トリプルストア、RDF、SPARQL：今勉強しないと遅れますよ！

2015/04/21, Qcon Tokyo 2015
トルヴェ　アントワン (九州大学)
trouve@soc.ait.kyushu-u.ac.jp

Today's Technologies

Data Model	**RDF** (Ressource-Description Framework)
Database Technology	**Triple (Graph) Store**
Database Query Language	**SPARQL** (SPARQL Protocol and RDF Language)
Model /Schema Description Language	**RDFS** (RDF Schema)**, OWL** (Web Ontology Language)
Development Paradigm	**Model-driven** (development Ad-hoc, Fast Prototyping, Agile)

Disclaimer…

I don’t own any RDF company

I am not making this prevention for you to buy some of my product

I just want to talk about some technology I liked recently

The RDF Data Model

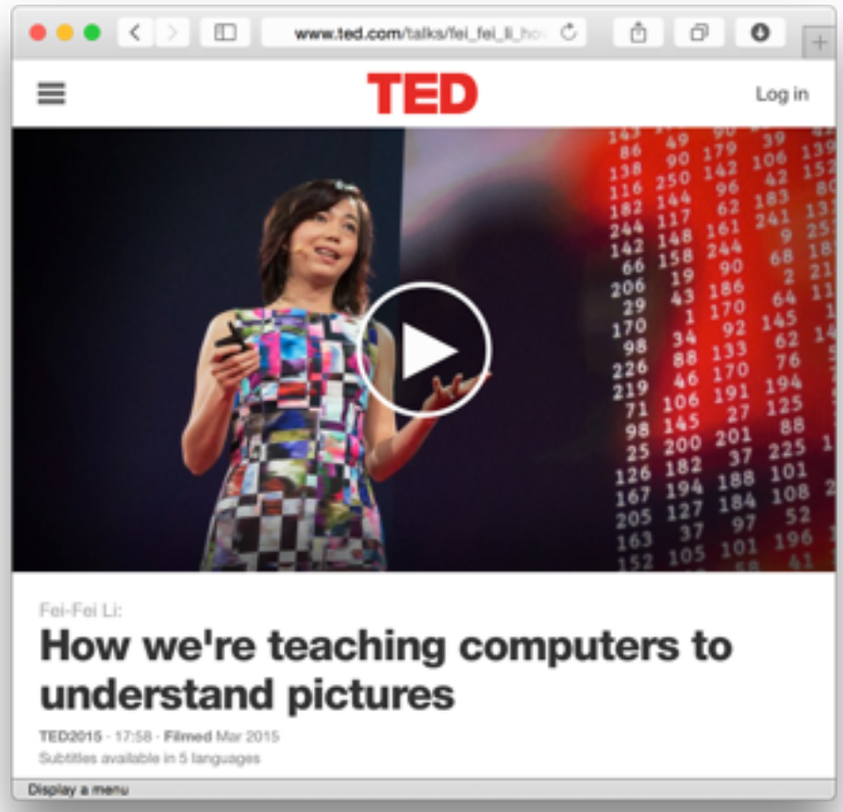
Inception of RDF: Context

- The www is a mine of information
- This is however mostly **unstructured information**
- Such www is useful to us human, because our brain is an outstanding information processor
- It is however **not amenable to machines**

How to Make the Web Amenable to Machines?

Solution 1

Make machines as smart as humans



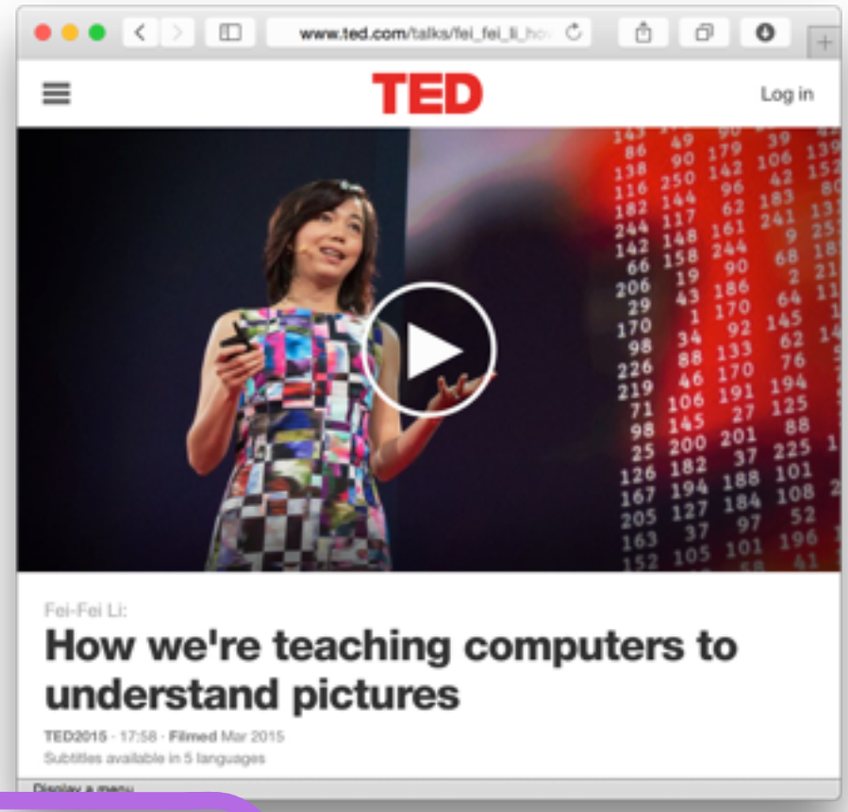
Solution 2

Structure the information on the www for dumb machines to understand it

How to Make the Web Amenable to Machines?

Solution 1

Make machines as smart as humans



Solution 2

Structure the information on the www for dumb machines to understand it

Which Data Model for the Semantic Web ?

Characteristics of the www × **Available Models**

The "AAA" principle: Anyone says Anything about Any topic
→ It is vain to force people to use the same data model
Open world: there is someone else that is saying something about the same topic
→ It is important to merge data from different sources
Non unique naming
→ The data will be heterogenous, even when about the same resources
Dynamic, always in beta

	Model 1 Relational	Model 2 NoSQL
Schema Rigidity	Fixed	Dynamic
Ease to merge	Complex	Transparent
Homo-geneity	Complete	Partial

Key-Value Metadata for the Information on the www

Resource identifier

http://city.fukuoka.lg.jp

is about	Fukuoka city
is a	Web page
last seen	2015-2-1

Keys

Values

Example of the webpage of Fukuoka city

Resource Description Framework

リソース（＝物）　　記述　　枠組み

英語について
1:singleton / 2: couple / 3: triple

Data is expressed as triples

Subject

The resource identifier

Predicate

The key, or property name

Object

The value

Example: from previous slide, as three triple

http://city.fukuoka.lg.jp	**is about**	Fukuoka city
http://city.fukuoka.lg.jp	**is a**	Web page
http://city.fukuoka.lg.jp	**last seen**	2015-2-1

About the W3C

- Founded in 1994 by Tim Berners-Lee et al.
- Regulates the technologies used in the www
  - HTML, XML, Javascript, CSS, **RDF**
- RDF is a W3C standard
  - RDF 1.1 has been published in 2014/2/25
  - It consists of several sub-standards, part of which we will see today

・Tim Berners-Lee, head of the W3C.

・He developed the early version of the www in 1989 (while working at CERN, France)

A Real RDF Example

Subject	Predicate	Object	Language / Type
http://city.fukuoka.lg.jp	http://www.w3.org/2000/01/rdf-schema#type	http://schema.org/WebSite	
http://city.fukuoka.lg.jp	http://schema.org/about	http://dbpedia.org/resource/Fukuoka	
http://city.fukuoka.lg.jp	http://schema.org/lastReviewed	“2015-2-1”	http://www.w3.org/2001/XMLSchema#date
http://city.fukuoka.lg.jp	http://www.w3.org/2000/01/rdf-schema#label	“Fukuoka city official homepage”	en
http://city.fukuoka.lg.jp	http://www.w3.org/2000/01/rdf-schema#label	“福岡市公式ホームページ"	ja

These addresses are **IRI**: Internationalized Resource Identifier (superset of URI)

A Real RDF Example

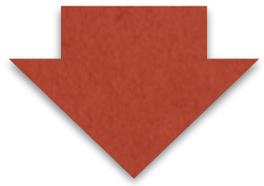
Subject	Predicate	Object	Language / Type
http://city.fukuoka.lg.jp	http://www.w3.org/2000/01/rdf-schema#type	http://schema.org/WebSite	
http://city.fukuoka.lg.jp	http://schema.org/about	http://dbpedia.org/resource/Fukuoka	
http://city.fukuoka.lg.jp	/schema.org/stReviewed	“2015-2-1”	http://www.w3.org/2001/XMLSchema#date
http://city.fukuoka.lg.jp	http://www.w3.org/2000/df-schema#label	“Fukuoka city official homepage”	en
http://city.fukuoka.lg.jp	://www.w3.org/2000/df-schema#label	“福岡市公式ホームページ"	ja

Qname and CURIE: make IRI more readable

http://www.w3.org/2000/01/rdf-schema#type

http://www.w3.org/2000/01/rdf-schema#label

Common prefix



Prefix

rdfs:type

rdfs:label

Local part

- **CURIE**: can contain slashes (used in RDFa - see slides later)
- **Qname**: no slash (used in RDF/XML)

A Real RDF Example with CURIE

Subject	Predicate	Object	Language / Type
http://city.fukuoka.lg.jp	rdfs:type	schema:WebSite	
http://city.fukuoka.lg.jp	schema:about	db:Fukuoka	
http://city.fukuoka.lg.jp	schema:lastReviewed	“2015-2-1”	xsd:date
http://city.fukuoka.lg.jp	rdfs:label	“Fukuoka city official homepage”	en
http://city.fukuoka.lg.jp	rdfs:label	“福岡市公式ホームページ"	ja

I use well-used prefix here. In the real world one should define them before use.

More on that later with turtle and SPARQL

A Real RDF Example with CURIE

Subject	Predicate	Object	Language / Type
http://city.fukuoka.lg.jp	rdfs:type	schema:WebSite	
http://city.fukuoka.lg.jp	schema:about	db:Fukuoka	
http://city.fukuoka.lg.jp	schema:lastReviewed	“2015-2-1”	xsd:date
http://city.fukuoka.lg.jp	rdfs:label	“Fukuoka city official homepage”	en
http://city.fukuoka.lg.jp	rdfs:label	“福岡市公式ホームページ”	ja

I use well-used prefix here. In the real world one should define them before use.

More on that later with turtle and SPARQL

Resource's IRI

Subject	Predicate	Object	Language / Type
http://city.fukuoka.lg.jp	rdfs:type	schema:WebSite	
http://city.fukuoka.lg.jp	schema:about	db:Fukuoka	
http://city.fukuoka.lg.jp	schema:lastReviewed	“2[illegible]2-1”	xsd:date
http://city.fukuoka.lg.jp	[illegible]	[illegible]	[illegible]
http://city.fukuoka.lg.jp	rdfs:label	ジ"	ja

I use as IRI the URL of the website

・This IRI is **not a website**, it designate the **city of Fukuoka** itself.
・Everyone is free to decide its own set of unique IRI to designate resources (we use one from the dbpedia project here)
・It is common to use URL for IRI, the domain being used as namespace

Vocabulary's IRI

Subject	Predicate	Object	Language / Type
http://city.fukuoka.lg.jp		schema:WebSite	
http://city.fukuoka.lg.jp		db:Fukuoka	
http://city.fukuoka.lg.jp	schema:lastReviewed	“2015-2-1”	xsd:date
http://city.fukuoka.lg.jp	rdfs:label	“Fukuoka city official home	
http://city.fukuoka.lg.jp	rdfs:label	“福岡市公式 ジ"	ja

One shall use IRI for predicates.

This IRI designate a type of RDF resources (from schema.org)

This IRI designate a type of litteral (from W3C XML standard)

- **IRI** are also used to designate well-defined **vocabulary** used to describe the meaning of the data (mostly **predicate**, **types**)
- It is possible to define one's own vocabulary, but it is often a good idea to **re-use the well established ones**

I use vocabulary from “schema.org” and W3C standards (XML, RDFS)

Duplicated Predicates

Subject	Predicate	Object	Language / Type
http://city.fukuoka.lg.jp	rdfs:type	schema:WebSite	
http://city.fukuoka.lg.jp	schema:about	db:Fukuoka	
http://city.fukuoka.lg.jp	schema:lastReviewed	“2015-2-1”	xsd:date
http://city.fukuoka.lg.jp	rdfs:label	“Fukuoka city official homepage”	en
http://city.fukuoka.lg.jp	rdfs:label	“福岡市公式ホームページ"	ja

- It is possible to specify several times the same predicate
- This is a way to model unsorted data collections

Specify the Language and Datatype of Literals

Subject	Predicate	Object	Language / Type
http://city.fukuoka.lg.jp	rdfs:type	schema:WebSite	
http://city.fukuoka.lg.jp	schema:about	db:Fukuoka	
http://	ec	“2015-2-1”	xsd:date
http://		“Fukuoka city official homepage”	en
http://city.fukuoka.lg.jp	rdfs:label	“福岡市公式ホームページ"	ja

Example of type tag

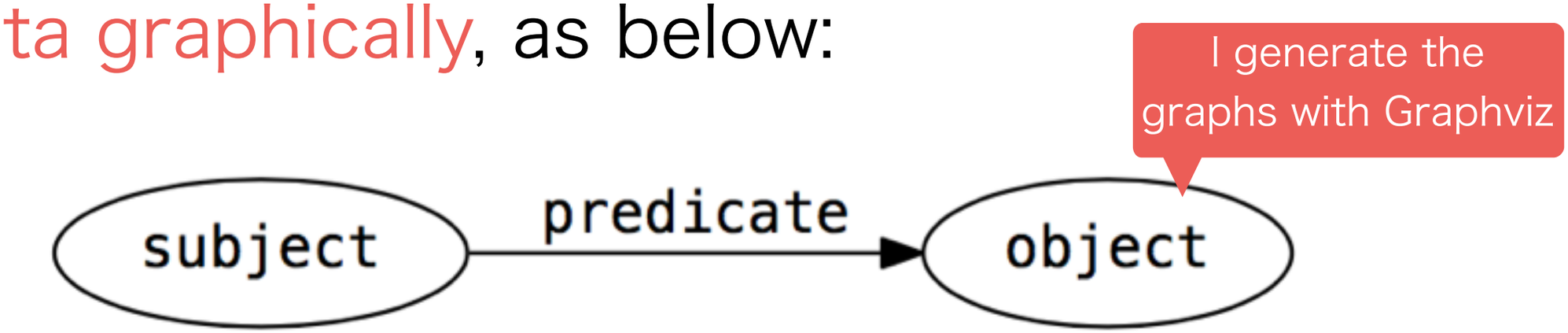
- Literals are non-IRI components
- They may be string, numbers, or any other type
- It is possible to specify language to strings ISO 639

Examples of language tags

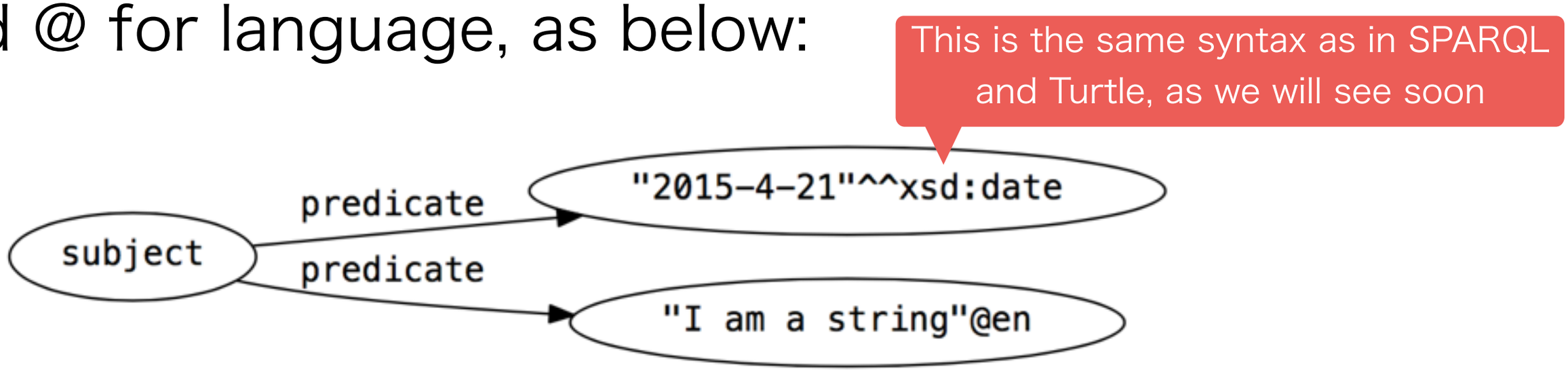
- The XML types are part of the RDF standard:
  - xsd:integer, xsd:decimal, xsd:float, xsd:double, xsd:string, xsd:boolean, xsd:dateTime, xsd:date, xsd:time

RDF Graph Representation

It is possible (and common usage) to represent RDF data graphically, as below:

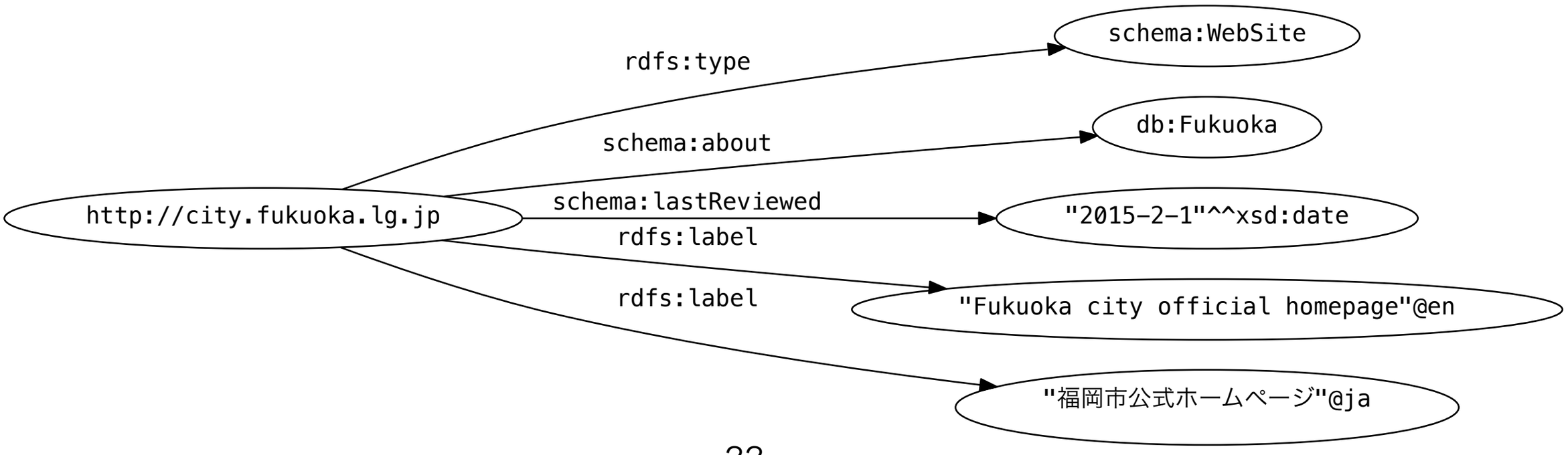


Literals' tags are represented with "^^" for datatypes, and @ for language, as below:



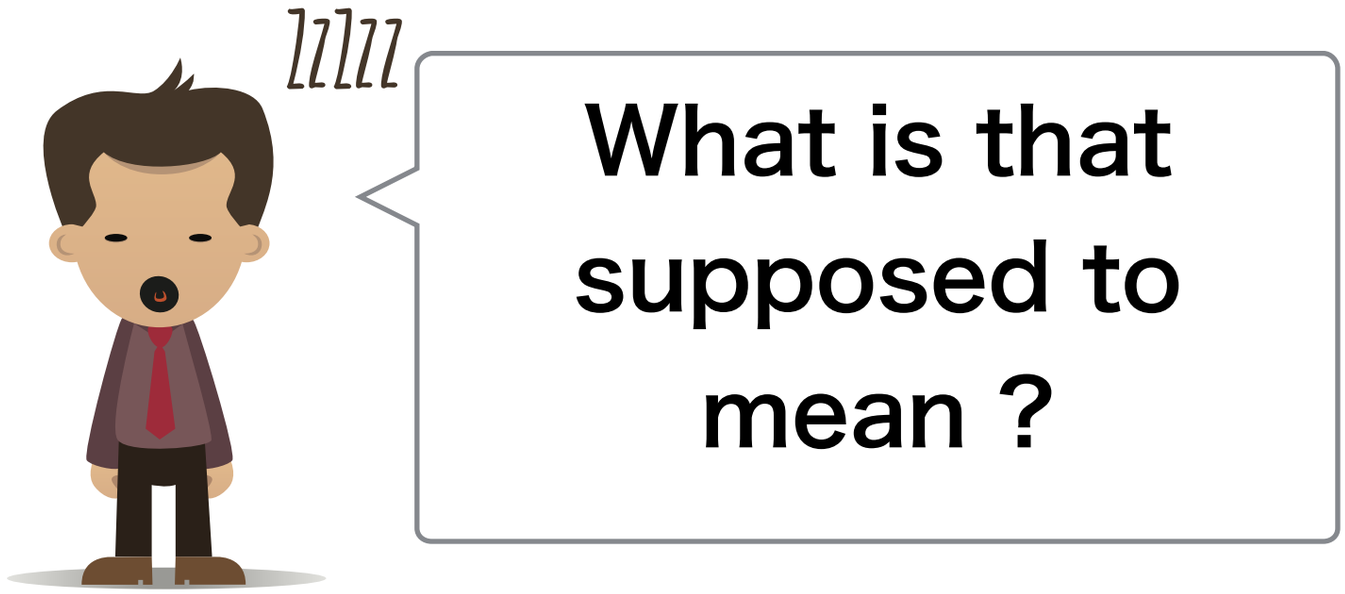
An Example of Graph

Subject	Predicate	Object	Language / Type
http://city.fukuoka.lg.jp	rdfs:type	schema:WebSite	
http://city.fukuoka.lg.jp	schema:about	db:Fukuoka	
http://city.fukuoka.lg.jp	schema:lastReviewed	“2015-2-1”	xsd:date
http://city.fukuoka.lg.jp	rdfs:label	“Fukuoka city official homepage”	en
http://city.fukuoka.lg.jp	rdfs:label	“福岡市公式ホームページ"	ja



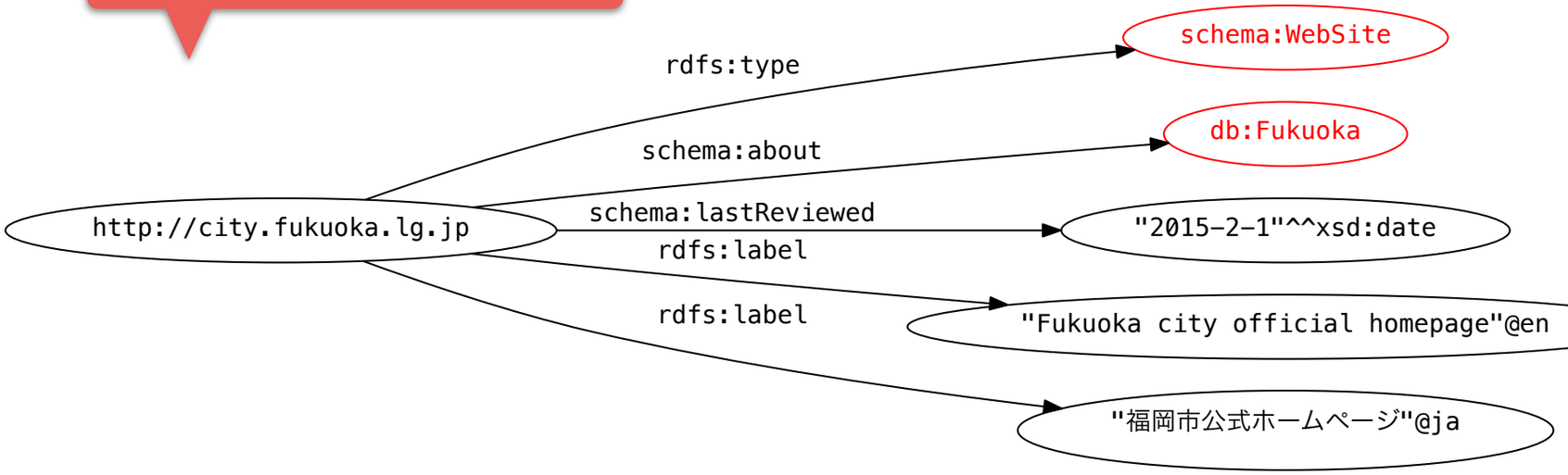
Linked RDF

- By re-using the same IRIs, it possible to express the relationship between resources

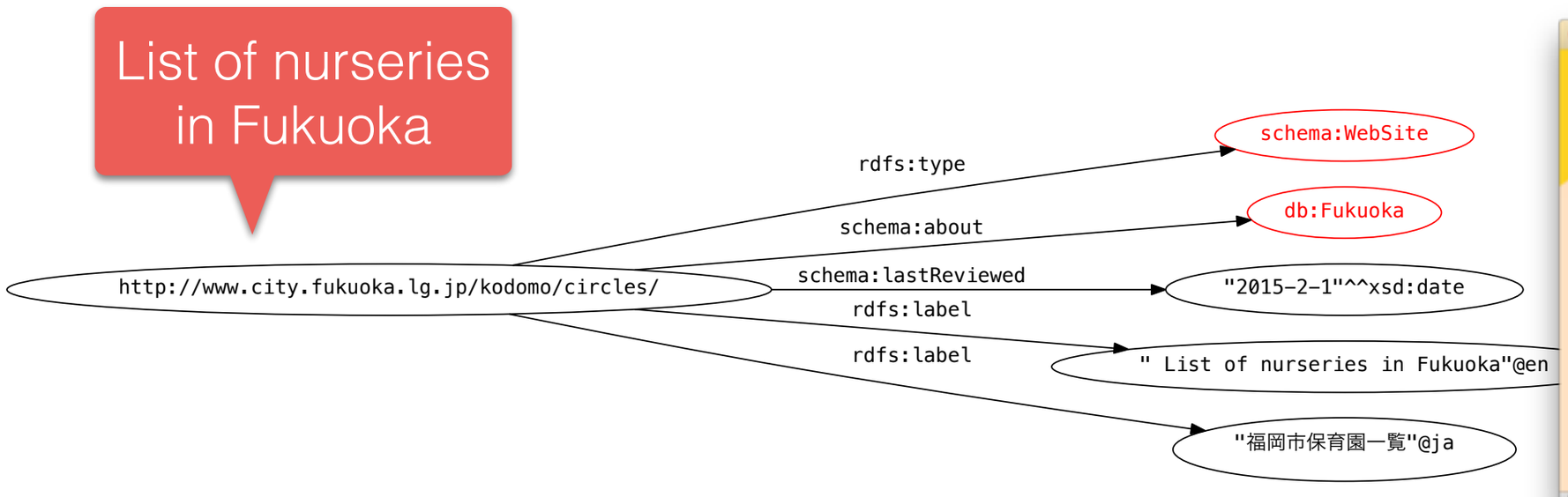


Linked RDF on an Example (1)

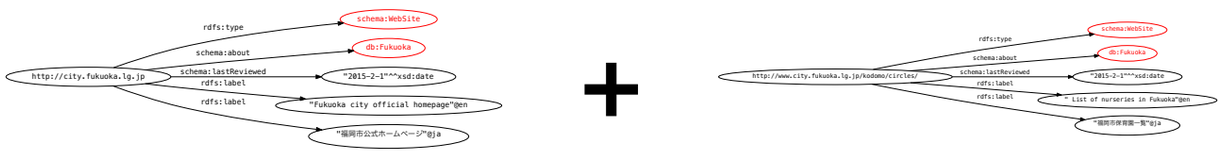
Fukuoka City Homepage

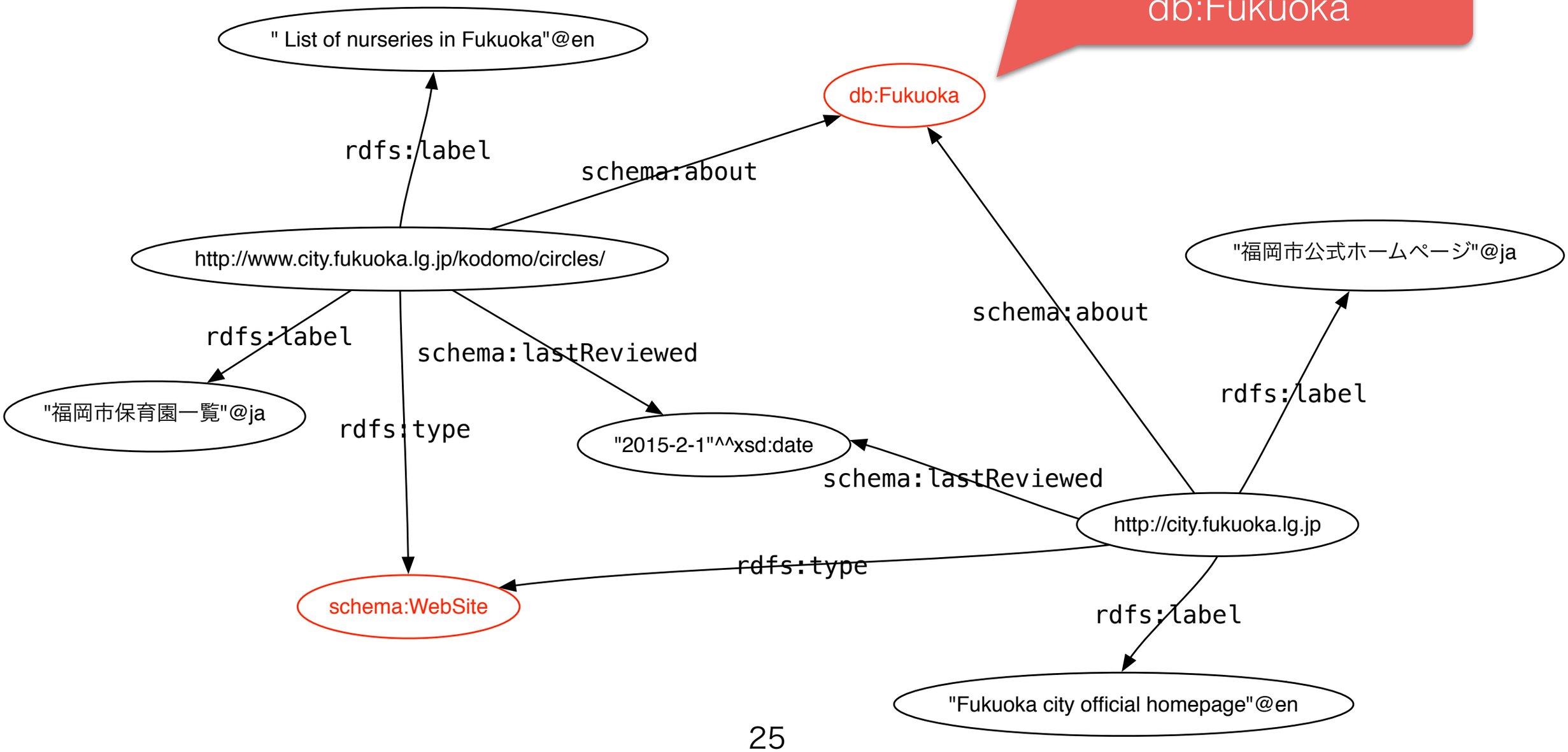


List of nurseries in Fukuoka



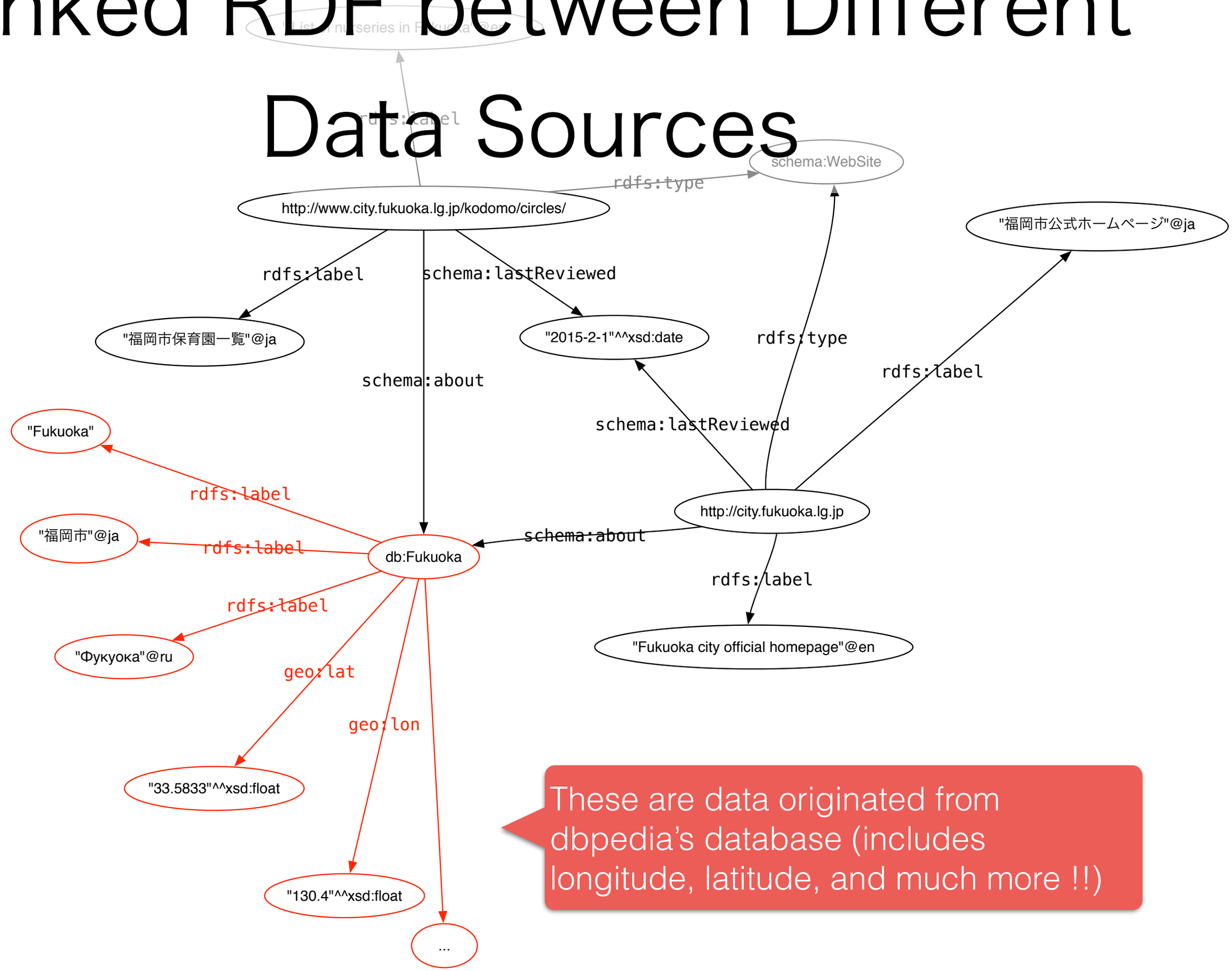
Linked RDF on an Example (2)





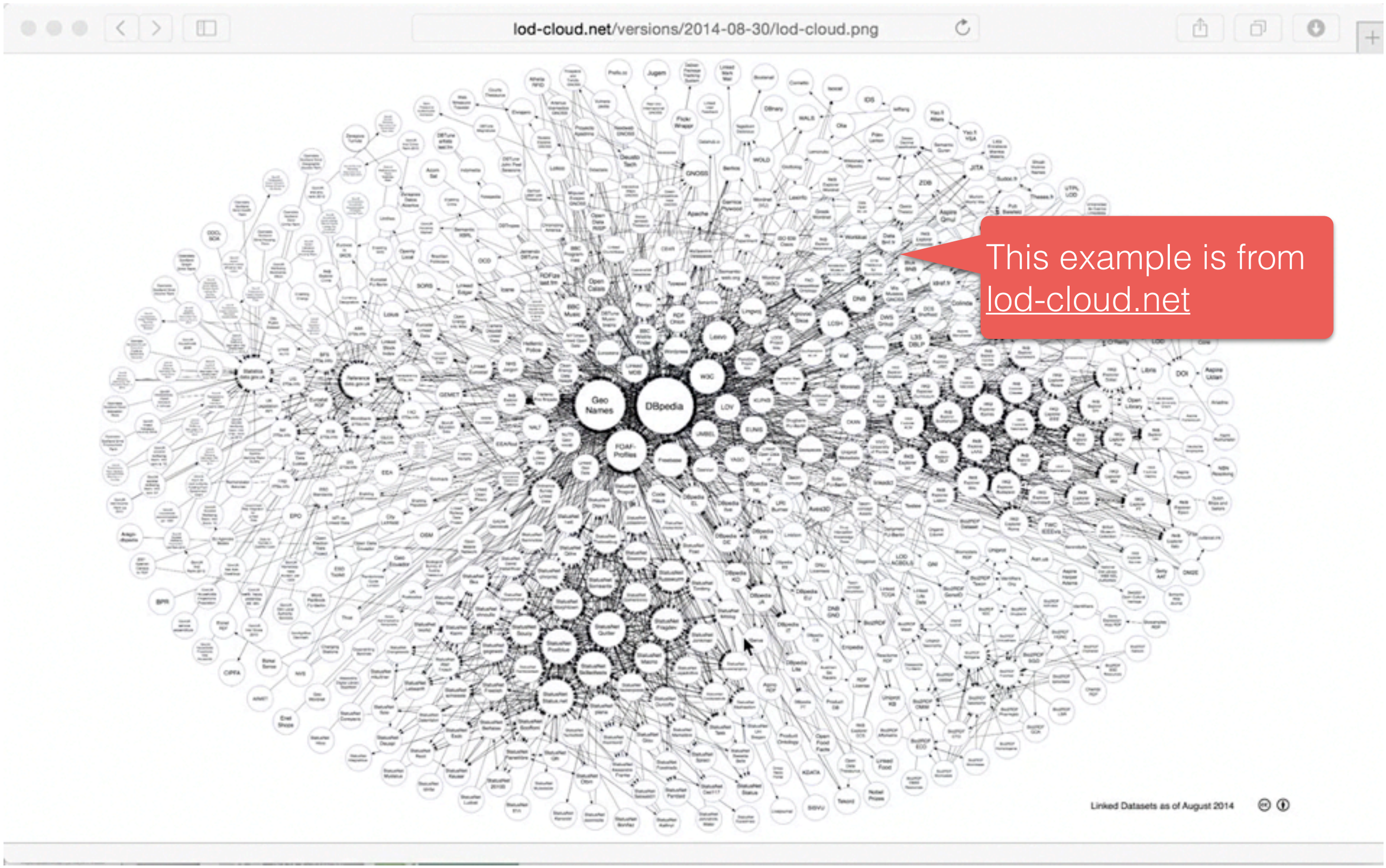
Both pages are about the same resource db:Fukuoka

Linked RDF between Different Data Sources



These are data originated from dbpedia's database (includes longitude, latitude, and much more !!)

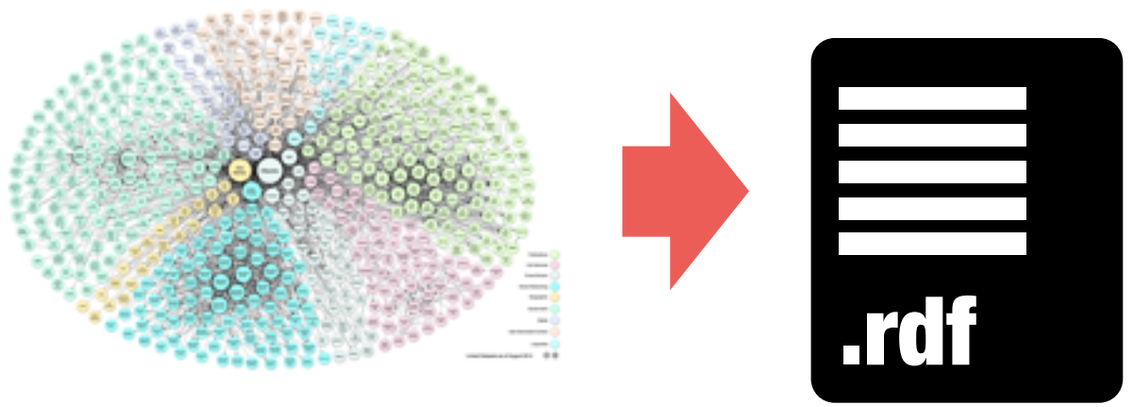
Real Example of Linked RDF



Store RDF Data

How do I **Serialize** my RDF dataset ?

- **N-Triple**
  - Naive representation
- **Turtle / N3**
  - Compact, human-readable
  - Used by SPARQL

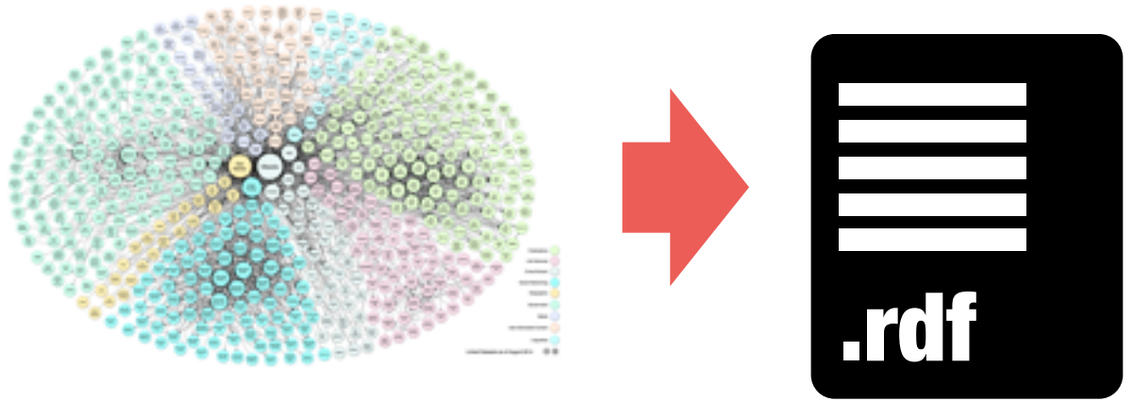


- **RDF/XML** (often referred as RDF)
  - Verbose, hard to read but understood by web browser
- **MicroData** / **RDFa**: RDF annotation for XHTML pages
  - Well-used vocabulary: schema.org
  - Often used for search engine optimization (SEO)

Schema.org
Consortium of major search engine companies, including Google, Microsoft and Yahoo.

How do I **Serialize** my RDF dataset ?

- **N-Triple**
  - Naive representation
- **Turtle / N3**
  - Compact, human-readable
  - Used by SPARQL



- **RDF/XML** (often referred as RDF)
  - Verbose, hard to read but understood by web browser
- **MicroData** / **RDFa**: RDF annotation for XHTML pages
  - Well-used vocabulary: schema.org
  - Often used for search engine optimization (SEO)

Schema.org
Consortium of major search engine companies, including Google, Microsoft and Yahoo.

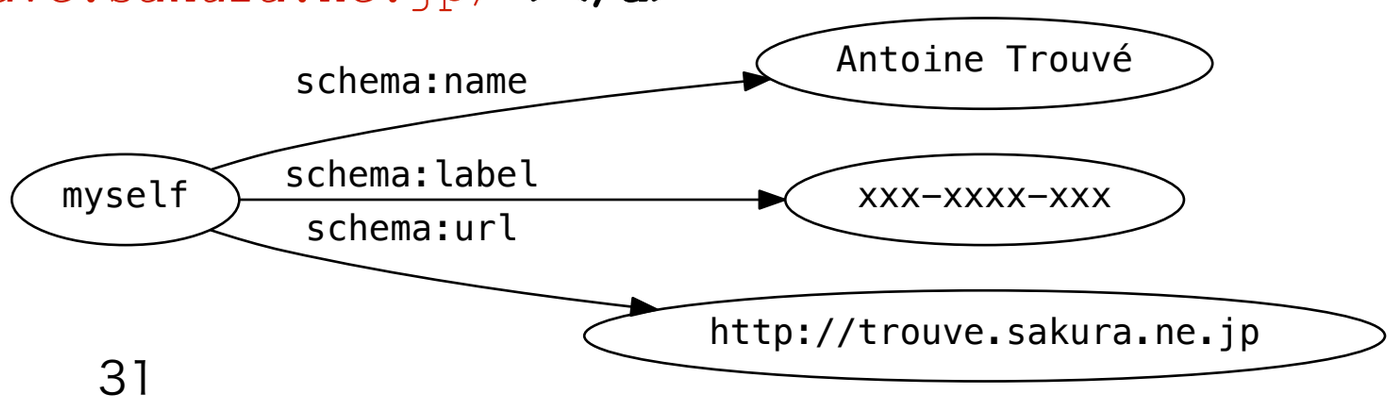
Microdata and RDFa

- Both are techniques to tag information contained in HTML
- Both consist in writing the information right in the HTML code as attributes to HTML elements
  - Microdata: supported by the schema.org consortium (including Google, Microsoft, Yahoo)
  - RDFa (especially RDFa Lite) is the W3C standard

```
<p about="myself" vocab="http://schema.org/" typeof="Person">
  My name is
  <span property="name">Antoine Trouvé</span>,
  my phone number is
  <span property="telephone">xxx-xxxx-xxx</span>
  and my homepage is
  <a property="url" href="http://trouve.sakura.ne.jp/"></a>
</p>
```

Example of RDFa information (use vocabulary from schema.org)

Equivalent RDF graph



Work with RDFa / Microdata

- **Search engines support**
  - Sindice is a search engine for such data, including SPARQL (see later)
  - Google, Yahoo, Bing are able to parse RDFa and Microdata
- **Facebook** uses RDFa (Open Graph API)
  - Extract information from websites
  - Used by features such as the “Like” button

RDFa / Microdata are good for your SEO !

- **Browser support**
  - Some browsers feature plugin that read RDFa data, and even allow to query them with SPARQL
- **Conversion** to/from RDFa and Microdata
  - http://rdf-translator.appspot.com
- Should I use **RDFa or Microdata** ?
  - RDFa enables to express more complex information
  - But Microdata suffices for most tasks
  - Most tools support both, so it does not matter so much !

A First Turtle Example

Subject	Predicate	Object	Language / Type
http://city.fukuoka.lg.jp	http://www.w3.org/2000/01/rdf-schema#type	http://schema.org/WebSite	
http://city.fukuoka.lg.jp	http://schema.org/about	http://dbpedia.org/resource/Fukuoka	
http://city.fukuoka.lg.jp	http://schema.org/lastR		http://www.w3.org/2001/
http://city.fukuoka.lg.jp	http://www 01/rdf-s		
http://city.fukuoka.lg.jp	http://www 01/rdf-s		

- Write the triples as it, separated by dots “.”
- IRI are enclosed within angle brackets “<…>”
- Literals are within quotes “”
- Type / Languages are specified with “^^” and “@”
  - The types “xsd:integer” and “xsd:float” are automatically recognized by Turtle parsers (numbers without quotes)

```
<http://city.fukuoka.lg.jp>
  <http://www.w3.org/2000/01/rdf-schema#type> <http://schema.org/WebSite> .
<http://city.fukuoka.lg.jp>
  <http://schema.org/about> <http://dbpedia.org/resource/Fukuoka> .
<http://city.fukuoka.lg.jp>
  <http://schema.org/lastReviewed> "2015-2-1"^^<http://www.w3.org/2001/XMLSchema#date> .
<http://city.fukuoka.lg.jp>
  <http://www.w3.org/2000/01/rdf-schema#label> "Fukuoka city official homepage"@en .
<http://city.fukuoka.lg.jp>
  <http://www.w3.org/2000/01/rdf-schema#label> "福岡市公式ホームページ"@ja .
```

A First Turtle Example

Subject	Predicate	Object	Language / Type
http://city.fukuoka.lg.jp	http://www.w3.org/2000/01/rdf-schema#type	http://schema.org/WebSite	
http://city.fukuoka.lg.jp	http://schema.org/about	http://dbpedia.org/resource/Fukuoka	
http://city.fukuoka.lg.jp	http://schema.org/lastReviewed		http://www.w3.org/2001/
http://city.fukuoka.lg.jp	http://www.w3.org/2000/	Fukuoka city official homepage	
http://city.fukuoka.lg.jp	df-schema#label	h	
http://city.fukuoka.lg.jp	ww.w3.org/2000/	“福岡市	
	f-schema#label		

- Write the triples as it, separated by dots “.”
- IRIs are enclosed within angle brackets “<…>”
- Literals are within quotes “”
- Type / Languages are specified with “^^” and “@”

```
<http://city.fukuoka.lg.j
  <http://www.w3.org/200      -schema#type> <http://schema.org/WebSite> .
<http://city.fukuoka.lg.
  <http://schema.org/ab      //dbpedia.org/resource/Fukuoka> .
<http://city.fukuoka.lg
  <http://schema.org/last      ed> “2015-2-1”^^<http://www.w3.org/2001/XMLSchema#date> .
<http://city.fukuoka.lg.jp
  <http://www.w3.org/2000      -schema#label> “Fukuoka city official homepage”@en .
<http://city.fukuo
  <http://www.w3.org/2000/01/rdf-schema#label> “福岡市公式ホームページ”@ja .
```

Use Qnames in Turtle

In fact, a slight superset of Qnames; but not CURIE !

```
<http://city.fukuoka.lg.jp>
  <http://www.w3.org/2000/01/rdf-schema#type> <http://schema.org/WebSite> .
<http://city.fukuoka.lg.jp>
  <http://schema.org/about> <http://dbpedia.org/resource/Fukuoka> .
<http://city.fukuoka.lg.jp>
  <http://schema.org/lastReviewed> "2015-2-1"^^<http://www.w3.org/2001/XMLSchema#date> .
<http://city.fukuoka.lg.jp>
  <http://www.w3.org/2000/01/rdf-schema#label> "Fukuoka city official homepage"@en .
<http://city.fukuoka.lg.jp>
  <http://www.w3.org/2000/01/rdf-schema#label> "福岡市公式ホームページ"@ja .
```

Definition of prefixes

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix schema: <http://schema.org/>
prefix db: <http://dbpedia.org/>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
```

Dot-separated triples

```
<http://city.fukuoka.lg.jp> rdfs:type schema:WebSite .
<http://city.fukuoka.lg.jp> schema:about db:Fukuoka .
<http://city.fukuoka.lg.jp> schema:lastReviewed "2015-2-1"^^xsd:date .
<http://city.fukuoka.lg.jp> rdfs:label "Fukuoka city official homepage"@en .
<http://city.fukuoka.lg.jp> rdfs:label "福岡岡市公式ホームページ"@ja .
```

Make Turtle more Compact

Factorize a subject

```
prefix rdfs: <<http://www.w3.org/2000/01/rdf-schema#>
prefix schema: <http://schema.org/>
prefix db: <http://dbpedia.org/>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>

<http://city.fukuoka.lg.jp>
  rdfs:type schema:WebSite ;
  schema:about db:Fukuoka ;
  schema:lastReviewed "2015-2-1"^^xsd:date ;
  rdfs:label "Fukuoka city official homepage"@en ;
  rdfs:label "福岡岡市公式ホームページ"@ja .
```

Factorize subject with semi-colon “;”

The last triple ends with a dot “.”

Factorize a predicate

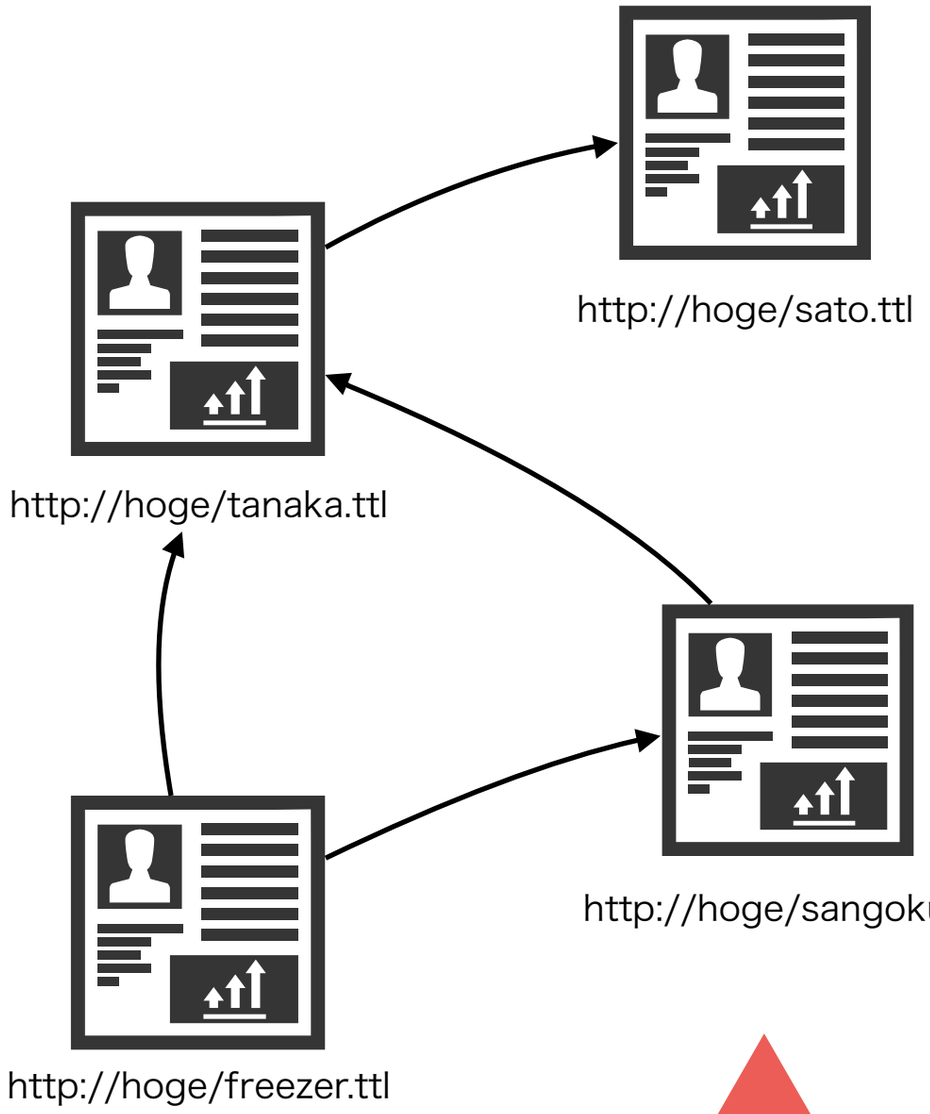
```
prefix rdfs: <<http://www.w3.org/2000/01/rdf-schema#>
prefix schema: <http://schema.org/>
prefix db: <http://dbpedia.org/>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>

<http://city.fukuoka.lg.jp>
  rdfs:type schema:WebSite ;
  schema:about db:Fukuoka ;
  schema:lastReviewed "2015-2-1"^^xsd:date ;
  rdfs:label
    "Fukuoka city official homepage”@en,
    "福岡岡市公式ホームページ"@ja .
```

Factorize predicates with colon “,”

Where do I **store** my RDF data ?

- Small dataset: as a **file**
  - RDF files can be put on the web and build the semantic Web
  - Tool such as Apache Jena (ark) allows to query files with SPARQL
- Large dataset: in a **database**
  - RDF data model is different from the relational one
  - RDF database: **Graph store** or **Triple store**
  - A kind of NoSQL database



The Friend of Friend (FOAF) Project

・The FOAF vocabulary for RDF enables to **express human relationships**

・The FOAF project consists in building a **social network based on distributed data**

Where do I store my RDF data ?

- Small dataset: as a file
  - RDF files can be put on the web and build the s… Web
  - Tool such as A… ena (ark) allows to quer… h SPARQL
- Large dataset: in a database
  - RDF data model is different from the relational one
  - RDF database: Graph store or Triple store
  - A kind of NoSQL database

The SPARQL Query Language for RDF Data

A bit of Background

SPARQL Protocol And RDF Query Language

Hum, this is a recursive acronym 😓

- SPARQL is a W3C standard
  - SPARQL 1.0 (15/1/2008)
  - SPARQL 1.1 (21/3/2013)

It appeared long after RDF itself (it was in 27/3/2000)

- It is supported by most RDF stores and frameworks

Comparison RDF vs. Relational

	Relational Database	RDF
Query language	SQL	**SPARQL**
Database technology	Relational database	Triple store (RDF store, graph store)
Data model	Relational Model	RDF 1.1 Abstract Syntax

An Example of SPARQL Query

Definition of prefixes, same as Turtle

```
prefix db: <http://dbpedia.org/resource/>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?s
WHERE {
   ?s rdfs:label "Fukuoka"@en
}
```

Graph pattern
Same syntax as triples in Turtle

An Example of SPARQL Query

SPARQL keyword. This is a SELECT query (other are CONSTRUCT, ASK, DESCRIBE)

```
prefix db: <http://dbpedia.org/resource/>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?s
WHERE {
   ?s rdfs:label "Fukuoka"@en
}
```

SPARQL keyword

An Example of SPARQL Query

```
prefix db: <http://dbpedia.org/resource/>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?s
WHERE {
  ?s rdfs:label "Fukuoka"@en
}
```

Names of the variables to return

Variable use a **placeholder** in a triple

Graph Pattern in SPARQL

- All SPARQL queries mainly use graph patterns to **match triples**
  - Constants (literals, IRIs) are constraints
  - Variables are placeholders
- A graph pattern consist of triples, using the Triple syntax
- It is evaluated in order
- **Examples:**

  All triples which English rdfs:label is “Fukuoka”

  ?s rdfs:label “Fukuoka”@en .

A more Complex Example of Graph Pattern

All triples which predicate is geo:lon, and which subject appears as subject in triples which rdfs:label is “福岡” in Japanese

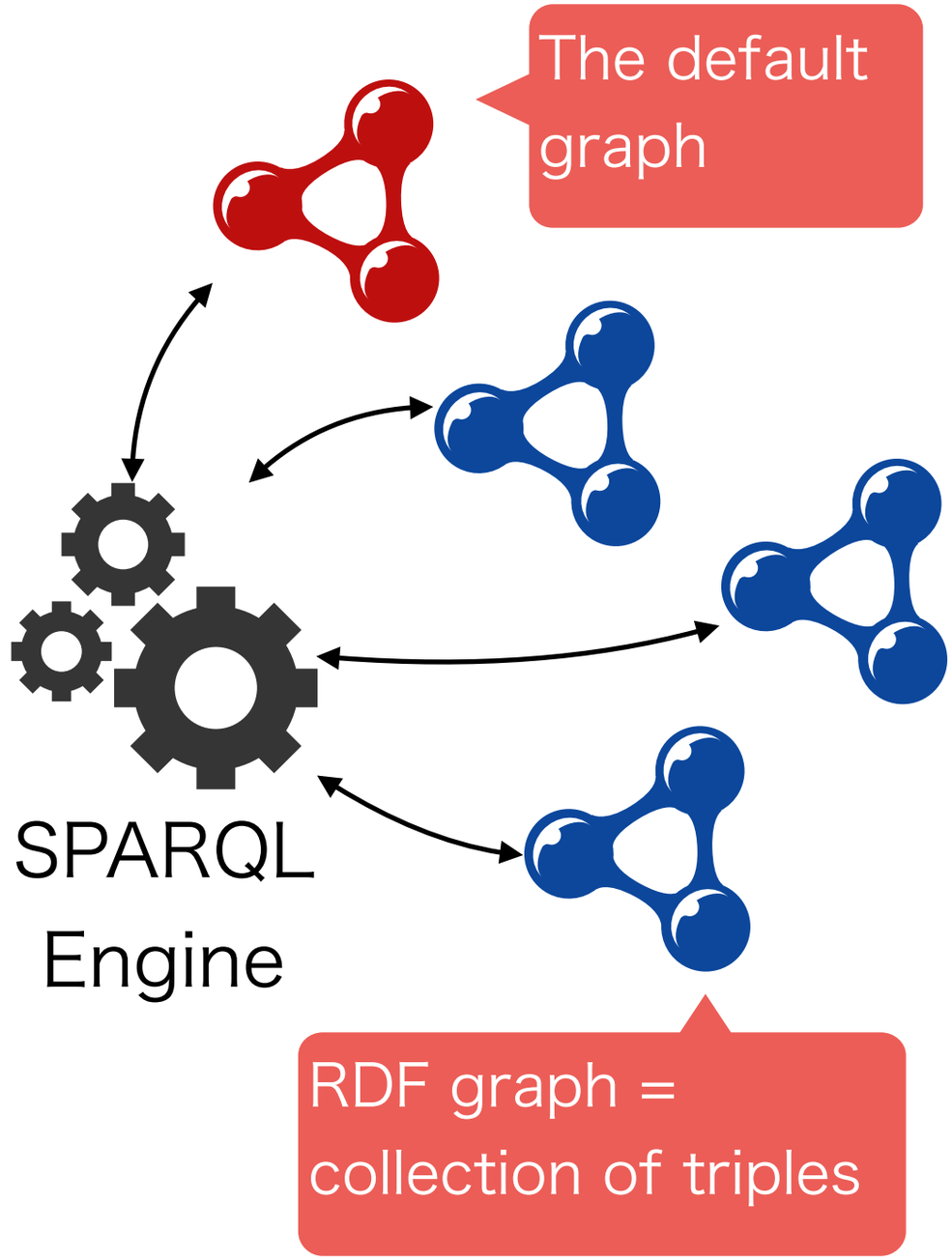
```
?s rdfs:label “福岡”@ja ;
geo:lon ?longitude .
```

That is, the longitude of Fukuoka !

Two triples with the same subject

Graphs in SPARQL

- Graphs are collections of RDF triples
  - They define a **logical partitioning** of the global dataset
- Two types of graphs
  - Named graphs (identified by an IRI)
  - The default, anonymous graph
- It is possible to **specify the graph** in a SPARQL query



Specify the graph with the GRAPH Keyword

```
SELECT ?name ?lon ?lat
WHERE {
   GRAPH <graph IRI> {
      ?s rdfs:label ?name ;
      geo:lon ?lon ;
      geo:lat ?lat .
   }
}
```

・The default graph can not be specified with a GRAPH keyword

・To query the default graph, one has to do so outside any GRAPH block

GRAPH IRI as part of the SPARQL Query

Example: select the IRI of all the named graph in a RDF database

Does not match the default graph

DISTINCT keyword: select only distinct values

```
SELECT DISTINCT ?g
WHERE {
   GRAPH ?g { ?s ?p ?o }
}
```

Store graph IRI in variables

This graph pattern matches any triples

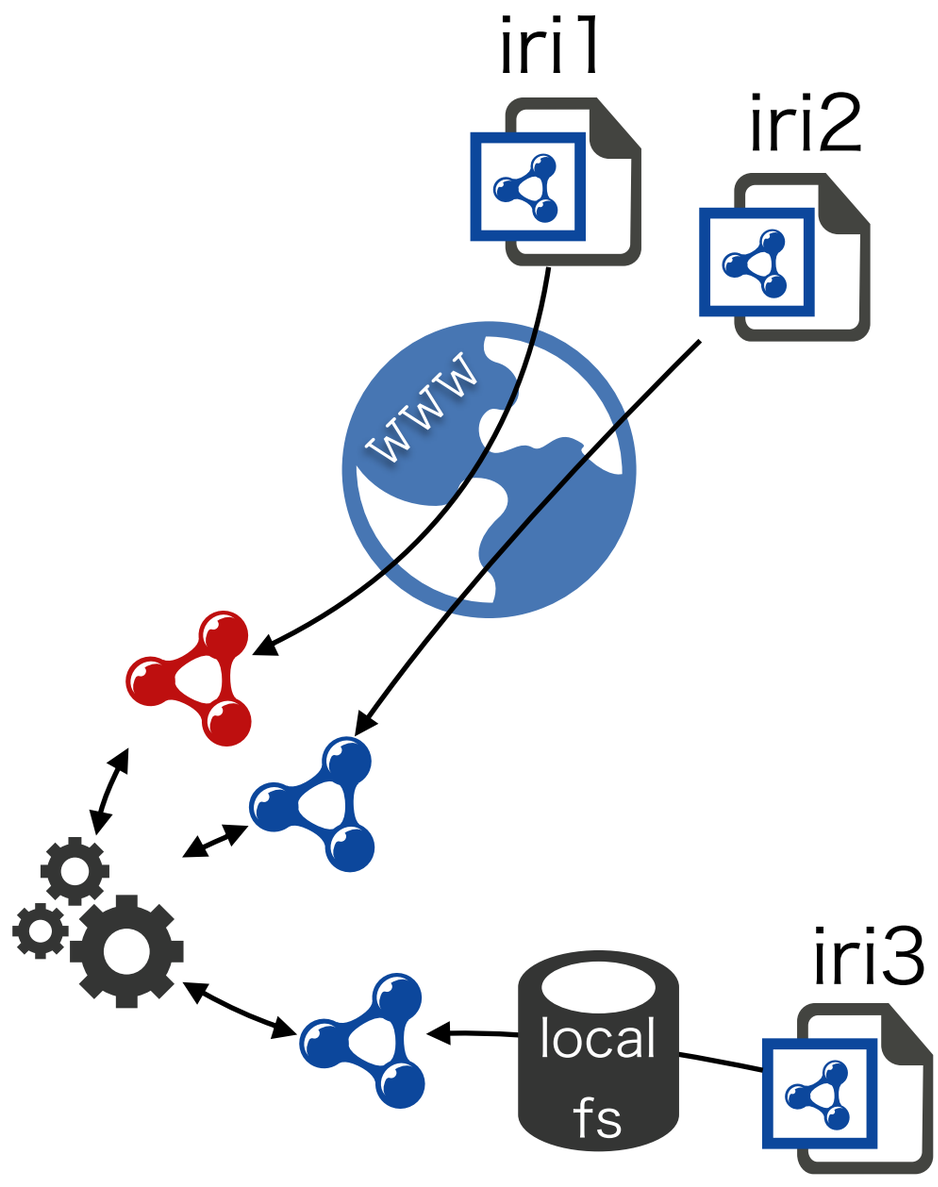
Construct the RDF Dataset with

FROM and FROM NAMED

Specifies the default graph

Specifies the named graphs

```
SELECT ?name
FROM <iri1>
FROM NAMED <iri2>
FROM NAMED <iri3>
WHERE {
   { ?s rdfs:label ?name }
   UNION
   { GRAPH ?g
     { ?s rdfs:label ?name }
   }
 }
```



・The IRI specified in FROM /FROM named designate serialized RDF file

・The IRI may designate local files or remote locations

Allows remote query !

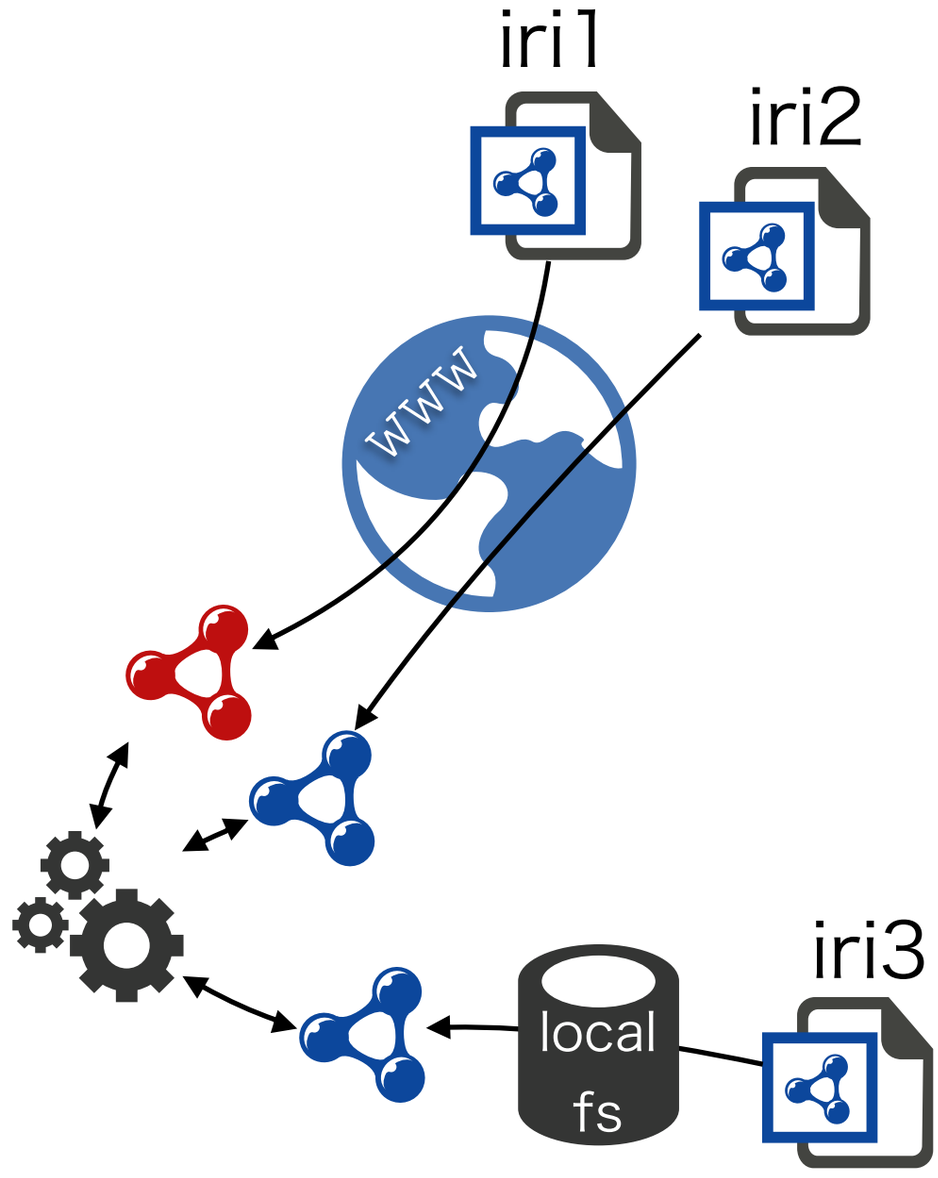
Construct the RDF Dataset with

FROM and FROM NAMED

```
SELECT ?name
FROM <iri1>
FROM NAMED <iri2>
FROM NAMED <iri3>
WHERE {
   { ?s rdfs:label ?name }
   UNION
   { GRAPH ?g
     { ?s rdfs:label ?name }
   }
}
```

Query within data from **iri1**

Query within data from **iri2** and **iri3**

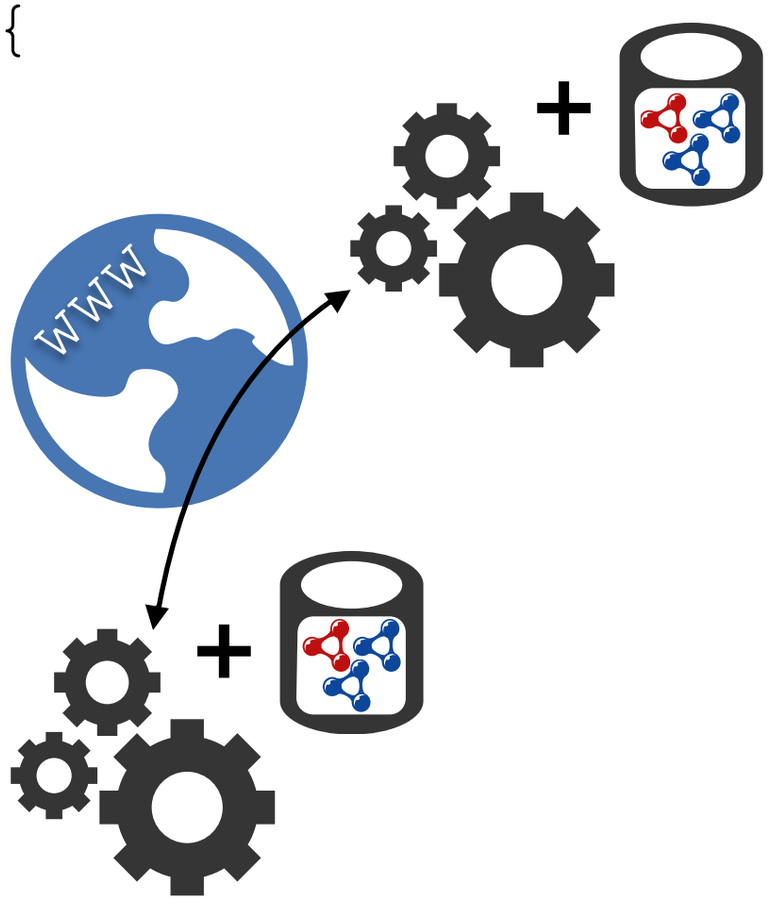


Federated Query with SERVICE

Collaboration between SPARQL endpoints

The URL of dbpedia SPARQL endpoint

```
SELECT DISTINCT ?name
WHERE {
   SERVICE <http://dbpedia.org/sparql> {
      ?s rdfs:label ?name
   }
}
```



・SPARQL enables to distribute the process of a query between physically separated RDF datasets

・This allows do **mashup right inside a single SPARQL query**

Federated Query with SERVICE

Collaboration between SPARQL endpoints

The URL of dbpedia SPARQL endpoint

```
SELECT DISTINCT ?name
WHERE {
  SERVICE <http://dbpedia.org/sparql> {
    ?s rdfs:label ?name
  }
}
```

・SPARQL enables to distribute the process of a query between physically separated RDF datasets

・This allows do **mashup right inside a single SPARQL query**

Not only a query language

SPARQL Protocol And RDF Query Language

- SPARQL 1.1 defines two kinds of protocols
  - An HTTP REST API to submit SPARQL queries, with two urls
  - A protocol for SPARQL endpoints to discuss between each other
- Let's take a look at the HTTP REST API

The SPARQL HTTP REST API

- Given an API endpoint http://endpoint.org
- SPARQL queries
  - http://endpoint.org**/sparql**
    - Submit a read-only SPARQL query (SELECT/CONSTRUCT/ASK/DESCRIBE)
  - http://endpoint.org**/update**
    - Submit an update SPARQL query (LOAD/INSERT/DELETE/DROP/COPY/MOVE)
- Direct action on RDF data (at url http://endpoint.org?graph=graph_name)
  - **GET** request: returns a whole graph
  - **PUT** request: replaces a whole graph
  - **POST** request: adds triples to a graph
  - **DELETE** request: deletes a graph

More on CONSTRUCT/INSERT on next slide

Most triple stores organize the database in datasets, accessible via http://endpoint.org/dataset

CONSTRUCT and INSERT Queries

- The two following queries have similar syntax
  - **CONSTRUCT**: generates in output new triples derived from the current RDF dataset
  - **INSERT**: inserts to the RDF database new triples derived from the current RDF dataset
- They are often used for
  - ETL (Extract / Tranform / Load)
  - Refactoring (e.g. change vocabulary)
  - Ontology inference (see next)

An Example of INSERT Query

Use CONSTRUCT instead of INSERT for a construct query

A graph pattern to construct new triples

```
INSERT
{
  ?a foaf:friend ?b ;
     foaf:knows ?b .
  ?b foaf:knows ?a .
}
WHERE
{ ?b foaf:friend ?a }
```

A graph pattern to match triples and store data to variables

- This query uses the vocabulary friend of a friend (foaf)
- This query stances that if ?a is friend with ?b then
  - the opposite is also true
  - they know each other

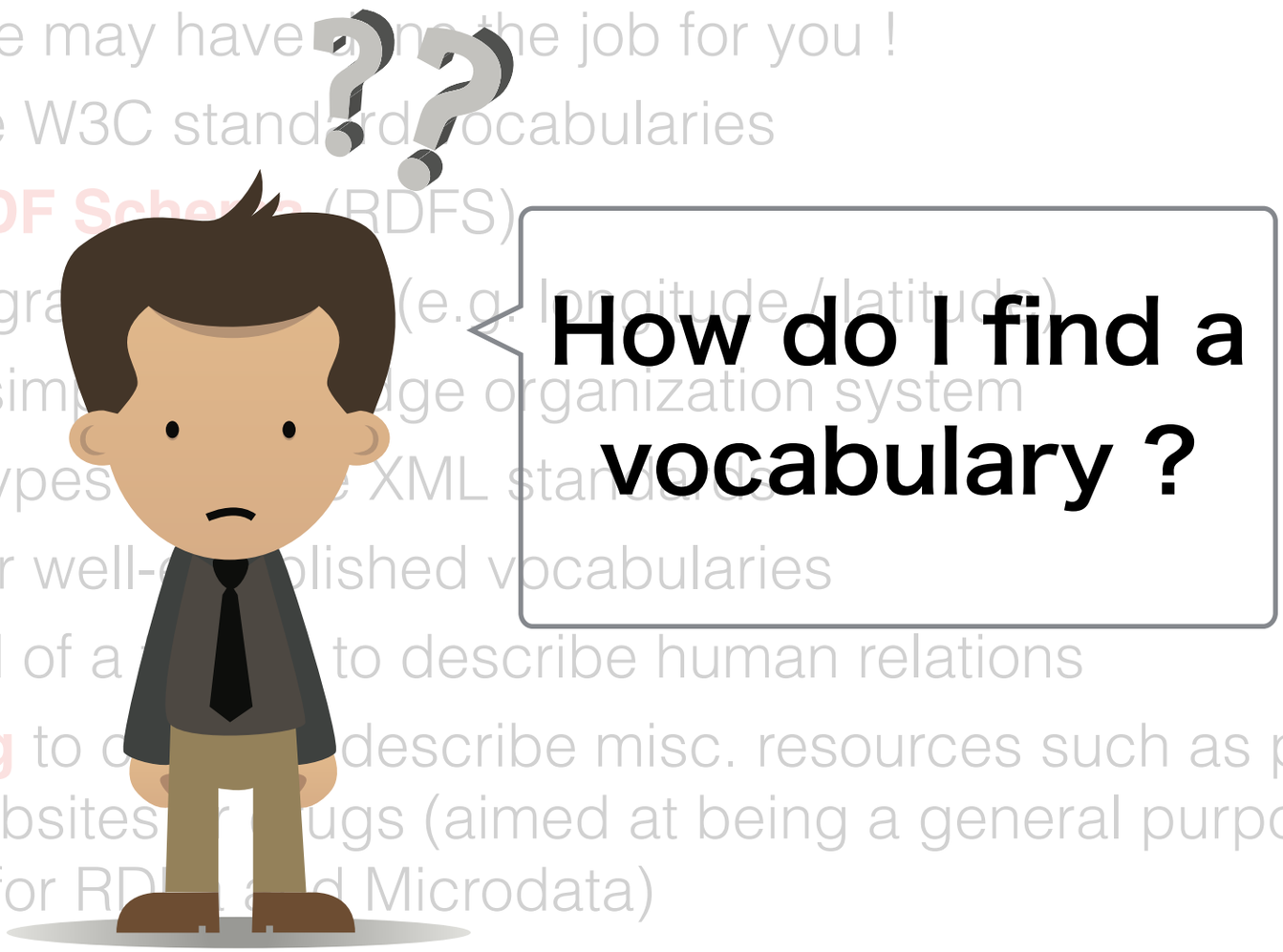
Vocabulary, RDF Schema and Ontology

About Vocabulary

- Depending on the data you hold, you may need various vocabulary
  - You may create your own
  - But someone may have done the job for you !
- There are some W3C standard vocabularies
  - **RDF** and **RDF Schema** (RDFS)
  - **geo** for geographical data (e.g. longitude / latitude)
  - **SKOS**, the simple knowledge organization system
  - **XSD**, data types from the XML standards
- And some other well-established vocabularies
  - **Foaf** (Friend of a friend) to describe human relations
  - **Schema.org** to casually describe misc. resources such as public facilities, websites or drugs (aimed at being a general purpose vocabulary for RDFa and Microdata)
  - **Dublin Core** to describe bibliographical resources
  - **DBPedia**, **Yago**, two general-purpose vocabularies, used for online encyclopedia

About Vocabulary

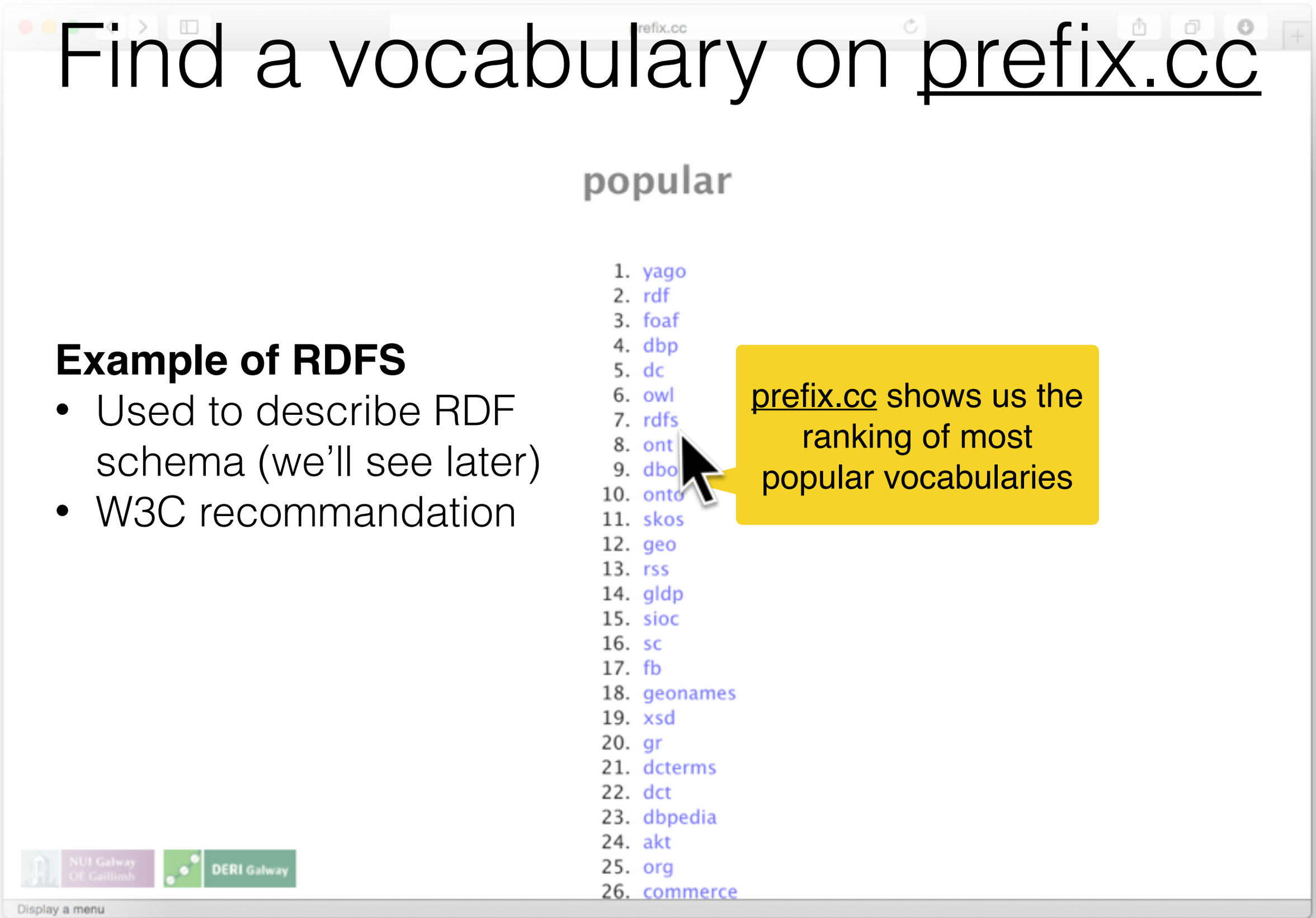
- Depending on the data you hold, you may need various vocabulary
  - You may create your own
  - But someone may have done the job for you !
- There are some W3C standard vocabularies
  - **RDF** and **RDF Schema** (RDFS)
  - **geo** for geography (e.g. longitude /latitude)
  - **SKOS**, the simple knowledge organization system
  - **XSD**, data types for the XML standards
- And some other well-established vocabularies
  - **Foaf** (Friend of a ...) to describe human relations
  - **Schema.org** to describe misc. resources such as public facilities, websites or blogs (aimed at being a general purpose vocabulary for RDFa and Microdata)
  - **Dublin Core** to describe bibliographical resources
  - **DBPedia**, **Yago**, two general-purpose vocabularies, used for online encyclopedia



Find a vocabulary on prefix.cc

Example of RDFS

- Used to describe RDF schema (we'll see later)
- W3C recommandation



prefix.cc

rdfs

http://www.w3.org/2000/01/rdf-schema#

+1
−1

Add alternative URI

ttl xml rdfa sparql txt json jsonld vann | lov | prefix.cc

NUI Galway
OÉ Gaillimh

DERI Galway

Display a menu

w3.org

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://www.w3.org/2000/01/rdf-schema#> a owl:Ontology ;
        dc:title "The RDF Schema vocabulary (RDFS)" .

rdfs:Resource a rdfs:Class ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
        rdfs:label "Resource" ;
        rdfs:comment "The class resource, everything." .

rdfs:Class a rdfs:Class ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
        rdfs:label "Class" ;
        rdfs:comment "The class of classes." ;
        rdfs:subClassOf rdfs:Resource .

rdfs:subClassOf a rdf:Property ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
        rdfs:label "subClassOf" ;
        rdfs:comment "The subject is a subclass of a class." ;
        rdfs:range rdfs:Class ;
        rdfs:domain rdfs:Class .

rdfs:subPropertyOf a rdf:Property ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
        rdfs:label "subPropertyOf" ;
        rdfs:comment "The subject is a subproperty of a property." ;
        rdfs:range rdf:Property ;
        rdfs:domain rdf:Property .

rdfs:comment a rdf:Property ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
        rdfs:label "comment" ;
        rdfs:comment "A description of the subject resource." ;
        rdfs:domain rdfs:Resource ;
        rdfs:range rdfs:Literal .

rdfs:label a rdf:Property ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
        rdfs:label "label" ;
        rdfs:comment "A human-readable name for the subject." ;
        rdfs:domain rdfs:Resource ;
        rdfs:range rdfs:Literal .
```

We get the turtle version of the vocabulary !

Display a menu

w3.org

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://www.w3.org/2000/01/rdf-schema#> a owl:Ontology ;
        dc:title "The RDF Schema vocabulary (RDFS)" .

rdfs:Resource a rdfs:Class ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
        rdfs:label "Resource" ;
        rdfs:comment "The class resource, everything."

rdfs:Class a rdfs:Class ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schem
        rdfs:label "Class" ;
        rdfs:comment "The class of classes." ;
        rdfs:subClassOf rdfs:Resource .

rdfs:subClassOf a rdf:Property ;
        rdfs:isDefinedBy <http://www.w                    ema#> ;
        rdfs:label "subClassOf" ;
        rdfs:comment "The subject is a                    " ;
        rdfs:range rdfs:Class ;
        rdfs:domain rdfs:Class .

rdfs:subPropertyOf a rdf:Property ;
        rdfs:isDefinedBy <http://www.w3.or                -schema#> ;
        rdfs:label "subPropertyOf" ;
        rdfs:comment "The subject is a subpro             f a property." ;
        rdfs:range rdf:Property ;
        rdfs:domain rdf:Property .

rdfs:comment a rdf:Property ;
        rdfs:isDefinedBy <http://www.w3.or                -schema#> ;
        rdfs:label "comment" ;
        rdfs:comment "A description of the                urce." ;
        rdfs:domain rdfs:Resource ;
        rdfs:range rdfs:Literal .

rdfs:label a rdf:Property ;
        rdfs:isDefinedBy <http://www.w3                   chema#> ;
        rdfs:label "label" ;
        rdfs:comment "A human-readable name for the subject." ;
        rdfs:domain rdfs:Resource ;
        rdfs:range rdfs:Literal .
```

Wait, how come a RDF vocabulary is described in RDF ?

We get the turtle version of the vocabulary !

Display a menu

Data Schema in RDF

- It is possible to describe the Schema of RDF data
  - We call it an **Ontology**
- The schema itself is stored in RDF, using some standard vocabulary (W3C recommandation)
  - **RDFS**: The simplest vocabulary
  - **OWL**: Very complex, and complete
  - **SPIN**: express rules using SPARQL
- These Ontology languages are real language
  - Toward model-driven development
- It is important to define the ontology in your RDF database so that **anyone can understand your data**

It is possible to express a large part of programs right in the ontology !

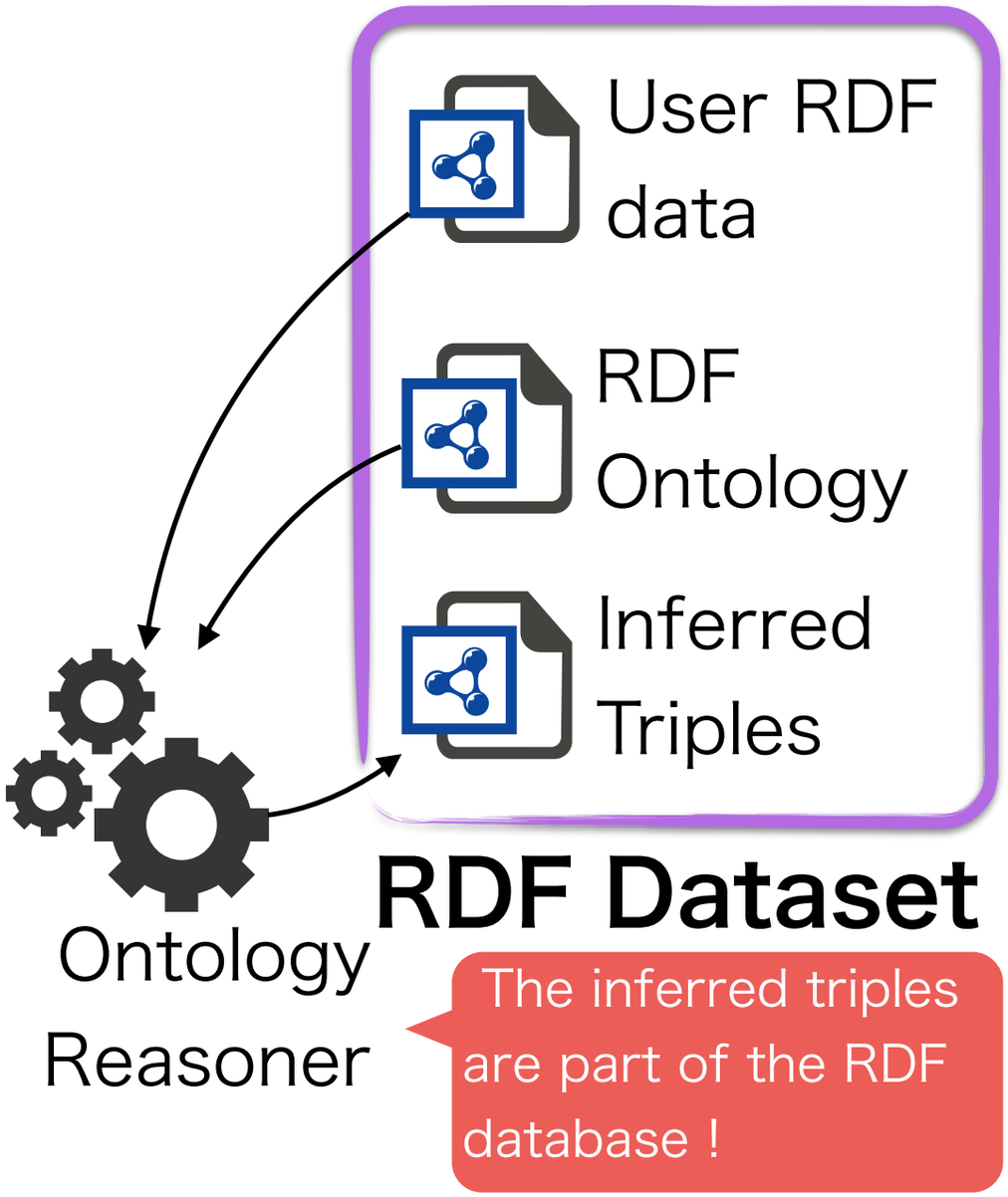
Basics of RDF Schema

- Similar to **object-oriented languages**
  - RDF resource have classes (typing system)
  - Types are organized in hierarchy of subclass / superclass
  - The kind of properties that an object of a given class can accept is well defined
- But with **some differences**
  - An RDF resource may have more than one class
  - Properties are first-class objects, that is, the properties of an RDF resource define its type
- Yet, this makes **object mapping** super-easy
  - For example the library “dotnetrdf” enables direct mapping between RDF objects and C# classes

How Ontologies are used by Triple Stores ?

- SPARQL engine do not (usually) check the Ontology on the fly
- Instead, one use Ontology reasoner to generate extra RDF triples
- This is called **inference**
- Inference rules can also be expressed in SPARQL (CONSTRUCT query)

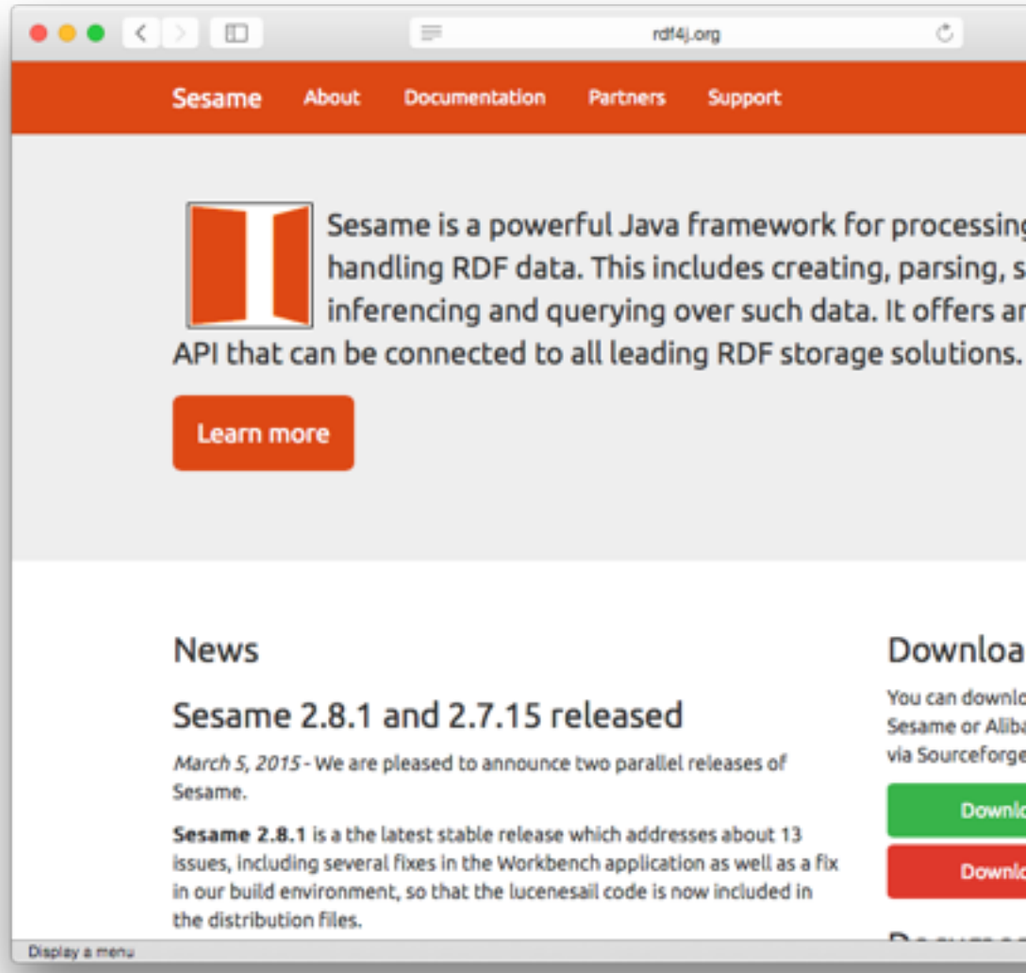
SPIN is a vocabulary that enable to use ontology rules written in SPARQL



Some Triple Stores

Sesame (rdf4j.org)

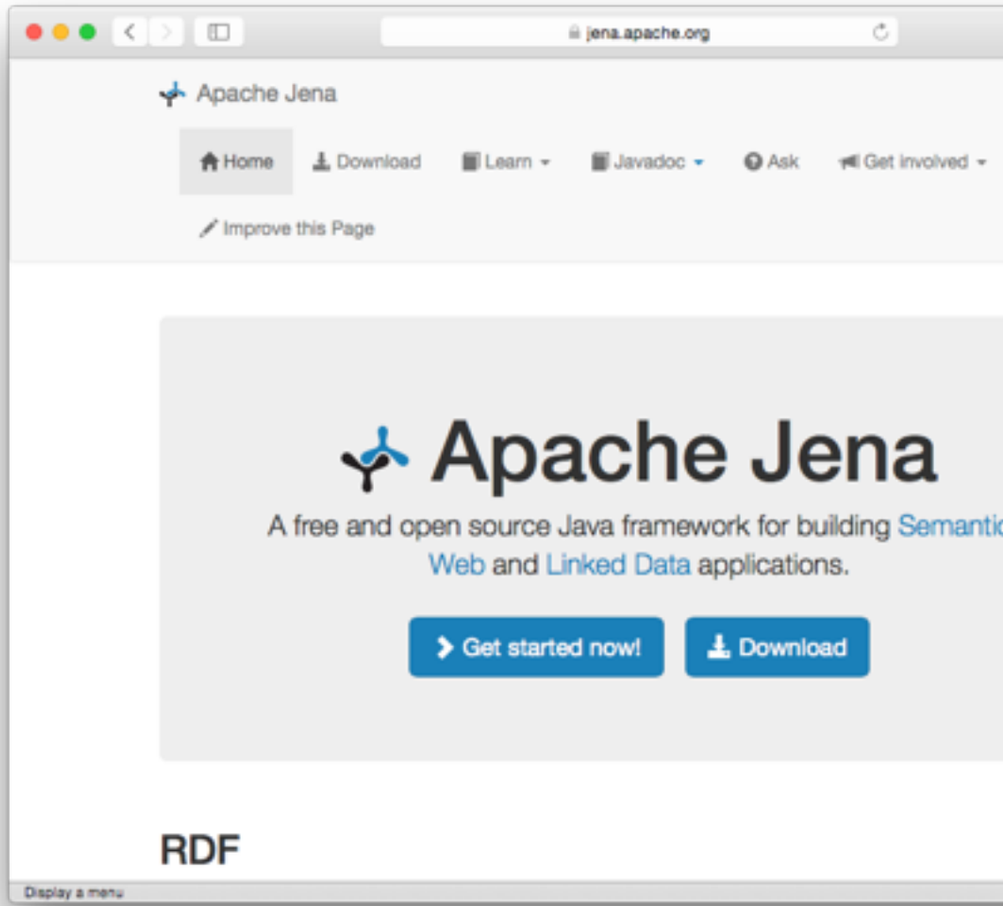
- Open-source, written in Java
- Supports plugins
- Several functionalities
  - Java RDF framework to programmatically work with RDF data
  - Triple Store Server (Java weblet for servers such as Tomcat or Jetty)
  - Inference in RDFS (not OWL)
- Originally developed as a research project
  - European Union project **On-To-Knowledge** (2000-2002)
  - Developed by the company Aduna (Dutch) for the
- Distributed as Java weblet (war)



Apache Jena (jena.apache.org)

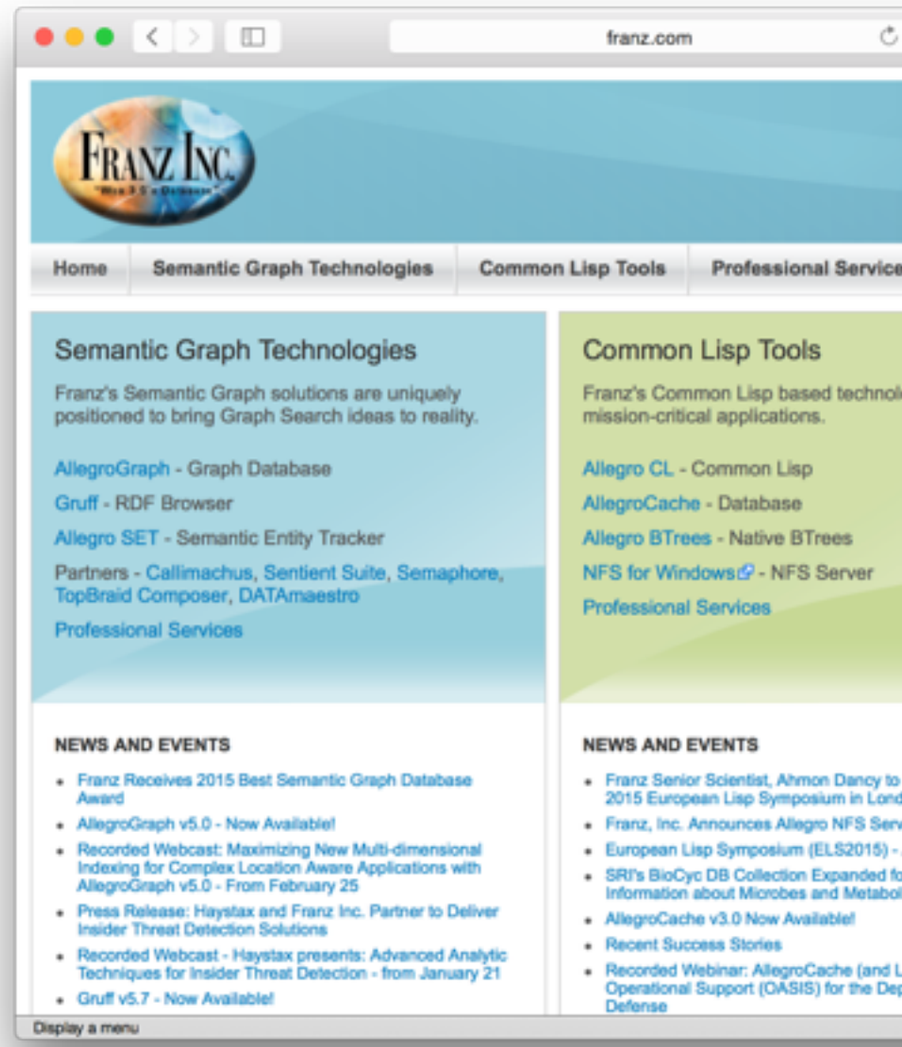
- Open source, written in Java
- Several functionalities
  - Java framework to manipulate RDF data
  - Triple store server
  - Inference in RDFS ans OWL
- Research project
  - From **Hewlett-Packard**'s Semantic Web Research Lab
  - The most popular project among researcher, therefore supports several cutting-edge plugins

Stand alone: makes it super-easy to install and run ! (a single jar)



AllegroGraph (franz.com)

- Closed-source, written in LISP
  - Bindings in most language
- Commercial database from Franz.inc
  - High performance
  - Powerful inference (RDFS, RDFS++)



Virtuoso (virtuoso.openlinksw.com)

- Open source, written in C
- Originated from the Finish database ecosystem in 1998
- Not only for RDF, also supports relational data
  - Supports RDF and SPARQL through mapping to relational model and SQL
- Multi-purpose server, notably:
  - Database (based on object-relational model)
  - Web application server
  - Web content management system
- Usually seen as the fastest and most scalable triple store (used by dbpedia)
- However it lacks powerful inference functionality like Apache Jena and Sesame

Conclusion

The RDF Developper Toolbox

A triple store

A RDF Library

Enable to programmatically manipulate RDF data.
Example: “dotnetrdf” for C#, “C RDF lirabry” for C, “RDFLib” for Python

A RDF graph visualizer

Example: “Gruff” from the company Franz

An ontology reasoner

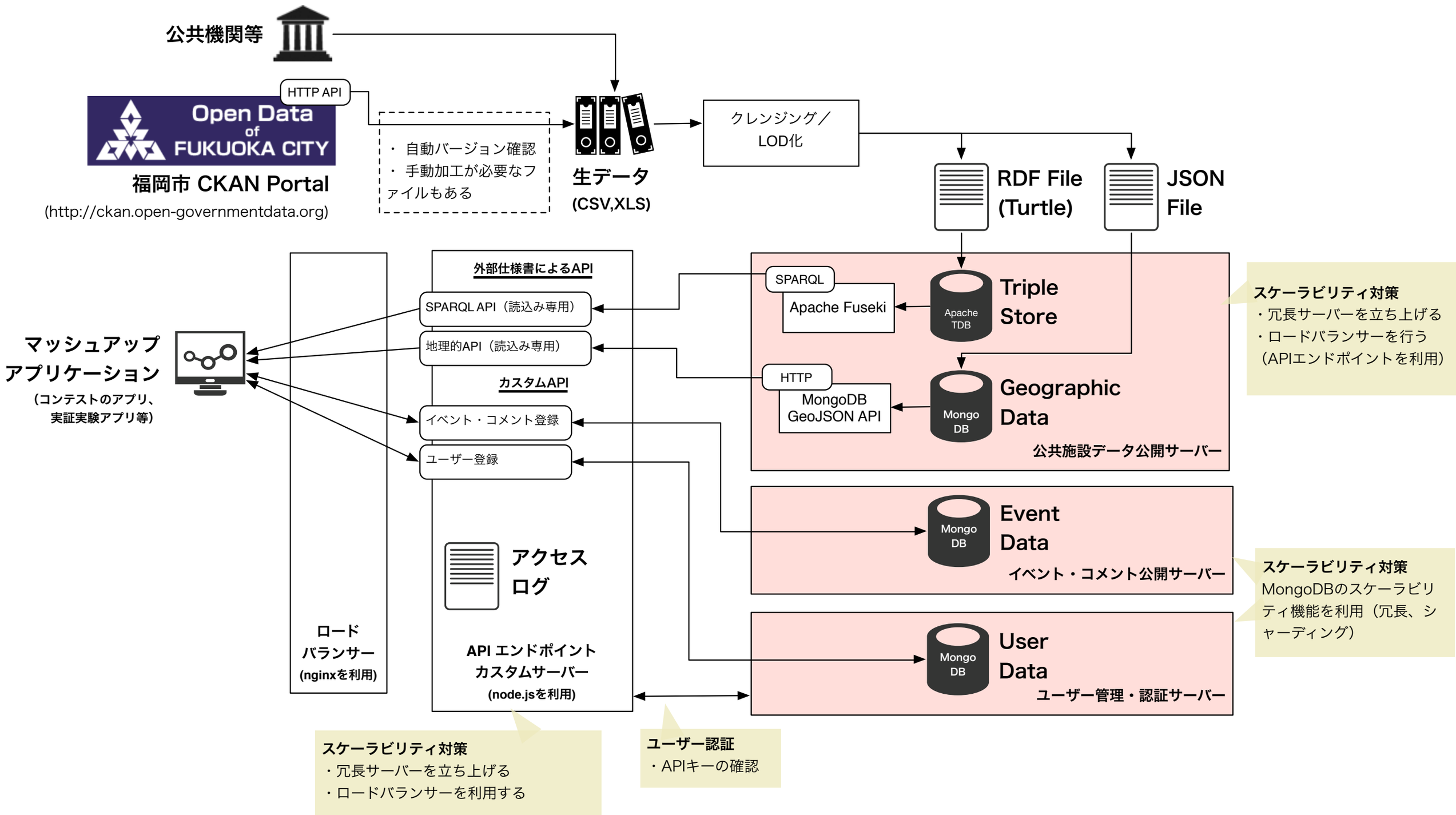
In most triple stores

An ontology debugger

Example: “Protégé” from Stanford University

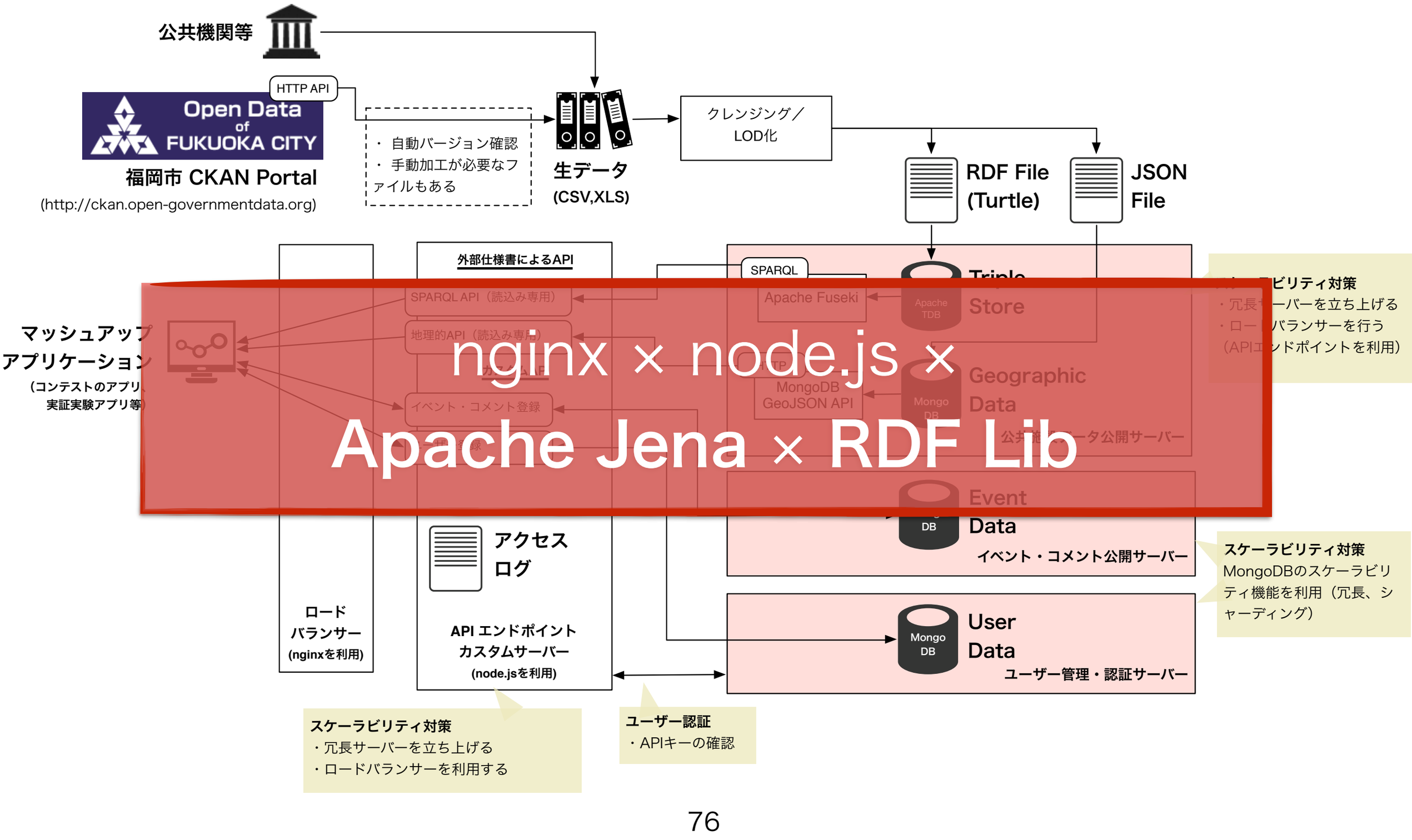
Example of System

from teapot at BODIC.org



Example of System

from teapot at BODIC.org



nginx × node.js ×

Apache Jena × RDF Lib

General conclusion…

Give it a try !

Thank you very much