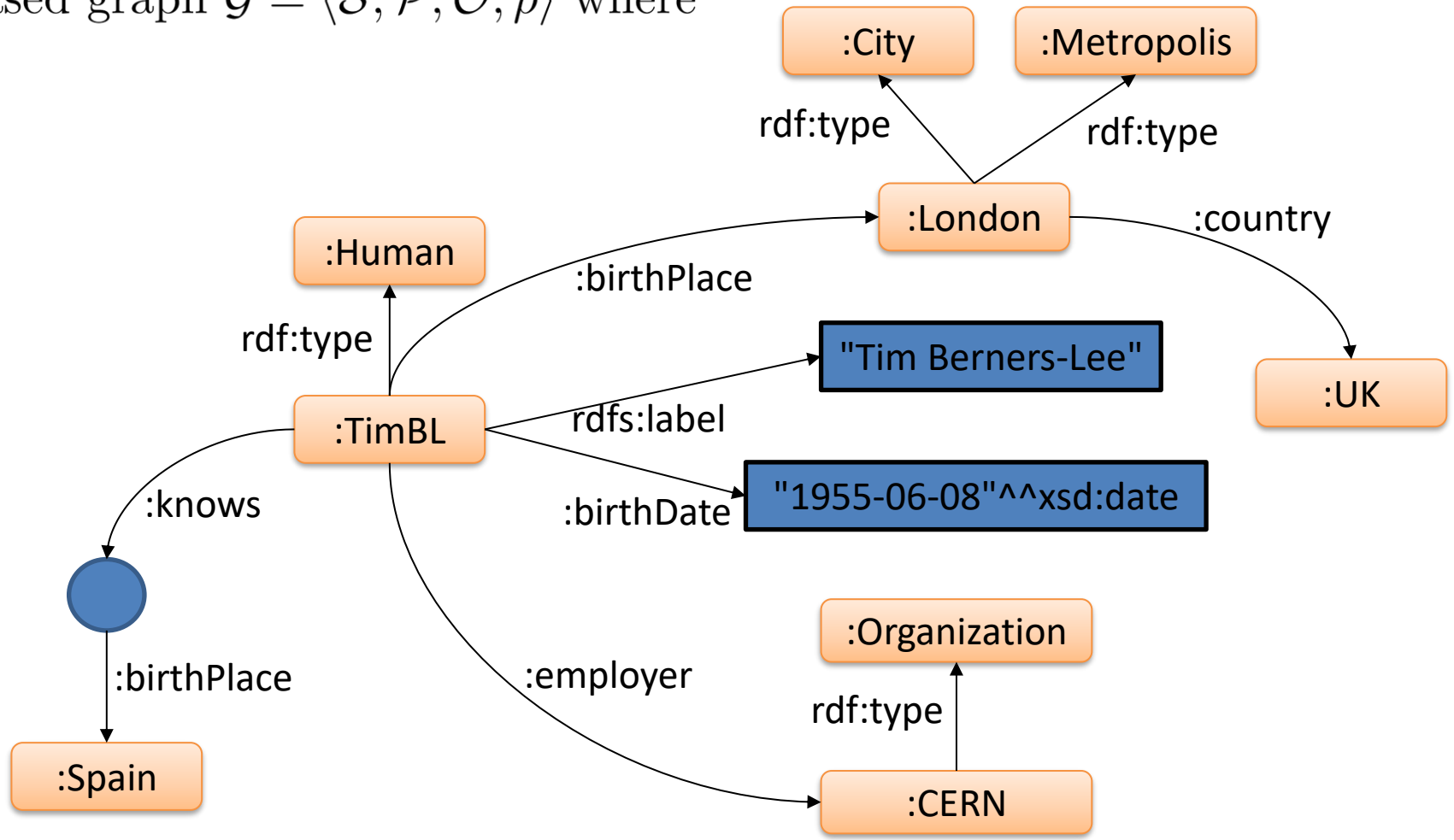
Given a set of IRIs $\mathcal{I}$, a set of blank nodes $\mathcal{B}$ and a set of literals $Lit$ an *RDF graph* is a triple based graph $\mathcal{G} = \langle \mathcal{S}, \mathcal{P}, \mathcal{O}, \rho \rangle$ where

$\mathcal{S} = \mathcal{I} \cup \mathcal{B}$,

$\mathcal{P} = \mathcal{I}$,

$\mathcal{O} = \mathcal{I} \cup \mathcal{B} \cup Lit$

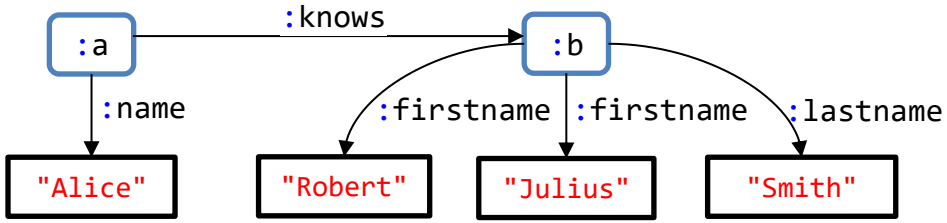
$\rho \subseteq \mathcal{S} \times \mathcal{P} \times \mathcal{O}$



# ShEx example

A Shape mainly describes the neighbourhood of a node

- Triple Expressions
- Node constraints
- Cardinality




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

<Person> {
 :firstname xsd:string * ;
 :lastname  xsd:string
 :knows     @<Person>  *
}
```

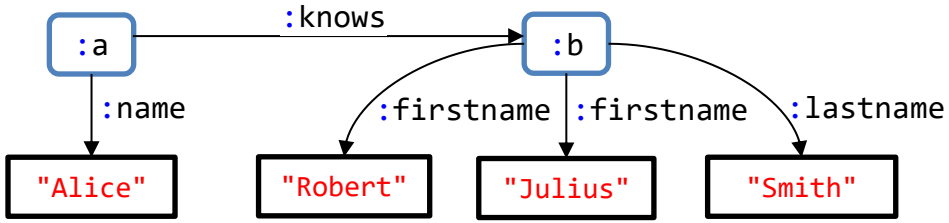
☹
```
prefix : <http://example.org/>

:a :name      "Alice" ;
   :knows     :b      .
```

☺
```
:b :firstname "Robert", "Julius" ;
   :lastname  "Smith" .
```

# ShEx example

ShEx accepts regular expresión operators on triple expressions

EachOf (;), OneOf (|), grouping




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

<Person> {
    ( :name      xsd:string
    | :firstName xsd:string * ;
      :lastname  xsd:string
    ) ;
    :knows       @<Person> *
}
```

☺
```
prefix : <http://example.org/>

:a :name      "Alice" ;
   :knows     :b      .
:b :firstname "Robert", "Julius" ;
   :lastname  "Smith" .
```

☺
```
:a :name      "Alice" ;
   :knows     :b      .
:b :firstname "Robert", "Julius" ;
   :lastname  "Smith" .
```

☹
```
:c :name      "Carol" ;
   :firstname "Carol" ;
   :knows     :b      .
```

# Abstract syntax of *Simplified* ShEx

ShEx schema is a tuple $<L, \delta>$

where $L$ = set of labels and $\delta$: $L \rightarrow se$

| | | | |
|---|---|---|---|
| $se$ | ::= | cond | Basic boolean condition on nodes (node constraint) |
| | \| | $s$ | Shape |
| | \| | $se_1$ AND $se_2$ | Conjunction of $se_1$ and $se_2$ |
| | \| | $@l$ | Shape label reference for $l \in L$ |
| | \| | CLOSED $\{te\}$ | Closed shape |
| | \| | $\{te\}$ | Open shape |
| $te$ | ::= | $te_1; te_2$ | Each of $te_1$ and $te_2$ |
| | \| | $te_1 \mid te_2$ | Either $te_1$ or $te_2$ |
| | \| | $te*$ | Zero or more $te$ |
| | \| | $\_ \xrightarrow{p} se$ | Outgoing Triple with predicate $p$ and object conforming to $se$ |
| | \| | $se \xrightarrow{p} \_$ | Incoming triple with predicate $p$ and subject conforming to $se$ |
| | \| | $\epsilon$ | Empty triple expression |

# Example with ShEx abstract syntax

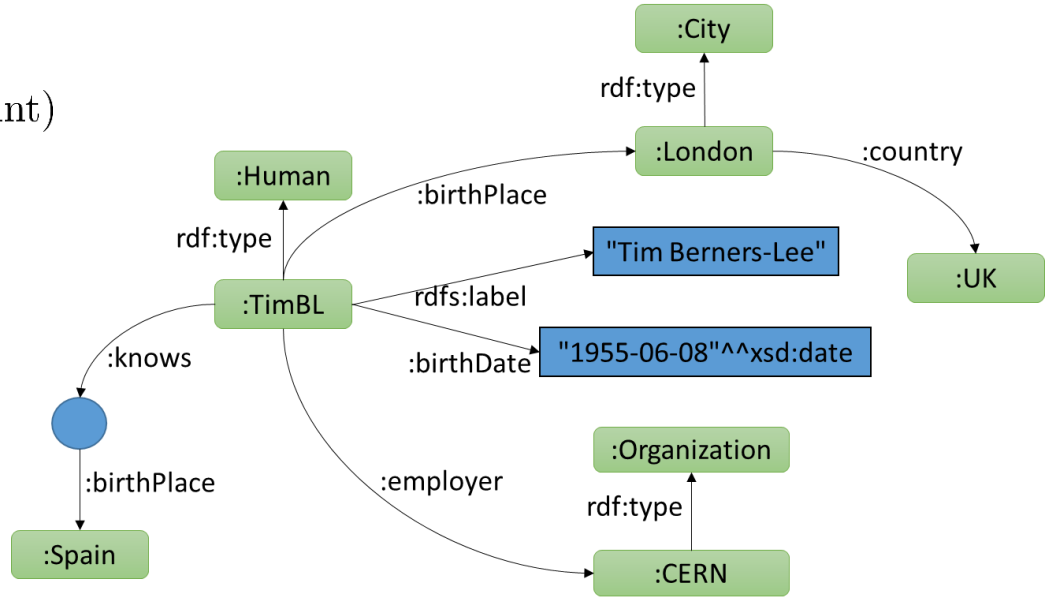
```
prefix :    <http://example.org/>
prefix xsd: <http://.../XMLSchema#>

<Person> {
    ( :name   xsd:string
    | :firstName xsd:string *;
      :lastname xsd:string
    ) ;
    :knwows @<Person> *
}
```

$$
\begin{array}{rcl}
L & = & \{Person\} \\
\delta(Person) & = & \{ \quad (\_ \xrightarrow{name} String \mid \_ \xrightarrow{fistname} String*; \_ \xrightarrow{lastname} String); \\
 & & \quad \_ \xrightarrow{knows} @Person\ * \\
 & & \}
\end{array}
$$

# Simplified ShEx for RDF

A ShEx Schema is a tuple $\langle \mathcal{L}, \delta \rangle$ where
$\mathcal{L}$ set of shape labels
$\delta : \mathcal{L} \rightarrow se$

| | | | |
|---|---|---|---|
| $se$ | ::= | cond | Basic boolean condition on nodes (node constraint) |
| | \| | $s$ | Shape |
| | \| | $se_1$ `AND` $se_2$ | Conjunction |
| | \| | $@l$ | Shape label reference for $l \in \mathcal{L}$ |
| $s$ | ::= | `CLOSED` $\{te\}$ | Closed shape |
| | \| | $\{te\}$ | Open shape |
| $te$ | ::= | $te_1; te_2$ | Each of $te_1$ and $te_2$ |
| | \| | $te_1 \mid te_2$ | Some of $te_1$ or $te_2$ |
| | \| | $te*$ | Zero or more $te$ |
| | \| | $\epsilon$ | Empty triple expression |
| | \| | $\llcorner \xrightarrow{p} @l$ | Triple constraint with predicate $p$ |




$$
\begin{array}{rcl}
\mathcal{L} & = & \{ \; Person, Place, Country, Organization, Date \} \\
\delta(Person) & = & \{ \; \llcorner \xrightarrow{birthDate} @Date; \; \llcorner \xrightarrow{birthPlace} @Place; \\
 & & \llcorner \xrightarrow{employer} @Organization* \; \} \\
\delta(Place) & = & \{ \; \llcorner \xrightarrow{country} @Country \} \\
\delta(Country) & = & \{ \; \} \\
\delta(Organization) & = & \{ \; \} \\
\delta(Date) & = & \text{xsd:Date}
\end{array}
$$

# Contents

- Introduction to Knowledge graphs
- Types of Knowledge Graphs:
  - RDF, Property graphs, Wikibase, RDF-Star
- Shaping RDF: ShEx & SHACL
- Shaping other types of Knowledge graphs:
  - Wikibase and Wikidata graphs
  - Property Graphs
  - RDF-1.2
- Applications:
  - Inferring shapes from data, Knowledge Graphs Subsets, etc.

# Wikidata

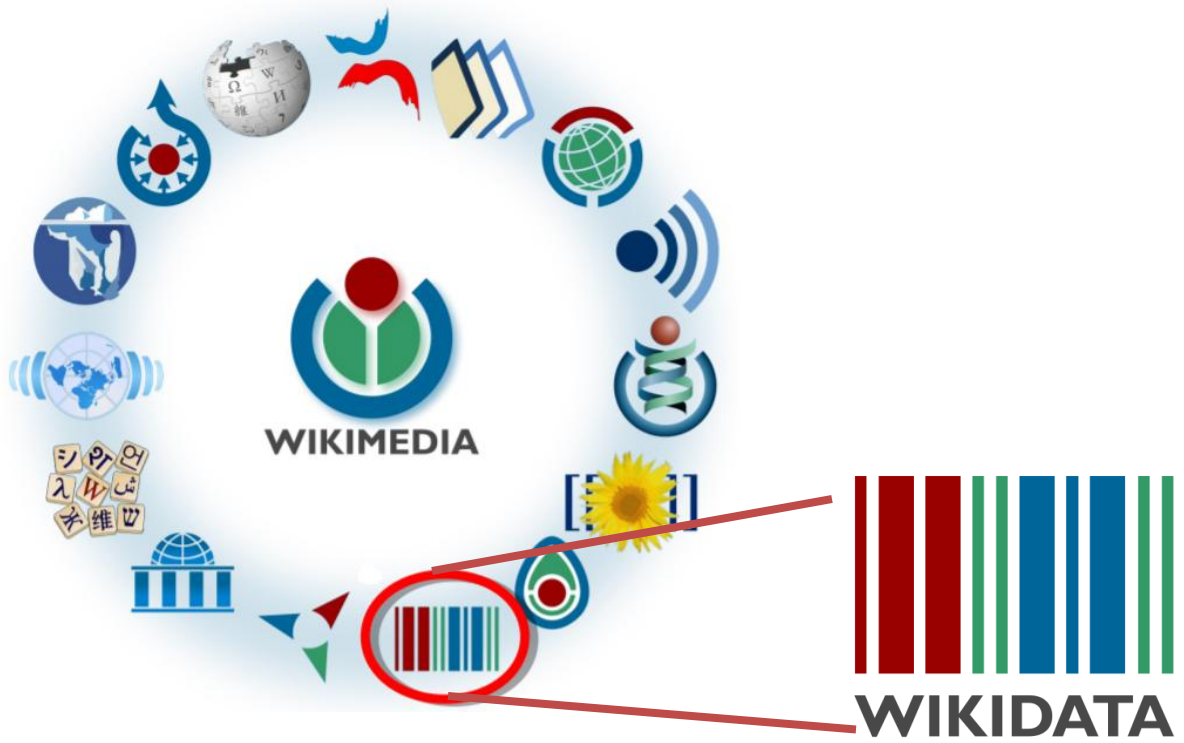
Created in 2012 as a collaborative knowledge graph

https://www.wikidata.org/

Developed and supported by Wikimedia Deutschland

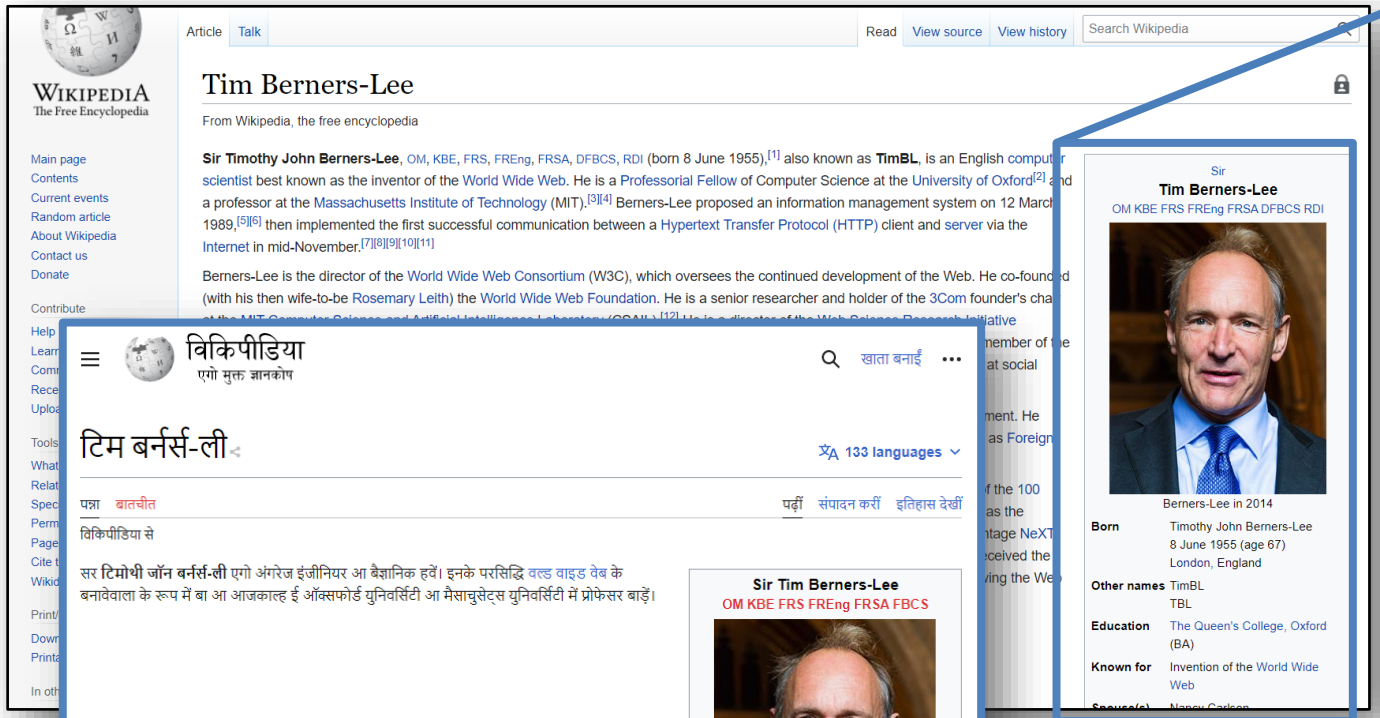
Initial goal:

Support multilingual infoboxes in Wikipedia



# Wikidata as an example

English Wikipedia page of Tim Berners-Lee

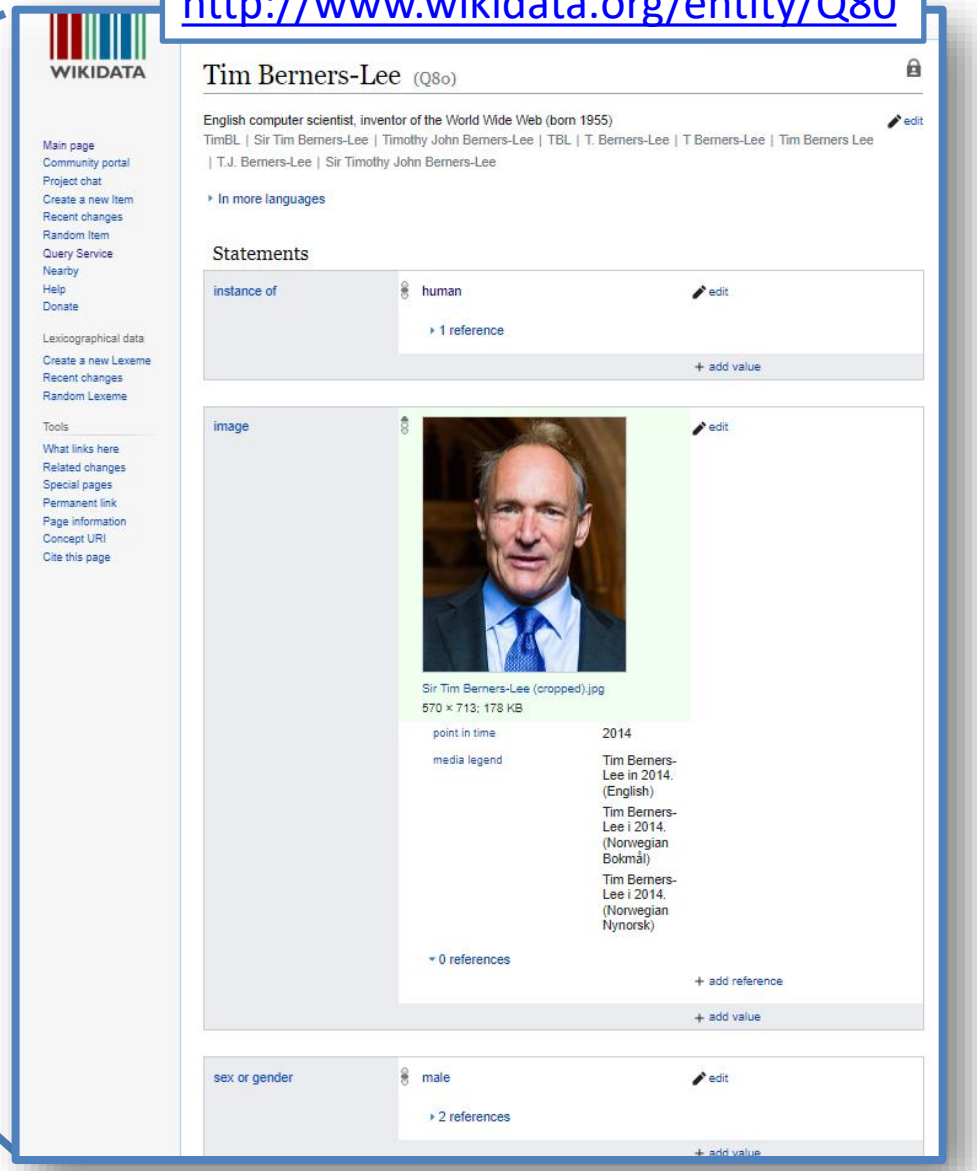






Bihari Wikipedia of Tim Berners-Lee

http://www.wikidata.org/entity/Q80



# Wikidata as a Commons Knowledge Graph

- Wikidata has proven a very successful Project
  - Showcases Semantic web and linked data
    - SPARQL query endpoint
    - Linked data browsing
  - Large adoption
    - 114,425,761 ítems, 2,271,032,314 edits (11/2024): https://www.wikidata.org/wiki/Wikidata:Statistics
- Free and open license: CC0
- Multilingual, collaborative
- Open Wikidata Query Service: Public SPARQL endpoint
  - Dumps freely available
- Nice applications like Scholia: https://scholia.toolforge.org/
  - Wikidata can be a data hub for other apps: IMDB, Wolfram, Reddit, Bionomia, iNaturalist, …
- Software that supports Wikidata = Wikibase

# Wikibase

- Wikibase: Software suite that implements Wikidata (https://wikiba.se/)
  - Set of extensions of Media Wiki (https://www.mediawiki.org/wiki/Wikibase)
    - Implemented in PHP (backend) + Javascript (frontend)
  - Can be hosted locally through Docker
  - Also available Cloud service: Wikibase.cloud (https://www.wikibase.cloud/)
- Wikibase instance: Application using Wikibase software
  - Examples:
    - Rhizome: https://artbase.rhizome.org/
    - enslaved.org: https://enslaved.org/
  - More: Wikibase.world

# Wikibase data model

The Wikibase data model mainly consists of:

- Entities (Items and properties), and
- Statements about entities

More information:

- Reference: https://www.mediawiki.org/wiki/Wikibase/DataModel
- Primer: https://www.mediawiki.org/wiki/Wikibase/DataModel/Primer

# Wikibase data model: Entities

Entities can be

- Items (identified by Qxxx), example: http://www.wikidata.org/entity/Q80
- Properties (identified by Pxxx), example: http://www.wikidata.org/entity/P19
  - Properties have an associated datatype (several built-in datatypes)
    - **Item**: Example, example: P19 (*birthplace*)
    - **Geographic locations**, example: P625 (*coordinate location*), contains latitude, longiture, precisión, coordinate system)
    - **Quantity**, example: P1082 (*population*), contains: value, lower bound, higher bound, unit
    - **Dates and times**, example: P569 (*date of birth*), contain: time, precision, before, after, timezone, …
    - **Monolingual text**, example: P1477 (*birth name*)
    - … others: see reference

Entities can also have lexical information

Multilingual labels, descriptions and aliases

Reference:
https://www.mediawiki.org/wiki/Wikibase/DataModel

# Wikibase data model: Statements

A statement consists of:

Claim:

- Property
- Value: Declaration of the posible value (snak in wikibase terms)
  - One specific value of the property datatype
  - `no_value`
  - `some_value` (unknown)
- Zero or more qualifiers (list of property-value pairs)

Zero or more referencedeclaration: preferred, normal, deprecateds (list of property-value pairs)

Rank

# Wikibase data model: JSON serialization

The Wikibase dumps are exported as JSON following the data model

- Each line in the dump contains information about an item and its statements
- JSON representations can directly be obtained as:
  - https://www.wikidata.org/wiki/Special:EntityData/Q80.json

Example:


```
{ "id": "Q80", "type": "item",
  "labels": {"en": {"language": "en", "value": "Tim Berners-Lee" }, . . . },
  "descriptions": { . . . },
  "aliases": { . . . },
  "claims": { "P19": [
     { "id": "q60$50...",
       "mainsnak": { "snaktype": "value", "datatype": "wikibase-item",
         "datavalue": { "value": { "entity-type": "item", "id": "Q84", "numeric-id": 84 },  "type": "wikibase-entityid" }},
       "type": "statement",
       "rank": "normal",
       "qualifiers": { . . . },
       "references": [ . . . ]
     }
    ]
    . . .
  }, "sitelinks": { . . .}, "lastrevid": 2265292559,  "modified": "2024-10-26T11:36:19Z"
}
```


More information: https://doc.wikimedia.org/Wikibase/master/php/docs_topics_json.html

# Wikibase datamodel: RDF serialization

Wikibase offers an RDF serialization of each entity

Several ways to get the RDF serialization*:

- SPARQL endpoint: Query service
- RDF Dumps
- Directly, example: https://www.wikidata.org/wiki/Special:EntityData/Q80.ttl

The RDF dump format is defined in: RDF_Dump_Format specification

OWL Wikibase ontology: http://wikiba.se/ontology

Several namespaces:

wd: for ítems

wdt: for properties

…

Custom reification model to serialize qualifiers and references

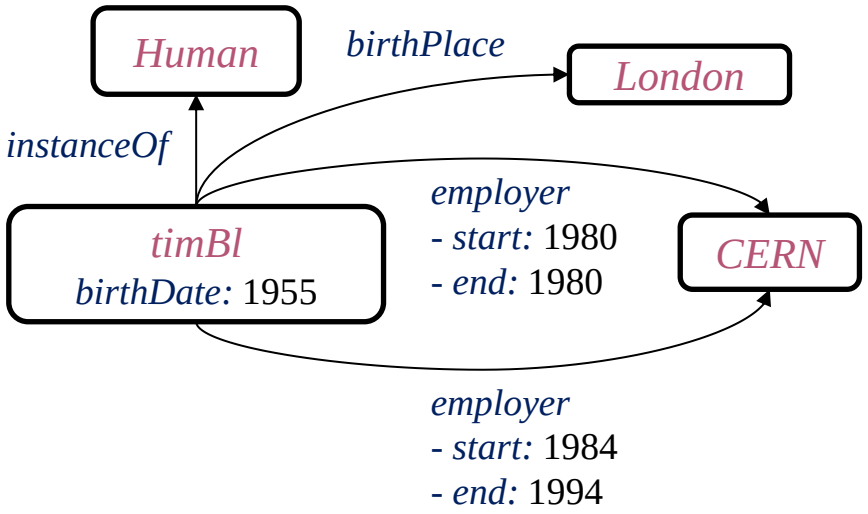
Serialization of values from compounded datatypes requires several triples

*There can be small differences between the RDF serializations as described here

# Wikibase RDF serialization (example)

## Simplified RDF dump

```
wd:Q80 a wikibase:Item ;
  wdt:P31  wd:Q5 ;
  wdt:P19  wd:Q84 ;
  wdt:P569 "1955-06-08T00:00:00Z"^^xsd:dateTime ;
  wdt:P108 wd:Q42944 ;
  p:P108   s:Q80-fcba864c, :Q80-4fe7940f
  #...
  .

:Q80-4fe7940f a wikibase:Statement ;
  ps:P108 wd:Q42944 ;
  pq:P580 "1984-01-01T00:00:00Z"^^xsd:dateTime ;
  pq:P582 "1994-01-01T00:00:00Z"^^xsd:dateTime .

s:Q80-fcba864c a wikibase:Statement ;
    ps:P108 wd:Q42944 ;
    pq:P580 "1980-06-01T00:00:00Z"^^xsd:dateTime ;
    pq:P582 "1980-12-01T00:00:00Z"^^xsd:dateTime .
```



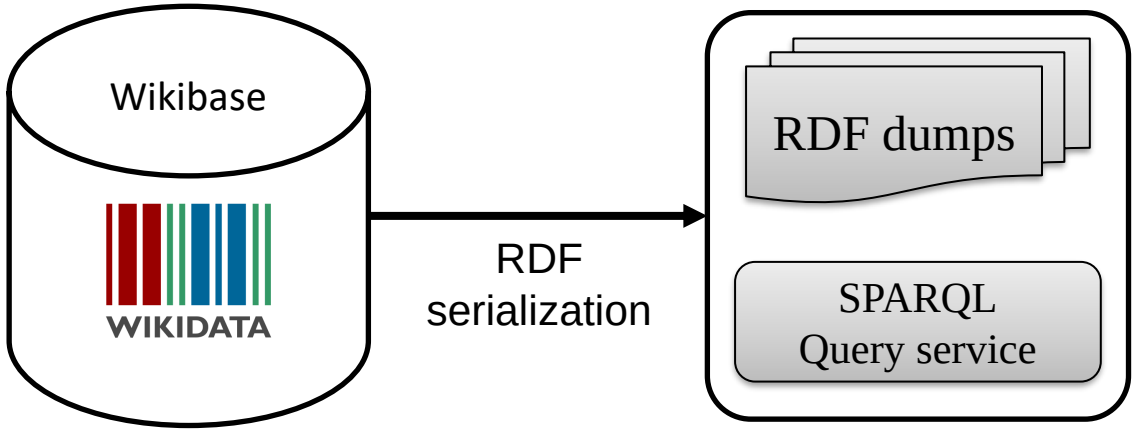

| | |
|---|---|
| *timBl* | wd:Q80 |
| *London* | wd:Q84 |
| *Human* | wd:Q5 |
| *CERN* | wd:Q42944 |
| *birthDate* | wdt:P569 |
| *instanceOf* | wdt:P31 |
| *birthPlace* | wdt:P19 |
| *employer* | wdt:P108 |
| *start* | pq:P580 |
| *end* | pq:P582 |

# Wikibase and SPARQL

Wikibase graphs are also available through SPARQL endpoint

Internally, Wikibase has 2 DBs:

- Relational database (MariaDB)
- RDF Triplestore (Blazegraph)
  - Plans to update



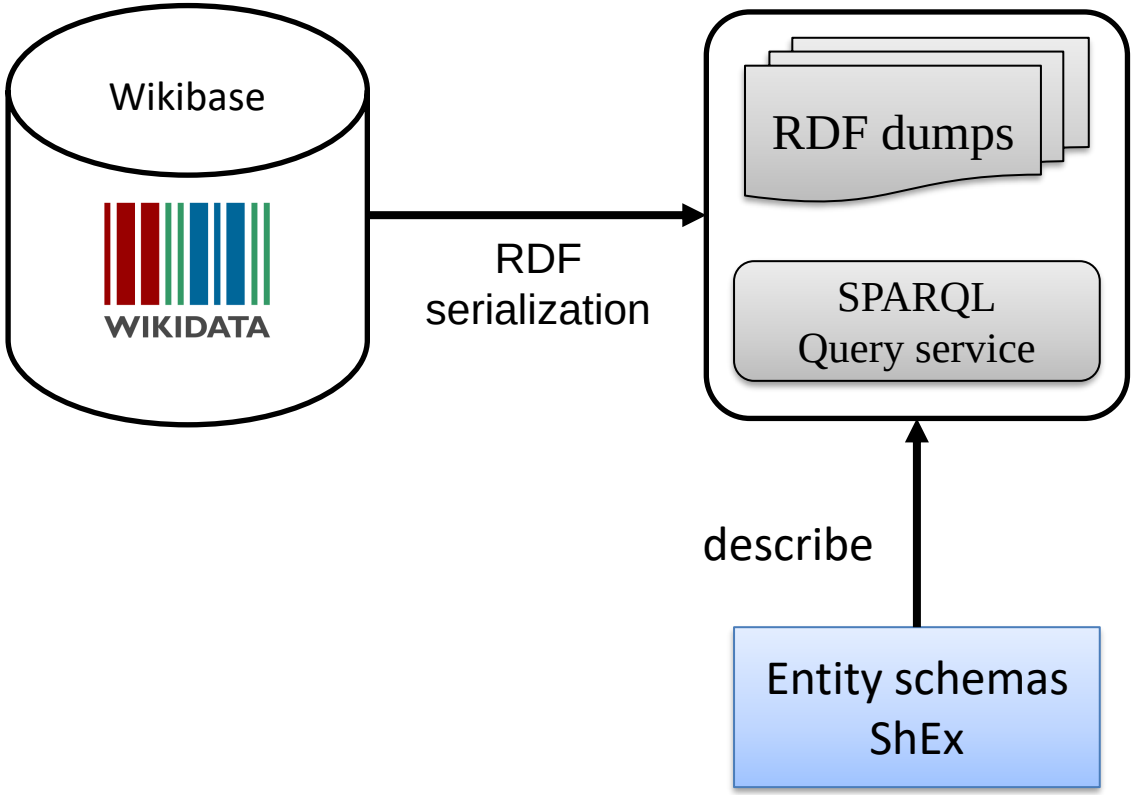

```
select ?name ?date?country where {
 wd:Q80 wdt:P1559 ?name .
 wd:Q80 wdt:P569  ?date .
 wd:Q80 wdt:P19   ?place .
 ?place wdt:P17   ?country
}
```

| ?name | ?date | ?country |
|---|---|---|
| Tim Berners-lee | 1955-06-08 | :UK |

Try it: https://w.wiki/5yGu

# Wikibase and RDF: Entity Schemas

Idea: If we have RDF, we can use ShEx to describe and validate entities



# Entity Schemas

They can be used to describe Wikidata entities

Adopted in 2019 as a new Wikidata namespace Exxx

Example:

https://www.wikidata.org/wiki/EntitySchema:E10

Directory of entity schemas:

https://www.wikidata.org/wiki/Wikidata:Database_reports/EntitySchema_directory

# Example of an Entity Schema

Q80.ttl

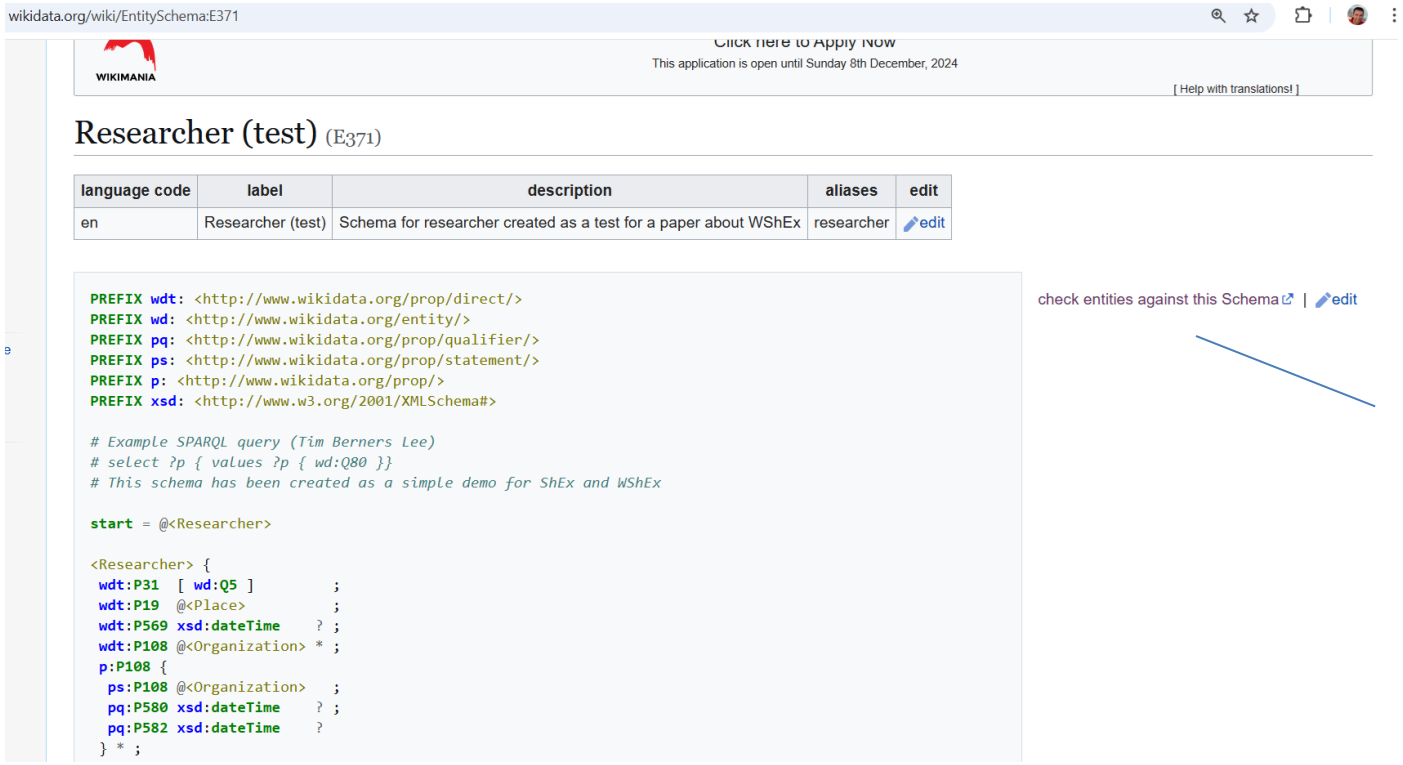
```
wd:Q80 rdf:type wikibase:Item ;
  wdt:P31 wd:Q5 ;
  wdt:P19 wd:Q84 ;
  wdt:P569 "1955-06-08T00:00:00Z"^^xsd:dateTime ;
  wdt:P108 wd:Q42944 ;
  p:P108 s:Q80-fcba864c, :Q80-4fe7940f
  #...
  .

:Q80-4fe7940f rdf:type wikibase:Statement ;
  wikibase:rank wikibase:NormalRank ;
  ps:P108 wd:Q42944 ;
  pq:P580 "1984-01-01T00:00:00Z"^^xsd:dateTime ;
  pq:P582 "1994-01-01T00:00:00Z"^^xsd:dateTime .

s:Q80-fcba864c a wikibase:Statement ;
    wikibase:rank wikibase:NormalRank ;
    ps:P108 wd:Q42944 ;
    pq:P580 "1980-06-01T00:00:00Z"^^xsd:dateTime ;
    pq:P582 "1980-12-01T00:00:00Z"^^xsd:dateTime .
```

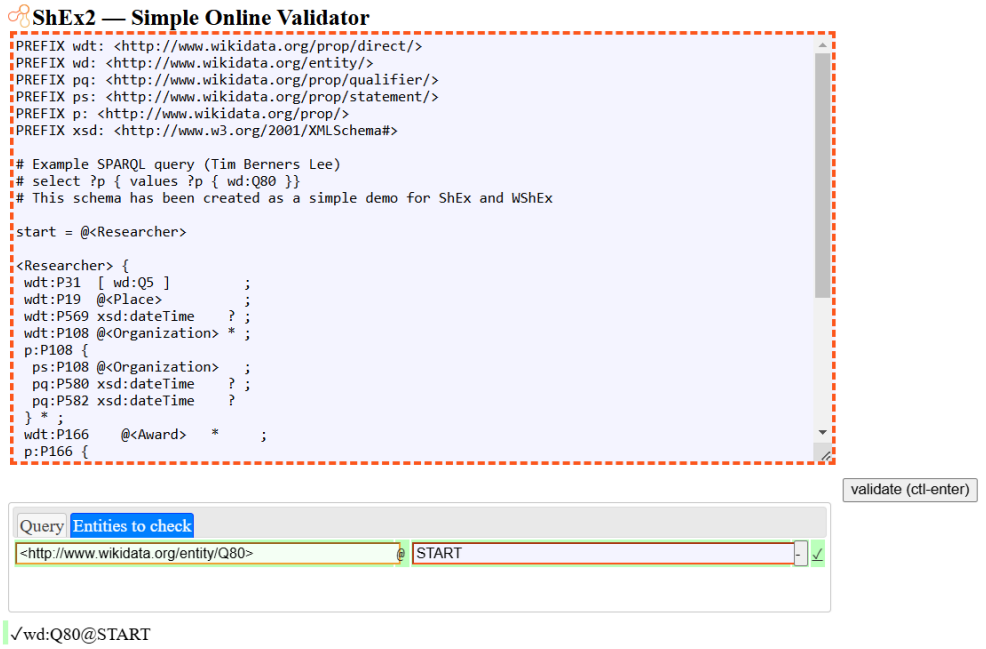
Entity-schema - ShEx

```
PREFIX pq: <.../prop/qualifier/>
PREFIX ps: <.../prop/statement/>
PREFIX p: <.../prop/>
PREFIX wdt: <.../prop/direct/>
PREFIX wd: <.../entity/>
PREFIX xsd: <...XMLSchema#>

<Researcher> {
 wdt:P31  [ wd:Q5 ]          ;
 wdt:P19  @<Place>           ;
 wdt:P569 xsd:dateTime       ;
 wdt:P108 @<Employer>      * ;
 p:P108 { ps:P108 @<Employer>     ;
          pq:P580 xsd:dateTime  ? ;
          pq:P582 xsd:dateTime  *
 } *
}
<Place> { }
<Employer> { }
```

# Using Entity Schemas for validation

Example: https://www.wikidata.org/wiki/EntitySchema:E371



# Wikibase also has property constraints for validation

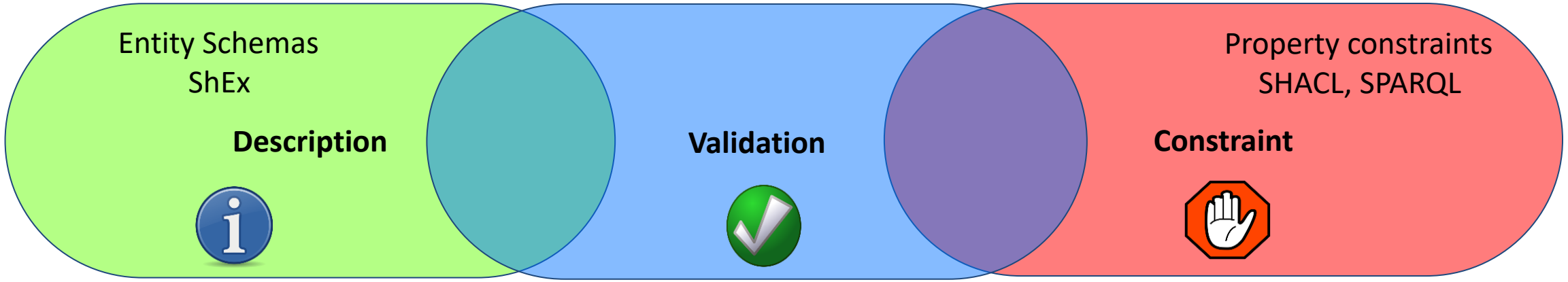
- Property constraints: rules on properties
  - Specify how the properties should be used
    - https://www.wikidata.org/wiki/Help:Property_constraints_portal
  - Constraints are hints, not firm restrictions (help or guidance to the editor)
- Example: single value constraint (Q19474404)
  - P19 (birth_place) in general is expected to conform to SingleValueConstraint
  - Example failing constraint (Noam Chomsky) and example with exception (Louis Seel)
- Property constraints definition/implementation is part of Wikibase

# Entity schemas vs property constraints

Entity schemas contain descriptions of sets of ítems

- Easy to create a new entity schemas: Overlapping entity schemas for different purposes
- Entity schemas ecosystem

Property constraints are rules that can be used to validate or recommend

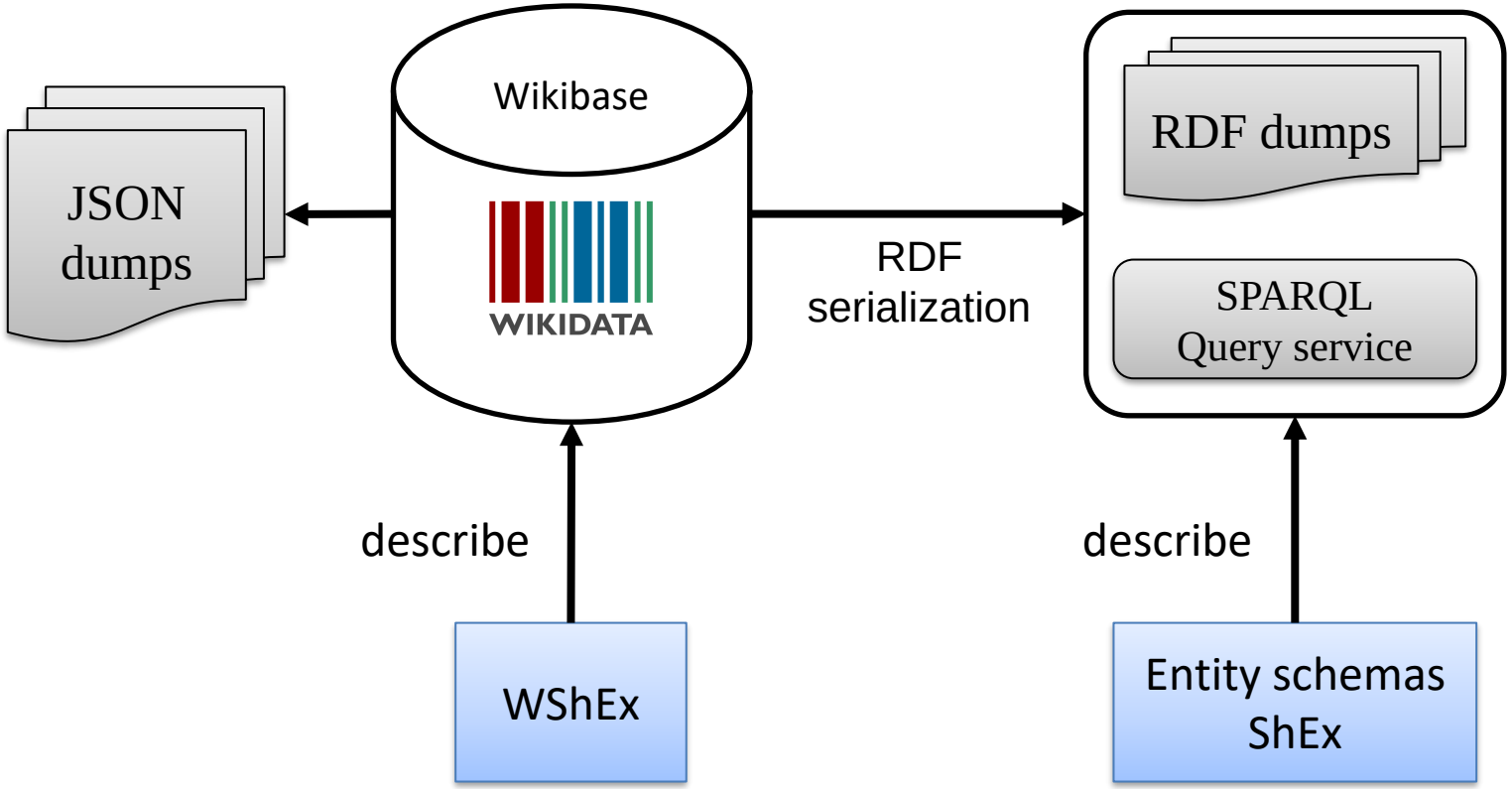



Paper about the relationship between property constraints and SHACL, SPARQL:
Ferranti, Nicolas et al. 'Formalizing and Validating Wikidata's Property Constraints Using SHACL and SPARQL'. 1 Jan. 2024 : 1 – 48.

# WShEx

Although Entity Schemas and ShEx just work, they are indirectly describing Wikibase

WShEx has been proposed as a language to describe the Wikibase data model?




https://www.weso.es/WShEx/

# Wikibase RDF representation


RDF

```
wd:Q80       wdt:P166  wd:Q3320352  ;
             p:P166    s:PQ80-494FA .

s:PQ80-494FA ps:P166   wd:Q3320352 ;
             pq:P585   2002        ;
             pq:P1706  wd:Q62843, wd:Q92935, wd:Q92743.
```

Could be represented as

```
:Q80 :P166   :Q3320352 {|
        :P585   2002;
        :P1796 :Q62843, :Q92935, :Q92743
   |}
```

Inspired by RDF-Star annotated triples syntax

# WShEx: A language to describe and validate Wikibase entities

Jose Emilio Labra Gayo

## Entity-schema - ShEx

```
PREFIX pq: <.../prop/qualifier/>
PREFIX ps: <.../prop/statement/>
PREFIX p: <.../prop/>
PREFIX wdt: <.../prop/direct/>
PREFIX wd: <.../entity/>
PREFIX xsd: <...XMLSchema#>

<Researcher> {
 wdt:P31  [ wd:Q5 ]          ;
 wdt:P166  @<Award>          * ;
 p:P166 {| ps:P166  @<Award>          ;
           pq:P585  xsd:dateTime    ? ;
           pq:P1706 @<Researcher>   *
 |} *
}
<Award> { wdt:P17 @<Country> ? }
<Country> {}
```

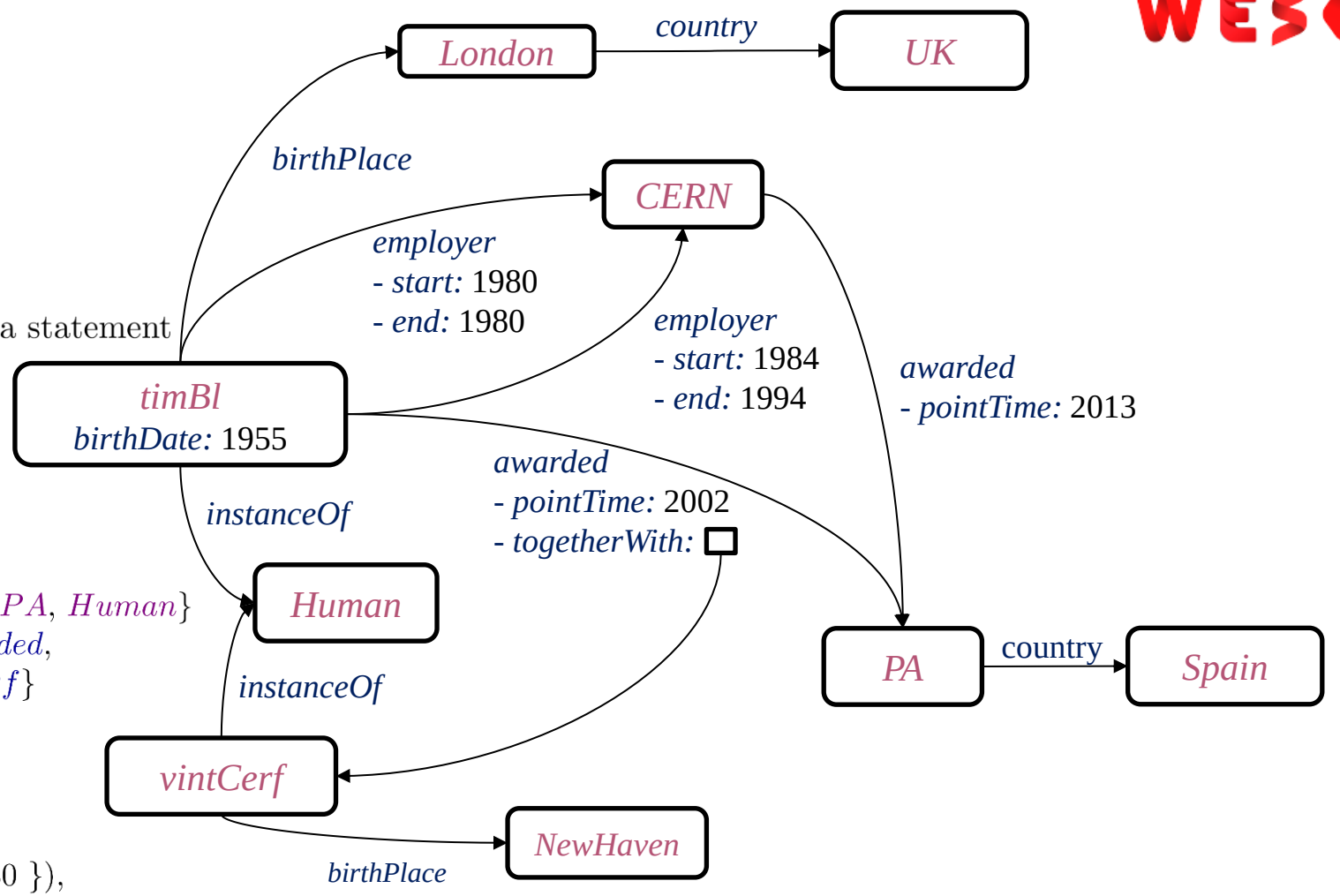
## WShEx

```
PREFIX : <.../entity/>

<Researcher> {
 :P31   [ :Q5 ]     ;
 :P166  @<Award> {| :P585  Time ? ,
                    :P1706 @<Researcher> ?
                 |} *
}
<Award> { :P17 @<Country> }
<Country> {}
```

https://arxiv.org/abs/2208.02697

# Wikibase graphs

Given a mutually disjoint set of items $\mathcal{Q}$,
a set of properties $\mathcal{P}$ and
a set of data values $\mathcal{D}$,
a *Wikibase graph* is a tuple $\langle \mathcal{Q}, \mathcal{P}, \mathcal{D}, \rho \rangle$ such that
$\rho \subseteq \mathcal{E} \times \mathcal{P} \times \mathcal{V} \times FinSet(\mathcal{P} \times \mathcal{V})$ where
$\mathcal{E} = \mathcal{Q} \cup \mathcal{P}$ is the set of entities which can be subjects of a statement
$\mathcal{V} = \mathcal{E} \cup \mathcal{D}$ is the set of possible values of a property.

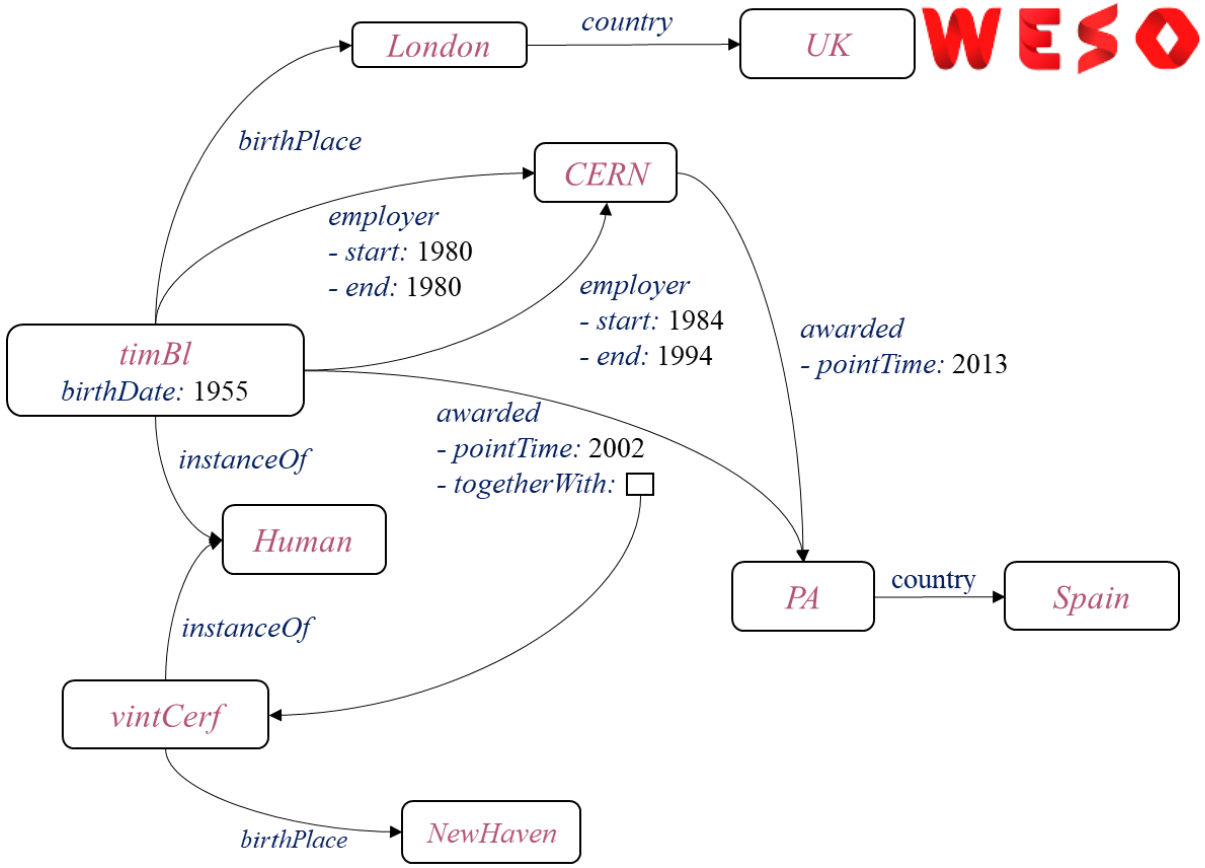



$$
\begin{aligned}
\mathcal{Q} &= \{ \; timBl, vintCerf, London, CERN, UK, Spain, PA, Human\} \\
\mathcal{P} &= \{ \; birthDate, birthPlace, country, employer, awarded, \\
&\quad\; start, end, pointTime, togetherWith, instanceOf\} \\
\mathcal{D} &= \{ \; 1984{,}1994{,}1980{,}1955\} \\
\rho &= \{ \; (timBl, instanceOf, Human, \{\}), \\
&\quad\; (timBl, birthDate, 1955, \{\}), \\
&\quad\; (timBl, birthPlace, London, \{\}), \\
&\quad\; (timBl, employer, CERN, \{\ start{:}1980, end{:}1980\ \}), \\
&\quad\; (timBl, employer, CERN, \{\ start{:}1984, end{:}1994\ \}), \\
&\quad\; (timBl, awarded, PA, \{pointTime{:}\ 2002, togetherWith{:}vintCerf\}), \\
&\quad\; (London, country, UK, \{\}), \\
&\quad\; (vintCerf, instanceOf, Human, \{\}) \\
&\quad\; (vintCerf, birthPlace, NewHaven, \{\}) \\
&\quad\; (CERN, awarded, PA, \{\ pointTime{:}\ 2013\ \}) \\
&\quad\; (PA, country, Spain, \{\ \}) \ \}
\end{aligned}
$$

# WShEx for Wikibase graphs

A WShEx Schema is a tuple $\langle \mathcal{L}, \delta \rangle$ where
$\mathcal{L}$ set of shape labels
$\delta : \mathcal{L} \rightarrow se$

| | | | |
|---|---|---|---|
| $se$ | ::= | cond | Basic boolean condition on nodes (node constraint) |
| | \| | $s$ | Shape |
| | \| | $se_1$ AND $se_2$ | Conjunction |
| | \| | $@l$ | Shape label reference for $l \in \mathcal{L}$ |
| $s$ | ::= | CLOSED $s'$ | Closed shape |
| | \| | $s'$ | Open shape |
| $s'$ | ::= | { te } | Shape definition |
| $te$ | ::= | $te_1; te_2$ | Each of $te_1$ and $te_2$ |
| | \| | $te_1 \mid te_2$ | Some of $te_1$ or $te_2$ |
| | \| | $te*$ | Zero or more $te$ |
| | \| | $\llcorner \xrightarrow{p} @l\ qs$ | Triple constraint with predicate $p$ value conforming to $l$ and qualifier specifier $qs$ |
| | \| | $\epsilon$ | Empty triple expression |
| $qs$ | ::= | $\lfloor ps \rfloor$ | Open property specifier |
| | \| | $\lceil ps \rceil$ | Closed property specifier |
| $ps$ | ::= | ps , ps | *EachOf* property specifiers |
| | \| | $ps \mid ps$ | *OneOf* property specifiers |
| | \| | ps* | zero of more property specifiers |
| | \| | $\epsilon$ | Empty property specifier |
| | \| | p:@$l$ | Property $p$ with value conforming to shape $l$ |




$$
\begin{array}{rcl}
\mathcal{L} & = & \{ \ Person, Place, Country, Organization, Date, \ Award\} \\
\delta(Person) & = & \{ \ \llcorner \xrightarrow{birthDate} @Date; \ \llcorner \xrightarrow{birthPlace} @Place; \\
 & & \llcorner \xrightarrow{employer} @Organization \ \lfloor start : @Date \, , \, end : @Date \rfloor * \\
 & & \llcorner \xrightarrow{awarded} @Award \ \lfloor pointTime : @Date \, , \, togetherWith : @Person \rfloor * \\
 & & \} \\
\delta(Place) & = & \{ \ \llcorner \xrightarrow{country} @Country\} \\
\delta(Country) & = & \{ \ \} \\
\delta(Award) & = & \{ \ \llcorner \xrightarrow{country} @Country\} \\
\delta(Organization) & = & \{ \ \} \\
\delta(\text{Date}) & = & \in xsd : date
\end{array}
$$

# Use cases

- Wikidata subsetting: Describe directly JSON dumps
- Entity schemas linter
  - Entity Schemas can be parsed to WShEx detecting inconsistencies
- Wikibase validation
  - Improve validation quality and messages
- Using WShEx ideas for Querying
  - Concise syntax and ability to handle dumps

Further information:

WShEx Specification:
https://www.weso.es/WShEx/

# Future work

- Complete WShEx specification
  - Semantic specification including other features: references, labels,...
  - Compact syntax grammar
- Converter Entity Schemas (ShEx) ↔ WShEx
- WShEx tooling: validation, editors, etc.

Further information:

WShEx Specification:
https://www.weso.es/WShEx/

# Contents

# Property graphs

- Popular model in industry
  - Neo4j, Amazon Neptune, Oracle, etc
- GQL has been published in 2024
  - Recent publication of ISO/IEC FDIS 39075
  - Developed by ISO/IEC JTC1 SC32 WG3: the “SQL” committee
  - Influenced by Cypher, PGQL, etc.
  - Specification behind a paywall*
    - But some open source tools: https://ldbcouncil.org
    - and public documents

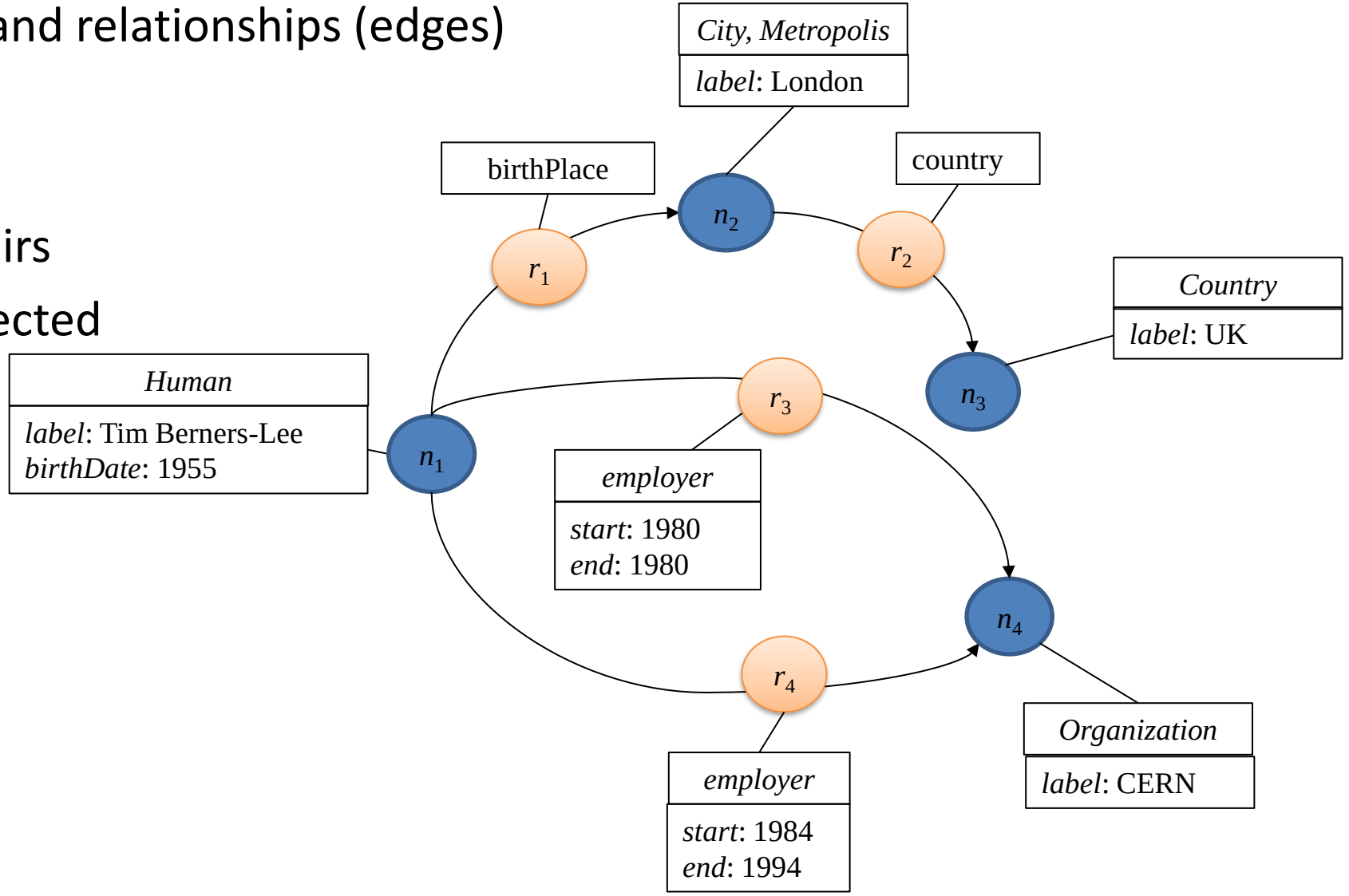
# Property graphs

Graph structure with nodes and relationships (edges)

Nodes and edges can have:

- Labels
- A set of property-value pairs

Edges can be directed/undirected



# Shaping Property graphs with GQL

GQL defines the concept of Graph Types

- It describes the graph in terms of restrictions on labels, properties, nodes, edges and topology
- Graph types constrain the set of nodes that can be contained in a graph

Multiple graphs can refer to the same graph type

Graph types can be created independently:

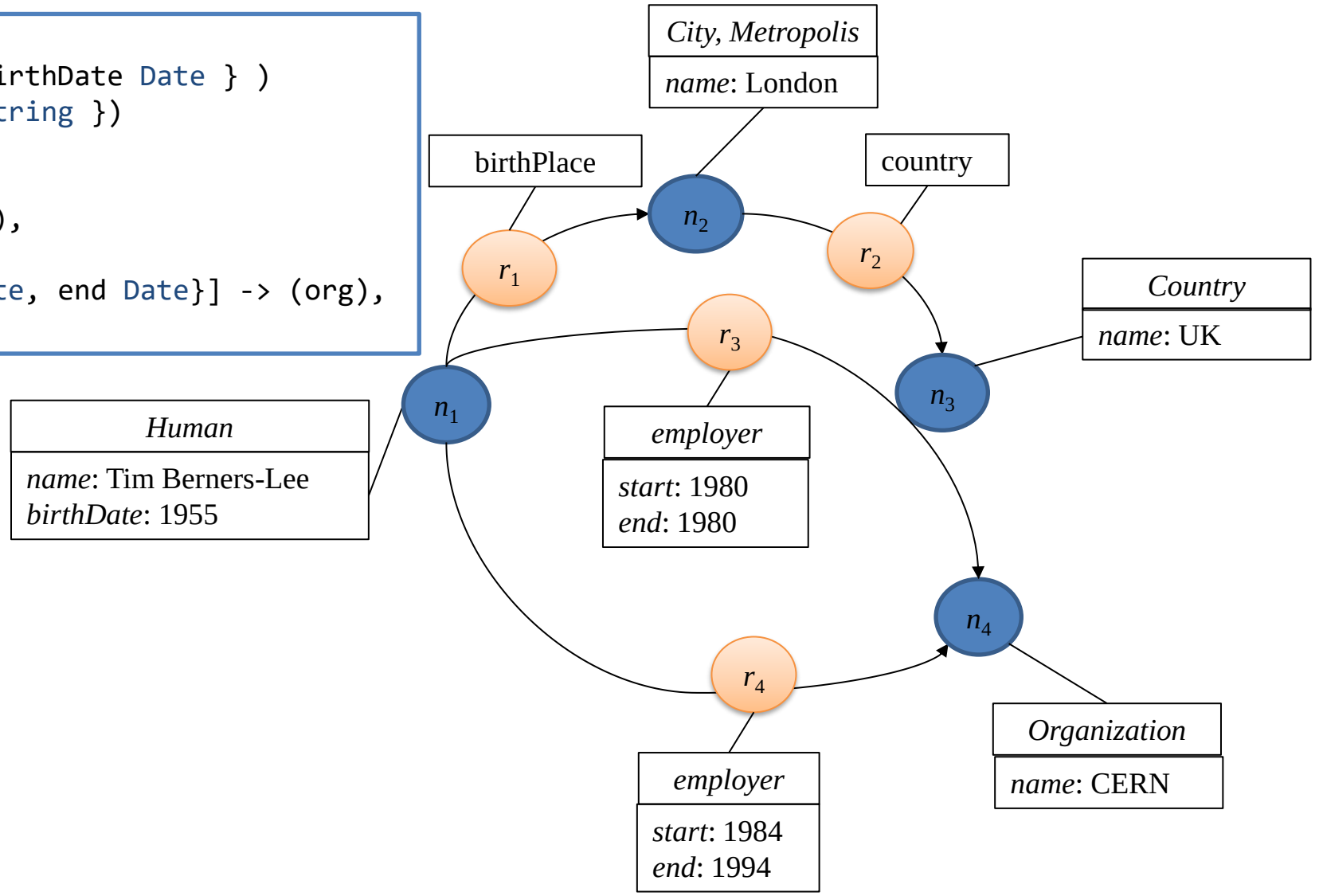
```
CREATE GRAPH TYPE name { graph type spec }
```

Or when creating the graph:

```
CREATE GRAPH …content… TYPED { graph type spec }
```

# Shaping property graphs with GQL: Example

```
CREATE GRAPH TYPE example {
  (researcher :Human => { name String, birthDate Date } )
  (place :City & :Metropolis => { name String })
  (country :Country => { name String })
  (org :Organization => { name String })
  (researcher) -[ :birthPlace ]-> (place),
  (place) -[ :country ]-> (country),
  (researcher) -[:employer => { start Date, end Date}] -> (org),
}
```



# Shaping Property graphs with GQL

Validating graphs with graph types

A graph is of a graph type if:

- Each of the nodes in the graph are of a node type specified in the graph type
- Each of the edges in the graph are of an Edge type in the graph

Inserting or updating nodes and edges in a graph that has a constraining graph type such that the graph would no longer be of that graph type causes an exception condition to be raised: **graph type violation**

# Shaping property graphs with GQL

Graph types in GQL are closed

- No open/partial semantics

No Cardinality constraints about the topology of the graph?

Schema fixed vs schema-less

- It is also possible to have a schema-less graph as:

```
CREATE GRAPH … TYPED ANY
```

- Schema-less graphs are not restricted by a graph type, they may contain anything the property graph model allows.

# Shaping property graphs: PG-Schema

PGSchema has been proposed as a joint effort of several researchers

- Open/Closed Record types
- Edge/Node types
- Labelling types
- Content types
- Constraints with PG-Keys
- No support for cardinality constraints on the graph topology

See also:

- Ora Lassila's presentation about a Common PG-Schema and SHACL
- Our recent work about a Common foundation for PGSchema, SHACL and ShEx

# PShEx

Proposal to extend ShEx to handle property graphs

Similar to WShEx, but adapted for Property graphs

We add a new description level for property-value pairs (in nodes and edges)

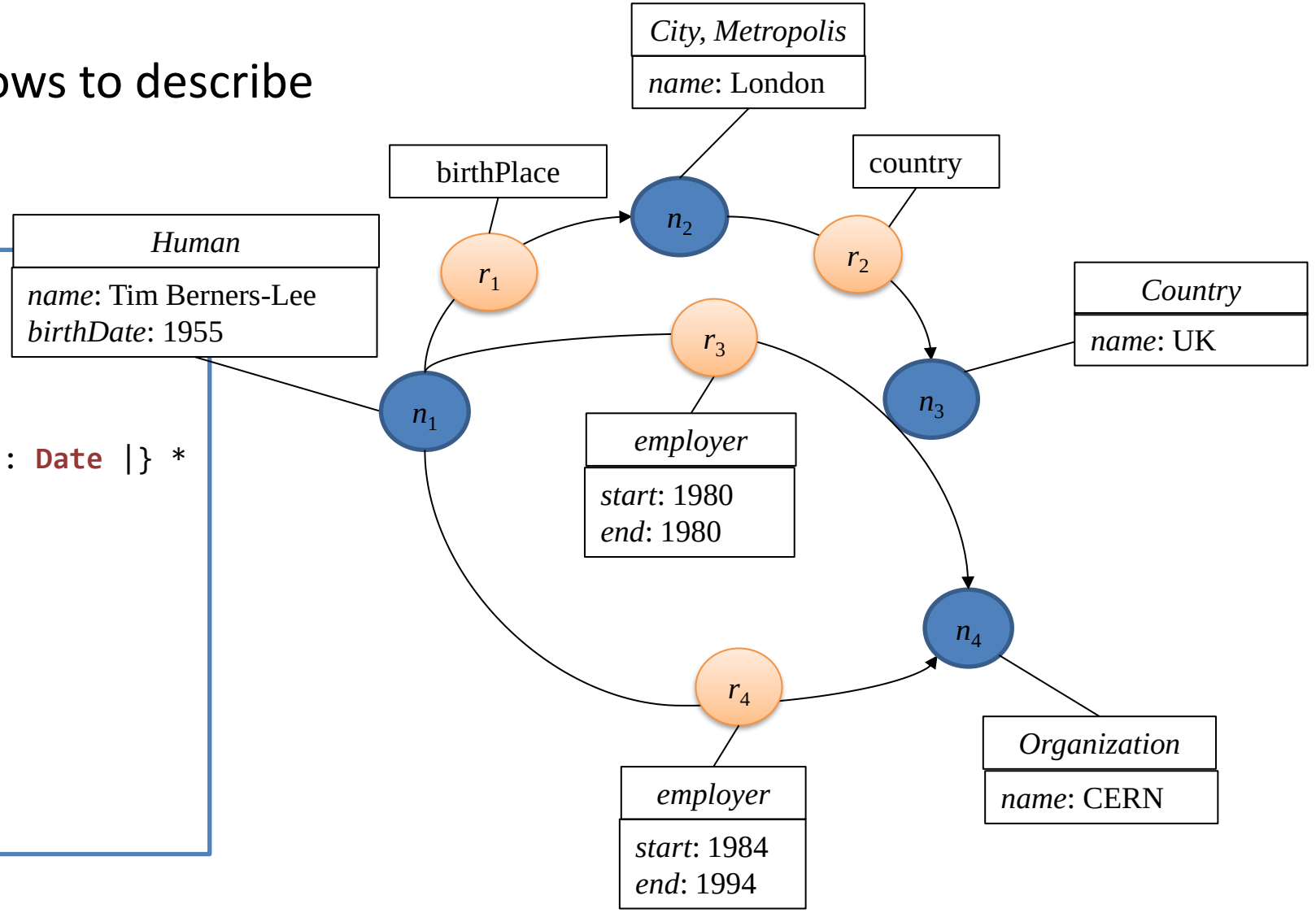
It allows to declare cardinality constraints on the topology of the graph

Extending Shape Expressions for different types of knowledge graphs , Jose Emilio Labra Gayo, In 1st Workshop on *Data Quality meets Machine Learning and Knowledge Graphs*, DQMLKG, part of *Extended Semantic Web Conference 2024*, ESWC24. - 2024, http://labra.weso.es/publication/2024_shex_extensions/

# Shaping property graphs: PShEx

PShEx = ShEx extension that allows to describe property graphs

```
<Researcher> [( Human )] [|
  name:       String     ;
  birthDate:  Date     ?
|] AND {
  birthPlace: @<Place> ? ;
  employer:   @<Org> {| start: Date; end: Date |} *
}
<Place> [( City)]{
  country: @<Country>
}
<Country> {
  name: String
}
<Org> {
  name: String
}
```

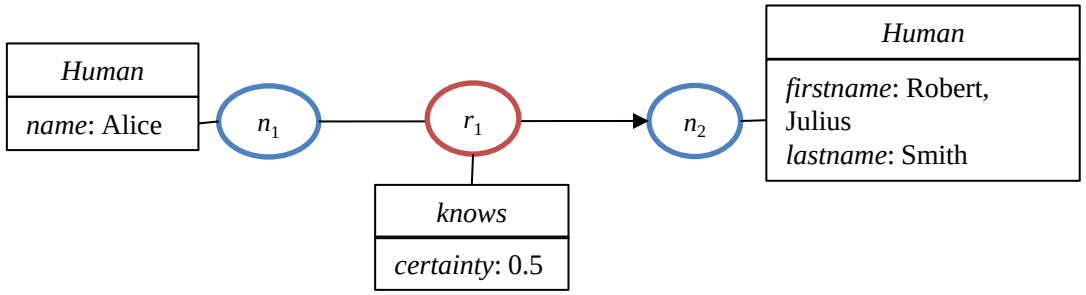


# Shaping property graphs with PShEx

PShEx supports regular expressions to describe the graph content

Another example

```
<Person> [( Human )] AND
  [| name: String |
     firstname: String *,
     lastname: String
  |] AND {
     knows @<Person> {|
       certainty: Float
     |} *
  }
```



# PShEx abstract syntax

## PShEx adds property value specifiers (*pvs*) to nodes and edges

| | | | |
|---|---|---|---|
| $se$ | ::= | $cond_{t_s}$ | Basic boolean condition on set of types $t_s \subseteq T$ |
| | \| | $s$ | Shape |
| | \| | $se_1$ AND $se_2$ | Conjunction |
| | \| | $@l$ | Shape label reference for $l \in L$ |
| | \| | $pvs$ | Property-value specifiers of a node |
| $s$ | ::= | CLOSED $\{te\}$ | Closed shape |
| | \| | $\{te\}$ | Open shape |
| $te$ | ::= | $te_1; te_2$ | Each of $te_1$ and $te_2$ |
| | \| | $te_1 \mid te_2$ | Some of $te_1$ or $te_2$ |
| | \| | $te*$ | Zero or more $te$ |
| | \| | $\_ \xrightarrow{p} @l\ pvs$ | Triple constraint with property type $p$ whose nodes satisfy the shape $l$ and property-values $pvs$ |
| $pvs$ | ::= | $\lfloor ps \rfloor$ | Open property-value specifiers $ps$ |
| | \| | $\lceil ps \rceil$ | Closed property-value specifiers $ps$ |
| $ps$ | ::= | $ps_1\ ,\ ps_2$ | Each of $ps_1$ and $ps_2$ |
| | \| | $ps_1 \mid ps_2$ | OneOf of $ps_1$ or $ps_2$ |
| | \| | $ps*$ | zero of more $ps$ |
| | \| | $p : cond_v$ | Property $p$ with value conforming to $cond_v$ $cond_{v_s}$ is a boolean condition on sets of values $v_s \subseteq V$ |

```
<Person> [( Human )] AND
  [| name: String |
     firstname: String *,
     lastname: String
  |] AND {
     knows @<Person> {|
       certainty: Float
     |} *
  }
```

$$
\begin{aligned}
L &= \{\ Person\} \\
\delta(Person) &= hasType_{Human}\ \text{AND} \\
&\quad \lfloor name : String \mid firstname : String*\ , lastname : String \rfloor\ \text{AND} \\
&\quad \{\ \_ \xrightarrow{knows} @Person\ \lfloor certainty : Float \rfloor^*\ \}
\end{aligned}
$$

# Contents

# RDF1.2 (RDF-Star)

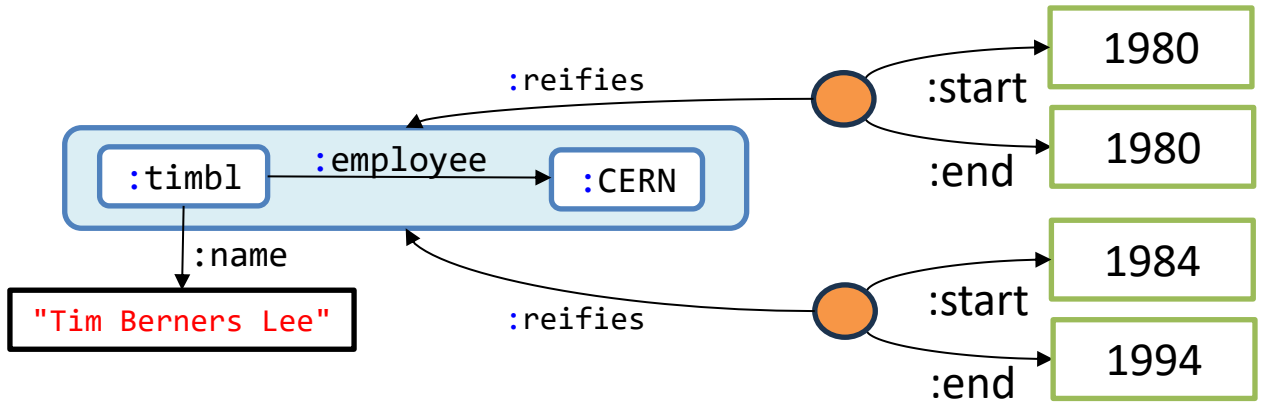
Currently under discussion (https://github.com/w3c/rdf-star-wg/wiki)

Add statements about triples

Reifiers

```
:timbl :name  "Tim Berners Lee“ ;
       :employer :CERN .
_:r1 rdf:reifies <<( :timbl :employer :CERN )>> .
_:r1 :start 1980 ;
     :end   1980 .
```

```
_:r2 rdf:reifies <<( :timbl :employer :CERN )>> .
_:r2 :start 1984 ;
     :end   1994 .
```




Alternative syntax

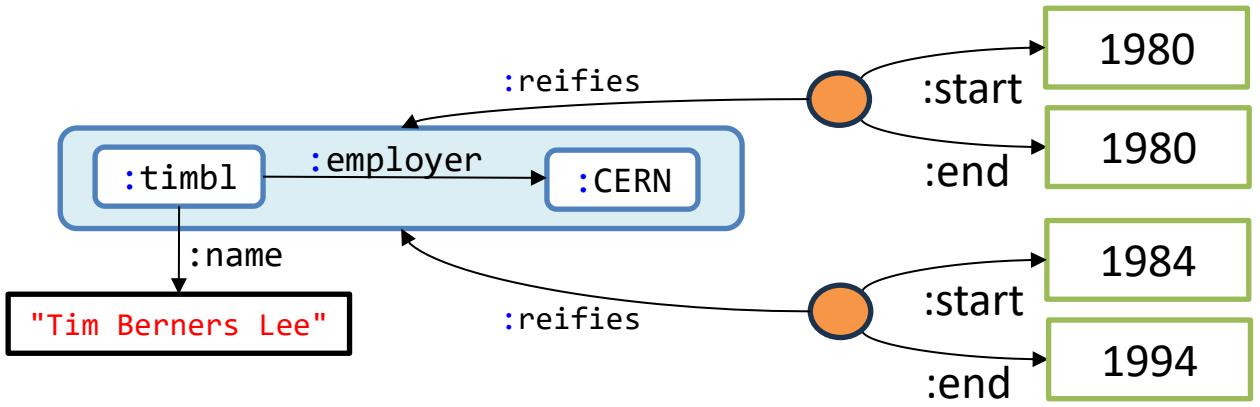
```
:timbl :name  "Tim Berners Lee";
       :employer :CERN {|
          :start 1980; :end 1980
        |} {|
          :start 1984; :end 1994
        |} .
```

# ShEx-Star

## Example

```
<Person> {
    ( :name   xsd:string
    | :firstName xsd:string + ;
      :lastname xsd:string
    ) ;
   << :employee @<Org> >> {|
        :start xsd:date,
        :start xsd:date
   |} *
}
<Org> {}
```

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

:timbl :name  "Tim Berners Lee";
       :employer :CERN {|
          :start 1980;
          :end   1980
        |} {|
          :start 1984;
          :end   1994
        |} .
```

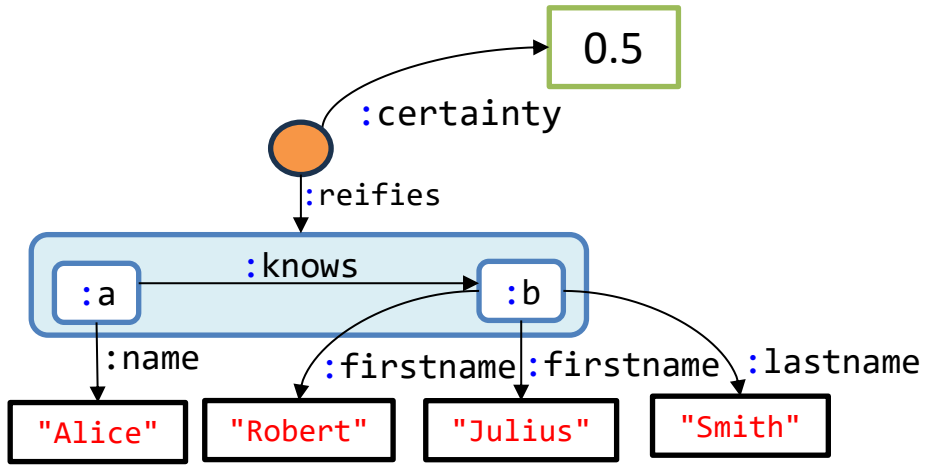


# ShEx-Star

## Example

```
<Person> {
    ( :name   xsd:string
    | :firstName xsd:string + ;
      :lastname xsd:string
    ) ;
   << :knwows @<Person> >> {| :certainty xsd:float |} *
}
```

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

:a :name  "Alice" .
<< :a :knows :b >> :certainty 0.5 .
:b :firstname "Robert", "Julius" ;
    :lastname  "Smith" .
```



# ShEx-Star abstract syntax

## ShEx-Star adds two new rules for triple expressions te

$$se \quad ::= \quad \ldots \text{same as in ShEx}$$

| | | | |
|---|---|---|---|
| $te$ | $::=$ | $te_1; te_2$ | Each of $te_1$ and $te_2$ |
| | $\mid$ | $te_1 \mid te_2$ | Either $te_1$ or $te_2$ |
| | $\mid$ | $te*$ | Zero or more $te$ |
| | $\mid$ | $\llcorner \xrightarrow{p} se$ | Outgoing Triple with predicate $p$ and object conforming to $se$ |
| | $\mid$ | $se \xrightarrow{p} \llcorner$ | Incoming triple with predicate $p$ and subject conforming to $se$ |
| | $\mid$ | $\epsilon$ | Empty triple expression |
| | $\mid$ | $\ll \llcorner \xrightarrow{p} se \gg \{\mid te \mid\}$ | Outgoing Triple term constraint with predicate $p$ |
| | $\mid$ | $\ll se \xrightarrow{p} \llcorner \gg \{\mid te \mid\}$ | Incoming triple term constraint with predicate $p$ |

$$\delta(Person) = \{ \; (\llcorner \xrightarrow{name} String \mid \llcorner \xrightarrow{fistname} String*; \llcorner \xrightarrow{lastname} String); \; \ll \llcorner \xrightarrow{knows} @Person \gg \{\mid \llcorner \xrightarrow{certainty} Float \mid\}* \; \}$$

```
<Person> {
    ( :name   xsd:string
    | :firstName xsd:string + ;
      :lastname xsd:string
    ) ;
   << :knwows @<Person> >> {| :certainty xsd:float |} *
}
```

# Semantics of these extensions

We define the semantics using 2 conformance relationships and several inference rules

$$G, n, \tau \vDash se = \text{node } n \text{ in graph } G \text{ conforms to } se \text{ with assignment } \tau$$

$$G, ts, \tau \Vdash te = \text{neighborhood } ts \text{ of graph } G \text{ conform to } te \text{ with assignment } \tau$$

More details in the paper

Extending Shape Expressions for different types of knowledge graphs , Jose Emilio Labra Gayo, In 1st Workshop on *Data Quality meets Machine Learning and Knowledge Graphs*, DQMLKG, part of *Extended Semantic Web Conference 2024*, ESWC24. - 2024
http://labra.weso.es/publication/2024_shex_extensions/

# ShEx semantics

Shape expressions *se*

$$Cond\frac{cond(n) = true}{G, n, \tau \vDash cond} \qquad AND\frac{G, n, \tau \vDash se_1 \quad G, n, \tau \vDash se_2}{G, n, \tau \vDash se_1 \texttt{ AND } se_2}$$

$$ShapeRef\frac{\delta(l) = se \quad G, n, \tau \vDash se}{G, n, \tau \vDash @l} \qquad ClosedShape\frac{neighs(n, G) = ts \quad G, ts, \tau \Vdash te}{G, n, \tau \vDash \texttt{CLOSED } \{te\}}$$

Triple expressions *te*

$$OpenShape\frac{ts = \{\langle x, p, y\rangle \in neighs(n, G) \mid p \in preds(te)\} \quad G, ts, \tau \Vdash te}{G, n, \tau \vDash \{te\}}$$

$$EachOf\frac{(ts_1, ts_2) \in part(ts) \quad G, ts_1, \tau \Vdash te_1 \quad G, ts_2, \tau \Vdash te_2}{G, ts, \tau \Vdash te_1; te_2}$$

$$OneOf_1\frac{G, ts, \tau \Vdash te_1}{G, ts, \tau \Vdash te_1 \mid te_2} \qquad OneOf_2\frac{G, ts, \tau \Vdash te_2}{G, ts, \tau \Vdash te_1 \mid te_2}$$

$$TC_1\frac{ts = \{\langle x, p, y\rangle\} \quad G, y, \tau \vDash @l}{G, ts, \tau \Vdash \_ \xrightarrow{p} @l} \qquad TC_2\frac{ts = \{\langle y, p, x\rangle\} \quad G, y, \tau \vDash @l}{G, ts, \tau \Vdash @l \xrightarrow{p} \_}$$

$$Star_2\frac{(ts_1, ts_2) \in part(ts) \quad G, ts_1, \tau \Vdash te \quad G, ts_2, \tau \Vdash te*}{G, ts, \tau \Vdash te*} \qquad Star_1\frac{}{G, \emptyset, \tau \Vdash te*}$$

# ShEx-Star semantics

Same rules as for ShEx plus:

$$TTC_1 \frac{ts = \{\langle \ll t \gg, p, y \rangle\} \quad G, y, \tau \vDash se \quad neighs(\ll t \gg, G) = ts' \quad G, ts', \tau \Vdash te}{G, ts, \tau \Vdash \ll \_ \xrightarrow{p} se \gg \{|te|\}}$$

$$TTC_2 \frac{ts = \{\langle x, p, \ll t \gg \rangle\} \quad G, x, \tau \vDash se \quad neighs(\ll t \gg, G) = ts' \quad G, ts', \tau \Vdash te}{G, ts, \tau \Vdash \ll se \xrightarrow{p} \_ \gg \{|te|\}}$$

$$\begin{aligned} \delta(Person) \quad = \quad \{ \quad & (\_ \xrightarrow{name} String \mid \_ \xrightarrow{fistname} String^*; \_ \xrightarrow{lastname} String); \\ & \ll \_ \xrightarrow{knows} @Person \gg \{|\_ \xrightarrow{certainty} Float|\}^* \\ \} & \end{aligned}$$

# PShEx semantics

Semantics of shape expressions *se* (similar to ShEx)

$$Cond_{ts}\ \frac{\lambda_n(n) = vs \qquad cond_{ts}(vs) = true}{G, n, \tau \vDash cond_{ts}} \qquad AND\ \frac{G, n, \tau \vDash se_1 \qquad G, n, \tau \vDash se_2}{G, n, \tau \vDash se_1\ \texttt{AND}\ se_2}$$

$$ClosedShape\ \frac{neighs(n, G) = ts \qquad G, ts, \tau \Vdash s'}{G, n, \tau \vDash \texttt{CLOSED}\ \{te\}}$$

$$OpenShape\ \frac{ts = \{\langle x, p, y\rangle \in neighs(n, G) \mid p \in preds(te)\} \qquad G, ts, \tau \Vdash te}{G, n, \tau \vDash \{te\}}$$

# PShEx semantics

Semantics of property value specifiers *ps*

$$OpenPVs\frac{s' = \{(p,v) \in s | p \in props(ps)\} \quad G, s', \tau \vdash ps}{G, s, \tau \vdash \lfloor ps \rfloor} \qquad ClosePVs\frac{G, s, \tau \vdash ps}{G, s, \tau \vdash \lceil ps \rceil}$$

$$EachOfPs\frac{G, s, \tau \vdash ps_1 \quad G, s, \tau \vdash ps_2}{G, s, \tau \vdash ps_1 \ , \ ps_2}$$

$$OneOfPs_1\frac{G, s, \tau \vdash ps_1}{G, s, \tau \vdash ps_1 \mid ps_2} \qquad OneOfPs_2\frac{G, s, \tau \vdash ps_2}{G, s, \tau \vdash ps_1 \mid ps_2}$$

$$StarPs_1\frac{}{G, \emptyset, \tau \vdash ps*} \qquad StarPs_2\frac{(s_1, s_2) \in part(s) \quad G, s_1, \tau \vdash ps \quad G, s_2, \tau \vdash ps*}{G, s, \tau \vdash ps*}$$

$$PropertyValue\frac{s = \{(p,w)\} \quad conv_v(w) = \texttt{true}}{G, s, \tau \vdash p : cond_v}$$

# PShEx semantics

Semantics of triple expressions *te* (similar to ShEx)

$$EachOf\ \frac{(ts_1, ts_2) \in part(ts) \quad G, ts_1, \tau \Vdash te_1 \quad G, ts_2, \tau \Vdash te_2}{G, ts, \tau \Vdash te_1; te_2}$$

$$OneOf_1\ \frac{G, ts, \tau \Vdash te_1}{G, ts, \tau \Vdash te_1 \mid te_2} \qquad OneOf_2\ \frac{G, ts, \tau \Vdash te_2}{G, ts, \tau \Vdash te_1 \mid te_2}$$

$$TripleConstraint\ \frac{ts = \{\langle x, p, y, s\rangle\} \quad G, y, \tau \vDash @l \quad G, s, \tau \vdash qs}{G, ts, \tau \Vdash \_ \xrightarrow{p} @l\ qs}$$

$$Star_1\ \frac{}{G, \emptyset, \tau \Vdash te*}$$

$$Star_2\ \frac{(ts_1, ts_2) \in part(ts) \quad G, ts_1, \tau \Vdash te \quad G, ts_2, \tau \Vdash te*}{G, ts, \tau \Vdash te*}$$

# Convergence on graph schema languages

More theoretical work is currently done:

***Common Foundations for SHACL, ShEx, and PG-Schema***, Shqiponja Ahmetaj, Iovka Boneva, Jan Hidders, Katja Hose, Jose Emilio Labra Gayo, Wim Martens, Fabio Mogavero, Filip Murlak, Cem Okulmus, Axel Polleres, Ognjen Savković, Mantas Šimkus, Dominik Tomaszuk, International World Wide Web Conference, Sidney, Australia, 2025 – 2025

https://labra.weso.es/publication/2025_common_foundations_shacl_shex_pgschema/

# SHACL 1.2

Data shapes working group launched in 2024

Still work in progress

See: https://github.com/w3c/data-shapes/issues/300