BODIC.orgと SPARQL

2015/11/23, 北九州学研都市第15回産学連携フェア

トルヴェ　アントワン

九州大学

九州先端科学技術研究所

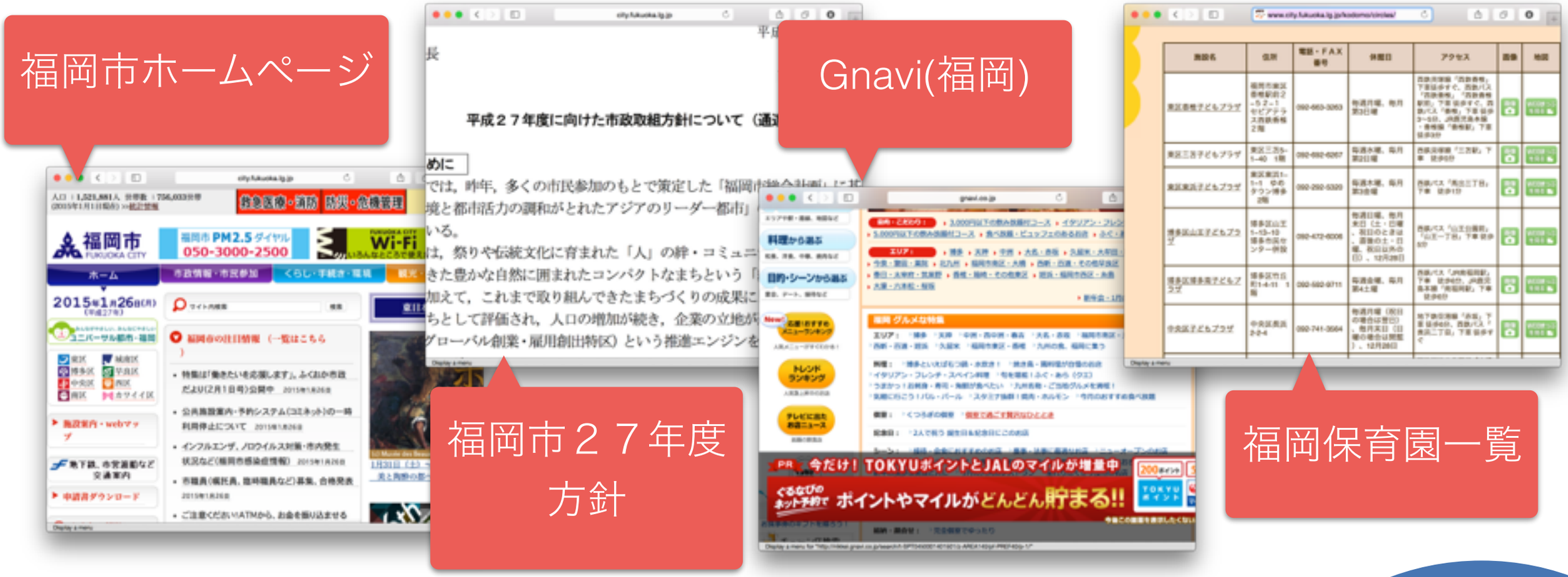
http://trouve.sakura.ne.jp

本日の技術

データモデル	**RDF** (Ressource-Description Framework)
クエリー言語	**SPARQL** (SPARQL Protocol and RDF Language)
スキーマ言語 (データの構成を記述するため)	**RDFS** (RDF Schema), **OWL** (Web Ontology Language)
データベース技術	**Triple (Graph) Store**

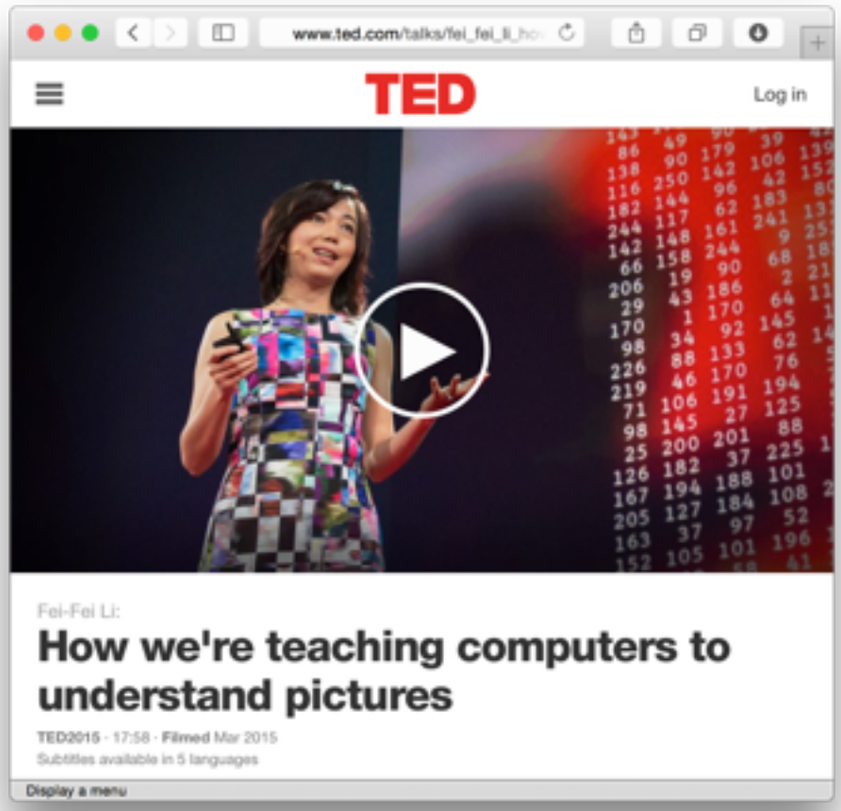
The RDF Data Model

RDFの背景



- WWWには膨大の情報がある
- しかしながら、ほとんどは**構造のない情報**
- 人間には問題なくその情報を解析できるが
  コンピュータは違う

コンピュータがWWW情報を理解するため、どうすればいいですか？



解決策１

アルゴリズムなど（例：機械学習）を使って、コンピュータを賢くする

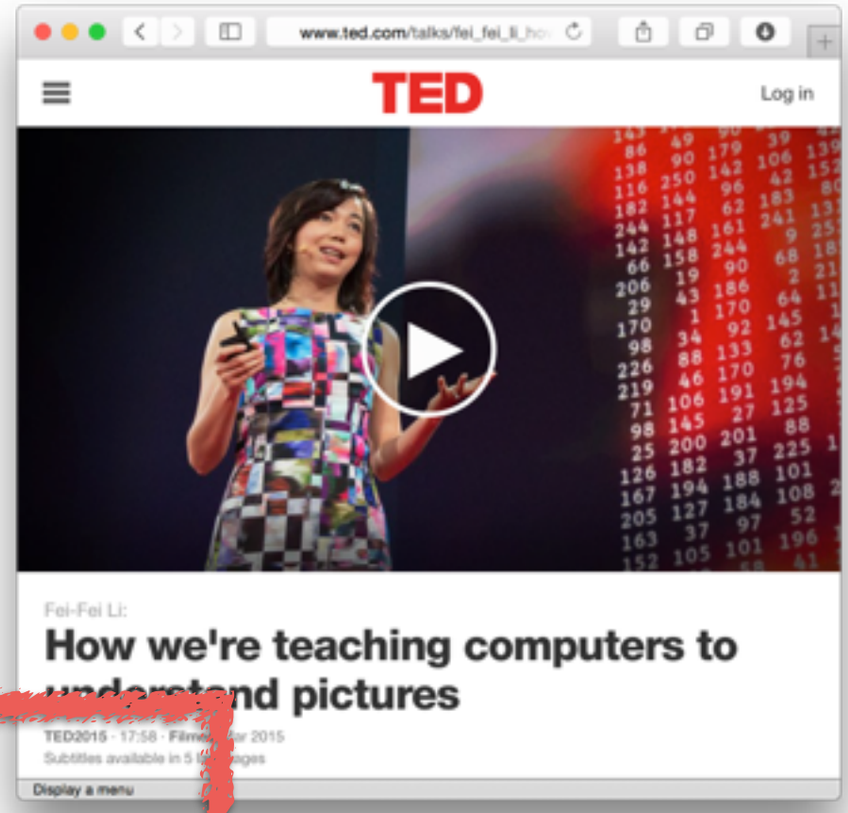
解決策２

WWWにある情報を手動（半自動）で構造化する

コンピュータがWWW情報を理解するため、どうすればいいですか？

解決策１

アルゴリズムなど（例：機械学習）を使って、コンピュータを賢くする



解決策２

WWWにある情報を手動（半自動）で構造化する

WWWにある情報をKey Valueで保存

リソースID

http://city.fukuoka.lg.jp

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

キー

値

福岡市ホームページ

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

主語	述語	目的語
リソースのID	プロパティ	プロパティ値

例: 前のスライドのサンプル、トルプルで表現した場合

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

W3Cについて

- 1994年に設立
- WWWで使われている技術の規格を管理する
  - HTML, XML, Javascript, CSS, **RDF**
- RDFはW3C規格である
  - 最新版はRDF 1.1（2014/2/25に発表）
- RDF規格の中に更に諸々な規格が定義されている

・Tim Berners-Lee, head of the W3C.

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

RDFの実例

主語	述語	目的語	言語・タイプ
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

このアドレス**IRI**:
Internationalized
Resource **I**dentifier
国際リソースID

RDFの実例

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/dateModified	“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)

IRIは読みにくい！

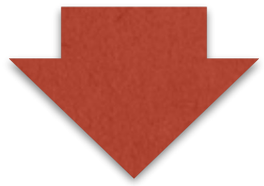
QnameとCURIE: IRIが読み

やすくなるように

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

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

共通プレフィックス



プレフックス

rdfs:type

rdfs:label

ローカル部分

・**CURIE**: スラッシュ「/」を使える

・**Qname**:スラッシュ「/」を使えない

CURIEを使ったRDF実例

主語	述語	目的語	言語・タイプ
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

リソースのIRI

主語	述語	目的語	言語・タイプ
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	[illegible]15-2-1”	xsd:date
http://city.fukuoka.lg.jp	schema:name	“福岡市”@en	en
http://city.fukuoka.lg.jp	schema:name	“福岡市”@ja	ja

・このIRIはサイトのURLではなく実世界に「福岡市」というリソースを示す
・誰でもIRIを作っても構いませんが、できるだけ既存のIRIを使った方がデータの利用者にとって使いやすい
・IRIとしてURLを使うことが多い（アクセスするとリソースについての情報が表示）

リソースはウエブサイトですのでIRIとしてサイトURLを使うのは無難

語彙におけるIRI

主語	述語	目的語	言語・タイプ
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

述語はIRIである

これはschema.orgという語彙の言葉を示すIRI

これは日付というタイプ（型）を示すIRI

- IRIは人物に加えて、語彙を示すこともある
- 語彙は意味がちゃんと定義されている言葉・概念のこと（人間言語に依存せずに定義する）
- その言葉は主に述語とタイプとして使う
- 自分の語彙を定義しても構いませんが、既存の語彙を使った方がデータ利用者に優しい

RDFの細かい機能：述語の重複

主語	述語	目的語	言語・タイプ
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

- RDFデータは何度も同じ述語を指定しても大丈夫です
  - よくあるユースケース：名前を複数言語で入れたいときに

リテラルの言語・タイプについて

主語	述語	目的語	言語・タイプ
http://city.fukuoka.lg.jp	rdfs:type	schema:WebSite	
http://city.fukuoka.lg.jp	schema:about	db:Fukuoka	
		“2015-2-1”	xsd:date
		“Fukuoka city official homepage”	en
		“福岡市公式ホームページ"	ja

タイプの例

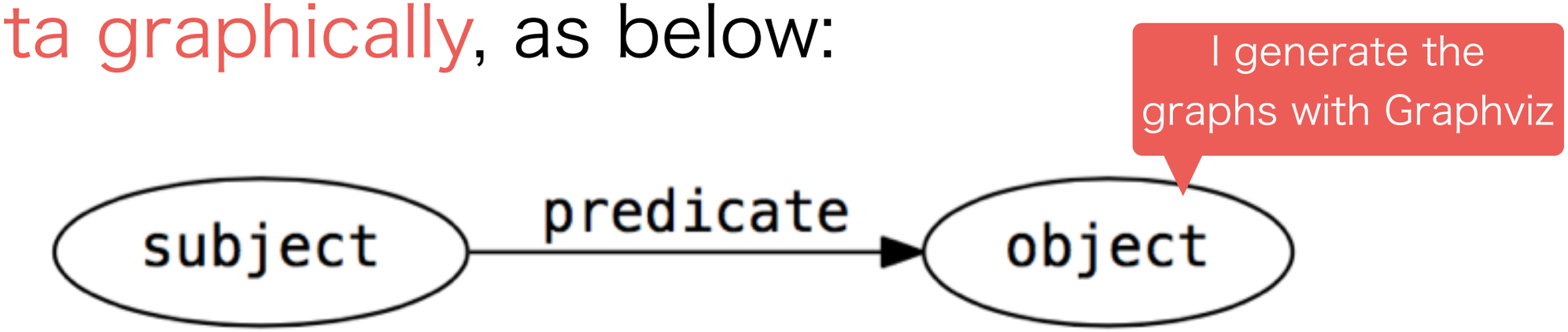
言語の例

・リテラルはIRIではない者
・引用符に囲むが、文字列ではないものがある
　・言語を指定すると「言語付き文字列」として特別に扱う（言語はISO 639で指定）
　・更にタイプも指定できる

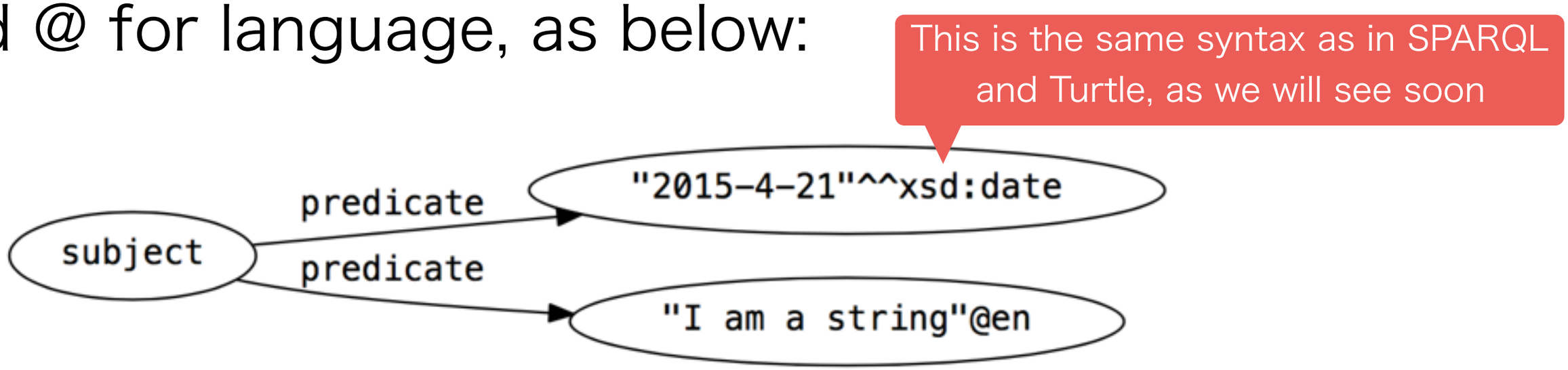
- RDF規格はXML標準タイプを含む:
  - xsd:integer, xsd:decimal, xsd:float, xsd:double, xsd:string, xsd:boolean, xsd:dateTime, xsd:date, xsd:time

RDFグラフ

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

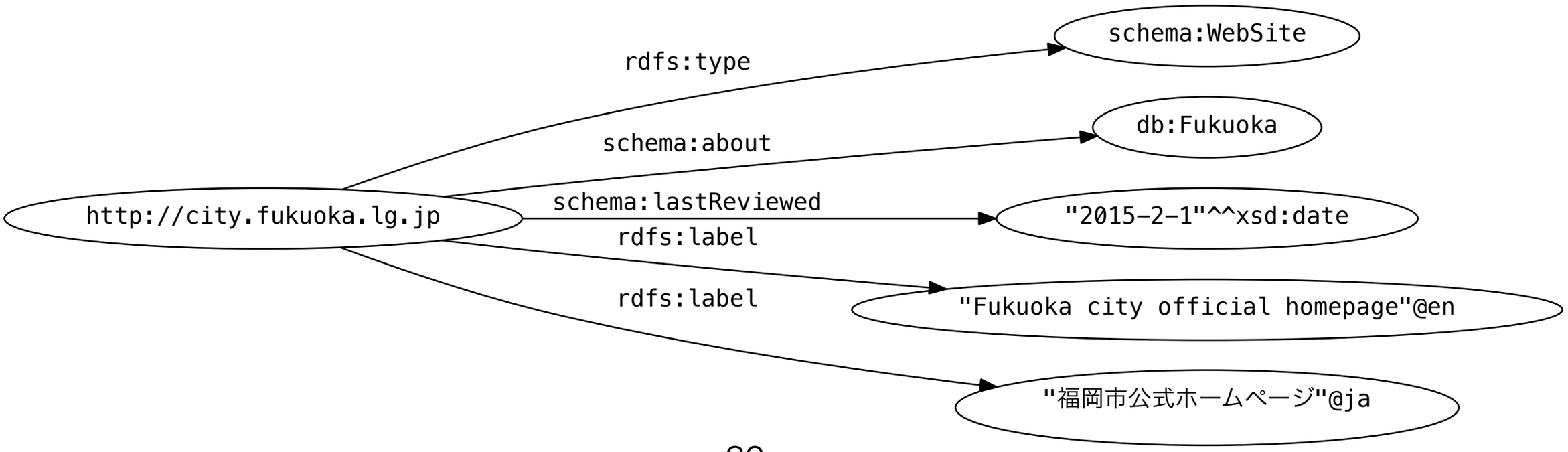


Literals' tags are represented with "^^" for datatypes, and @ for language, as below:



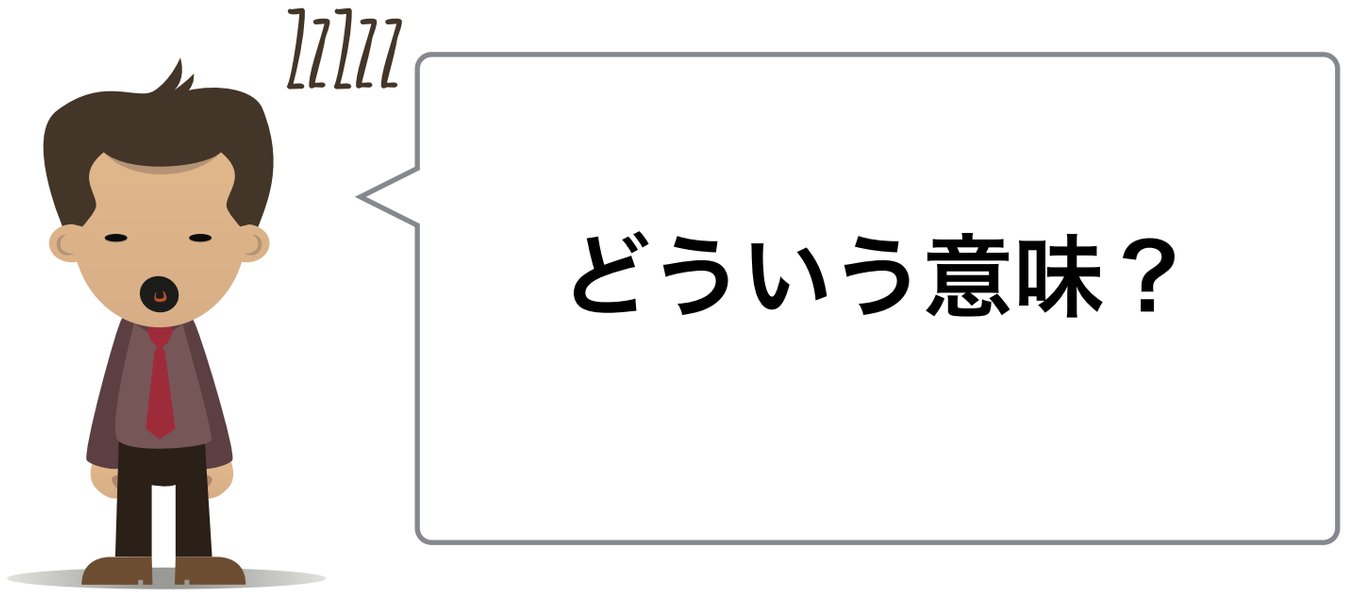
An Example of Graph

主語	述語	目的語	言語・タイプ
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



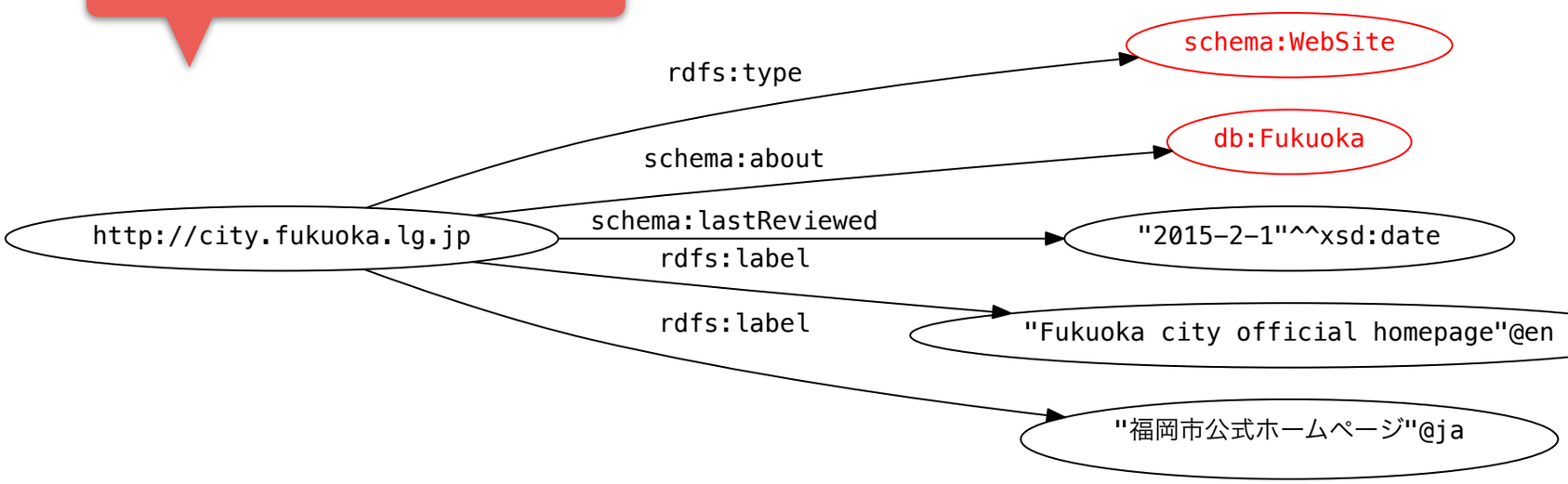
LODについて

- 同じIRIを再利用すると、リーソス間にリンクを貼ることができる
- 特に同じIRIが目的語と主語として使われている時
- **LOD**は**L**inked **O**pen **D**ataの略語です

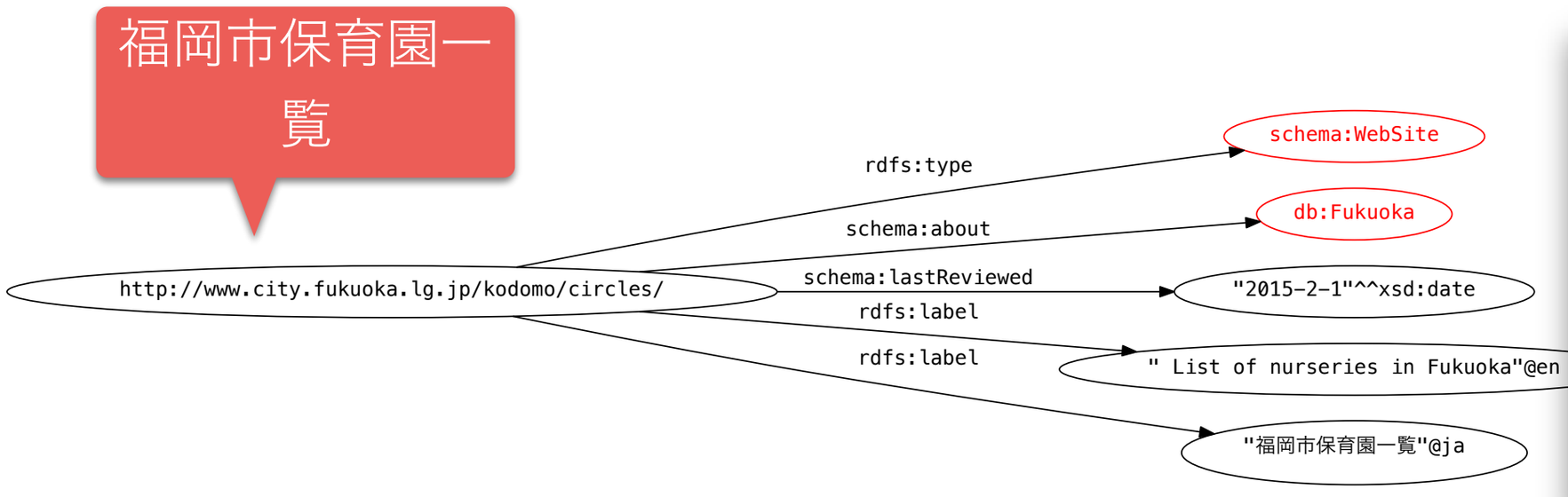


LODグラフの例（1）

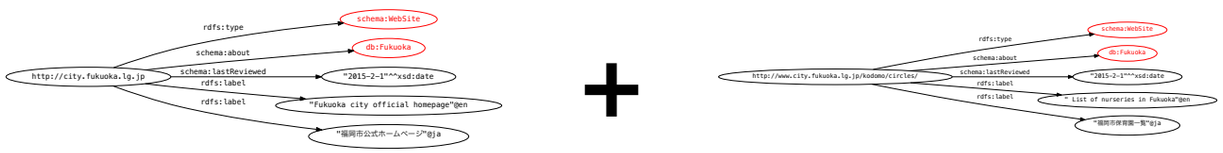
福岡市ホームページ



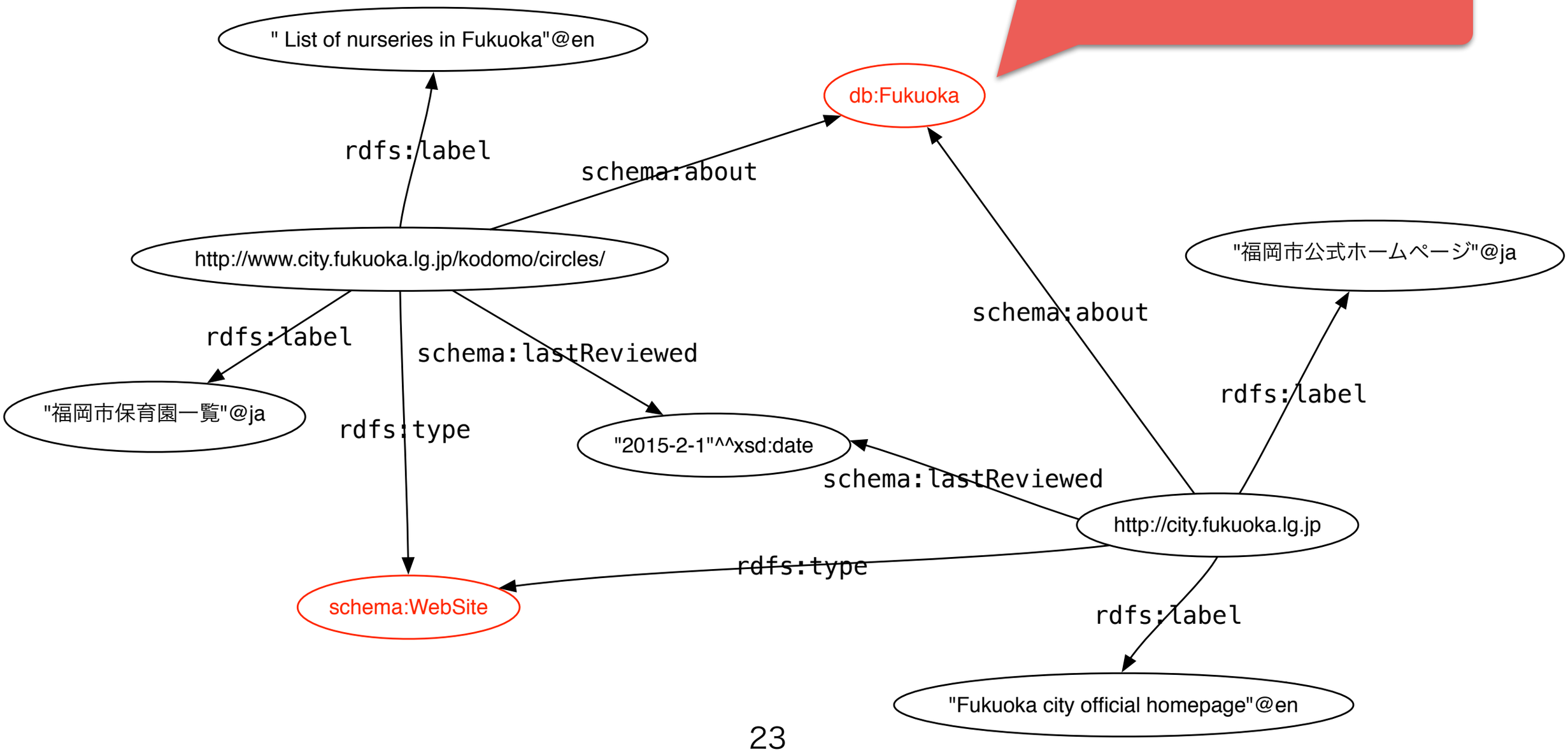
福岡市保育園一覧



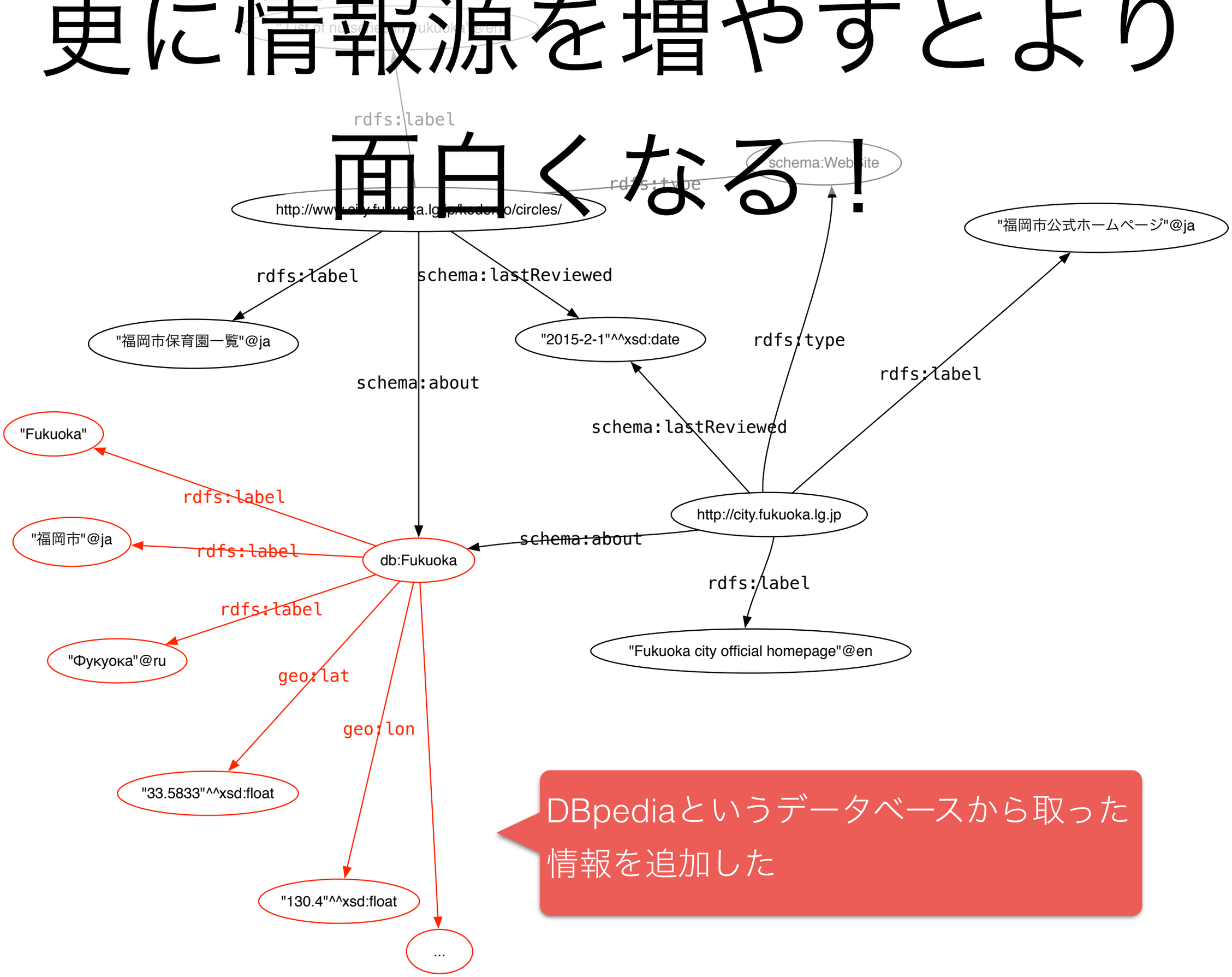
LODグラフの例（1）



両方福岡市について

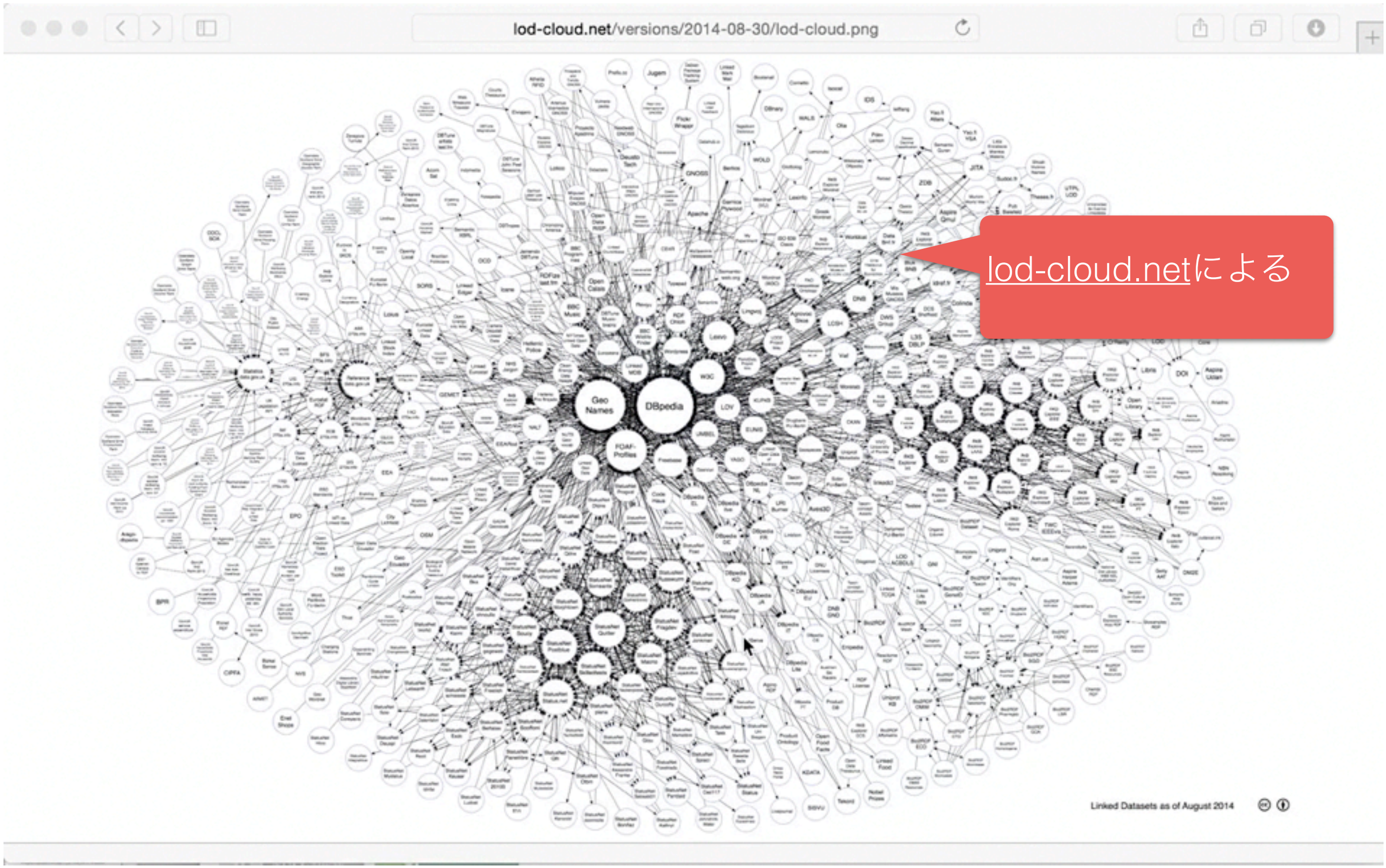


更に情報源を増やすとより面白くなる!



DBpediaというデータベースから取った情報を追加した

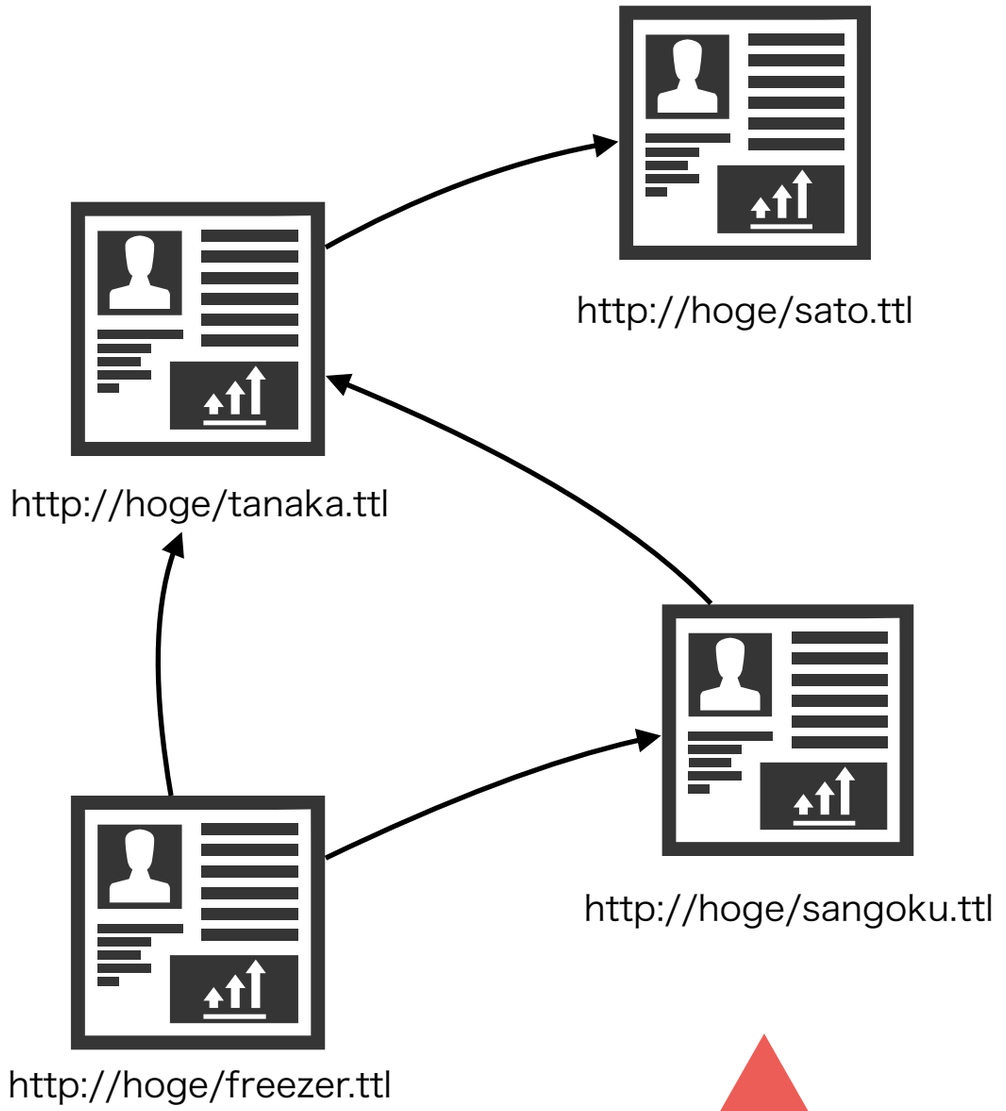
LODグラフの実例


lod-cloud.net/versions/2014-08-30/lod-cloud.png
lod-cloud.netによる
Geo Names
DBpedia
FOAF-Profiles
Linked Datasets as of August 2014

Serialize RDF Data

どこでRDFを保管すればいい？

- データが小さい場合：**テキストファイル**
  - テキスト形式でRDFファイルをオンラインで置くだけでセマンティクウエブに参加できる！
  - そのようなファイルを検索できるツールもある！
- データが多い場合：**データベース**
  - RDFデータベース：**グラフストア**又は**トリプルストア**/**RDFストア**という
  - NoSQLデータベースの１種類



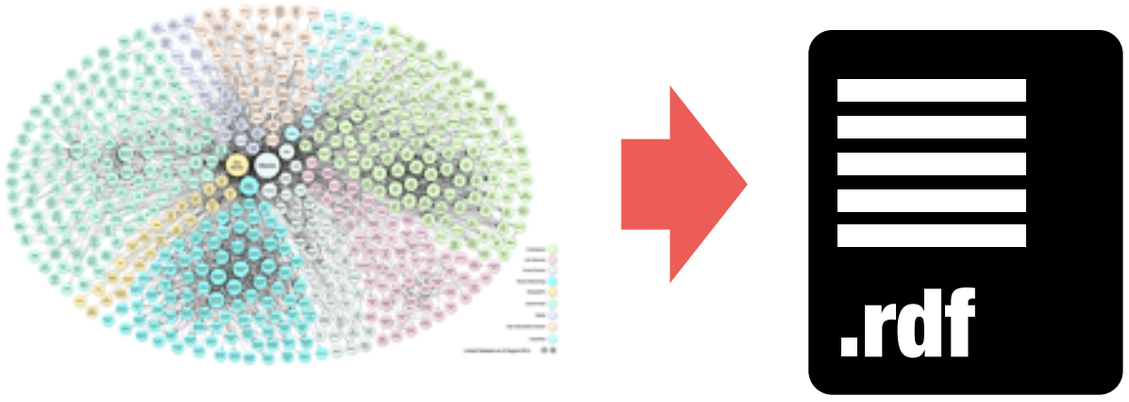
Friend of Friend (FOAF)プロジェクトについて

・FOAF語彙は**人間関係**をできるような語彙

・FOAFプロジェクトは分散SNSを構築しようとしている

RDFのテキスト形式

- **N-Triple**
  - トリプルを並べるだけ
- **Turtle / N3**
  - N-Tripleを読みやすくしたもの
  - SPARQLはTurtleに近いシンタックスを使う
- **RDF/XML**
  - Turtleが出る前に一番一般的なテキスト形式でした
  - Turtleと比べると文字数が多い
- **MicroData / RDFa**: HTMLページにRDFデータを組み込めるため
  - SEOに効果的！



グーグルはRDFaとMicroDataを解析しています！

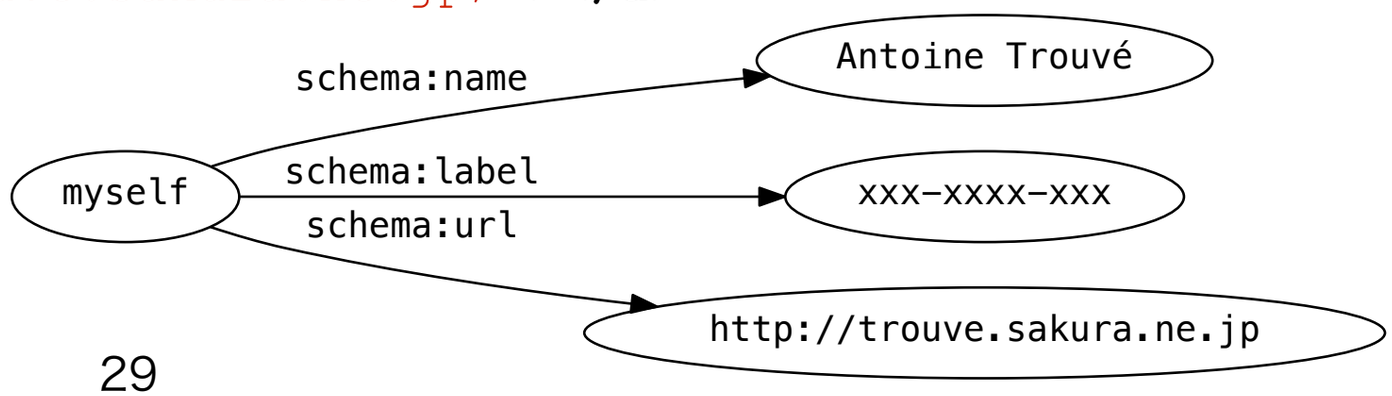
Microdata と RDFa

- HTMLページにRDFトリプルを組み込むためのフォーマット
- Microdata
  - CSSクラス名を使う
  - シンプルだが、表現できないRDF情報がある
  - schema.org consortium が提案(メンバー：Google, Microsoft, Yahoo等)
- RDFa (especially RDFa Lite)
  - W3C規格
  - 例：

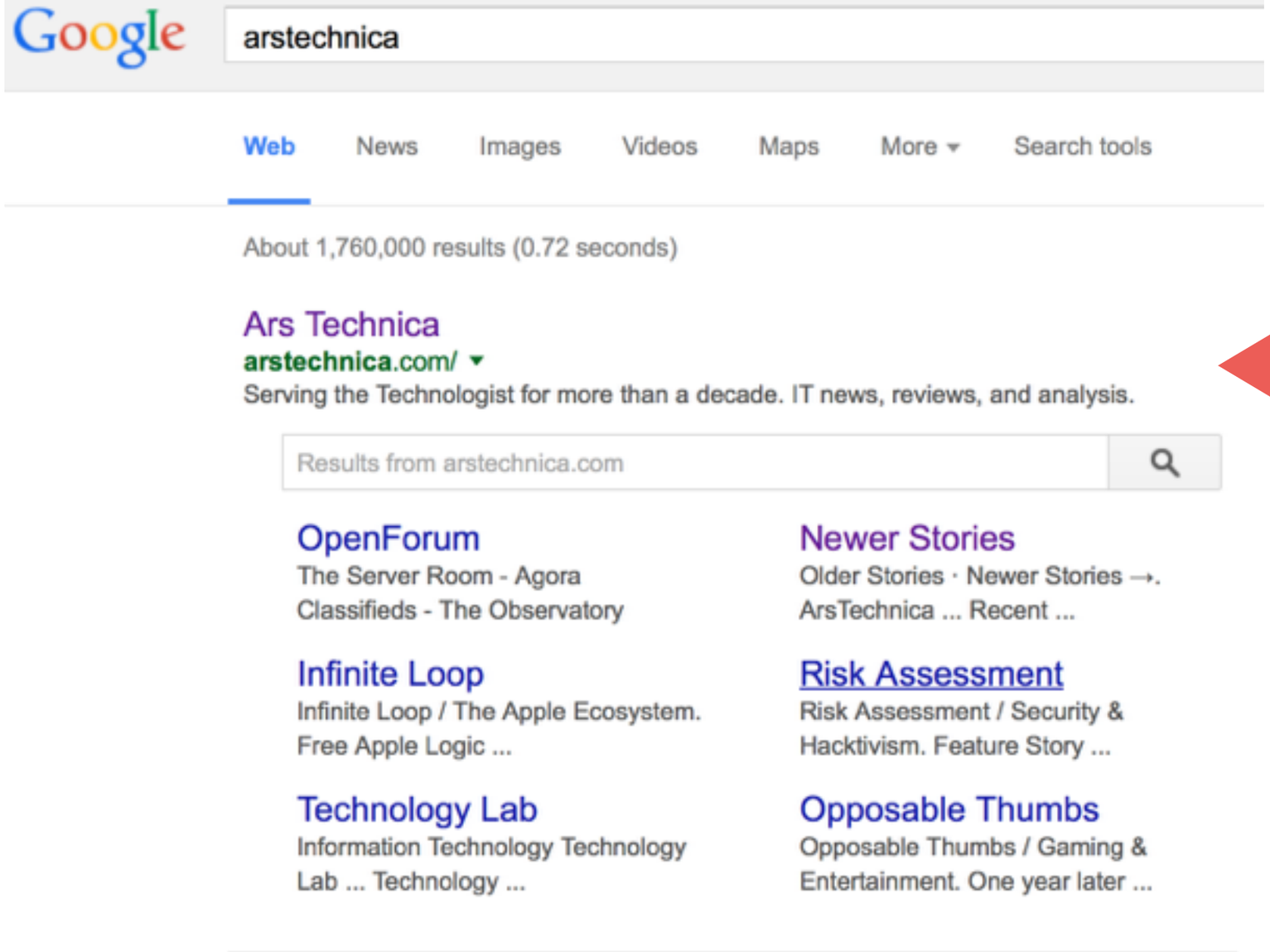
```
<p about="myself" vocab="http://schema.org/" typeof="Person">
  My name is
  <span property="name">Antoine Trouvé</span>,
  my phone number is
  <span property="telephone">xxx-xxxx-xxx</span>
  and my homepage is
  <a property="url" href="http://trouve.sakura.ne.jp/"></a>
</p>
```

schema.orh語彙を利用する場合

同等グラフ



グーグルにおけるRDFa/Microdata情報の扱い



グーグルのユーザーが見る検索結果をコントロールできる

RDFa / Microdata活用事例

- **検索エンジン**
  - SindiceはRDFa/Microdataデータを検索エンジン
  - Google, Yahoo, Bingは解析し、検索結果に反映している
- **Facebook** はRDFaを利用している(Open Graph API)
  - ウエブサイトからの情報抽出
  - Likeボタンを実装するため
- **Browser support**
  - FirefoxなどはRDFa/Microdataを検索できるようにプラグインが存在している
- RDFaとMicrodataの間の**変換**
  - http://rdf-translator.appspot.com
- **RDFa？Microdata**？どっちを使えばいい？
  - RDFaの方が複雑だが、より複雑な情報を表現できる
  - 現在はすべてのツールが両方サポートしているので、どちらでもいい！

SEOに効きます！

The SPARQL Query Language for RDF Data

A bit of Background

SPARQL Protocol And RDF Query Language

Hum, this is a recursive acronym 😓

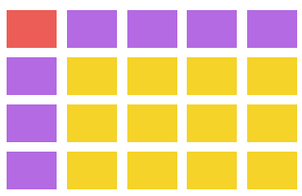
- SPARQL is a W3C standard
  - SPARQL 1.0 (15/1/2008)
  - SPARQL 1.1 (21/3/2013)

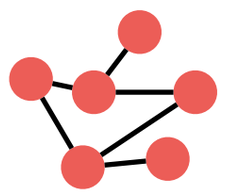
It appeared long after RDF itself (it was in 27/3/2000)

- It is supported by most RDF stores and frameworks

Comparison RDF vs. Relational

	Relational Database	RDF
Query language	SQL	**SPARQL**
Data topology	2D tables	**Graphs**
Database technology	Relational database	Triple store (RDF store, graph store)
Data model	Relational Model	RDF 1.1 Abstract Syntax





An Example of a SPARQL Query

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix bodic: <http://www.bodic.org/datasets/>

SELECT ?englishName
WHERE {
  GRAPH bodic:dataset1 {
    ?s rdfs:label “福岡"@ja ; rdfs:label ?englishName
  }
  FILTER ( lang(?englishName) == “en” )
}
```

An Example of a SPARQL Query (Structure)

Defines prefixes for Qnames (same as Turtle)

```
prefix rdfs: <http://www.w3.org/2000/0
prefix bodic: <http://www.bodic.org/da
```

Defines the type of query (SELECT) and the variables to output.

```
SELECT ?englishName
```

```
WHERE {
  GRAPH bodic:dataset1 {
    ?s rdfs:label “福岡"@ja ; rdfs:labe
  }
  FILTER ( lang(?englishName) == “en” )
}
```

Graph pattern. Defines conditions of the query.

Filter (optional) on variables.

An Example of a SPARQL Query (prefixes)

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix bodic: <http://www.bodic.org/datasets/>

SELECT ?englishName
WHERE {
   GRAPH bodic:dataset1 {
      ?s rdfs:label “福岡"@ja ; rdfs:label ?englishName
   }
   FILTER ( lang(?englishName) == “en” )
}
```

Definition / Use of prefixes

An Example of a SPARQL Query (Graph)

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix bodic: <h                                datasets/>

SELECT ?englishN
WHERE {
  GRAPH bodic:dataset1 {
      ?s rdfs:label “福岡"@ja ; rdfs:label ?englishName
  }
  FILTER ( lang(?englishName) == “en” )
}
```

Limits the scope of the query to a given graph (similar to a RDBMS table)

Variables (selection, match, filtering)

An Example of a SPARQL Query (Graph Pattern)

- Inner graph pattern: defines the conditions of the query (use the Turtle syntax)
- Equivalent to the following two triples
  - `?s rdfs:label “福岡”@ja`
  - `?s rdfs:label ?englishName`
- Explanation of the meaning of these triples later

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix bodic: <http://www.bodic.org/datasets/>
SELECT ?englishName
WHERE {
  GRAPH bodic:dataset1 {
    ?s rdfs:label “福岡"@ja ; rdfs:label ?englishName
  }
  FILTER ( lang(?englishName) == “en” )
}
```

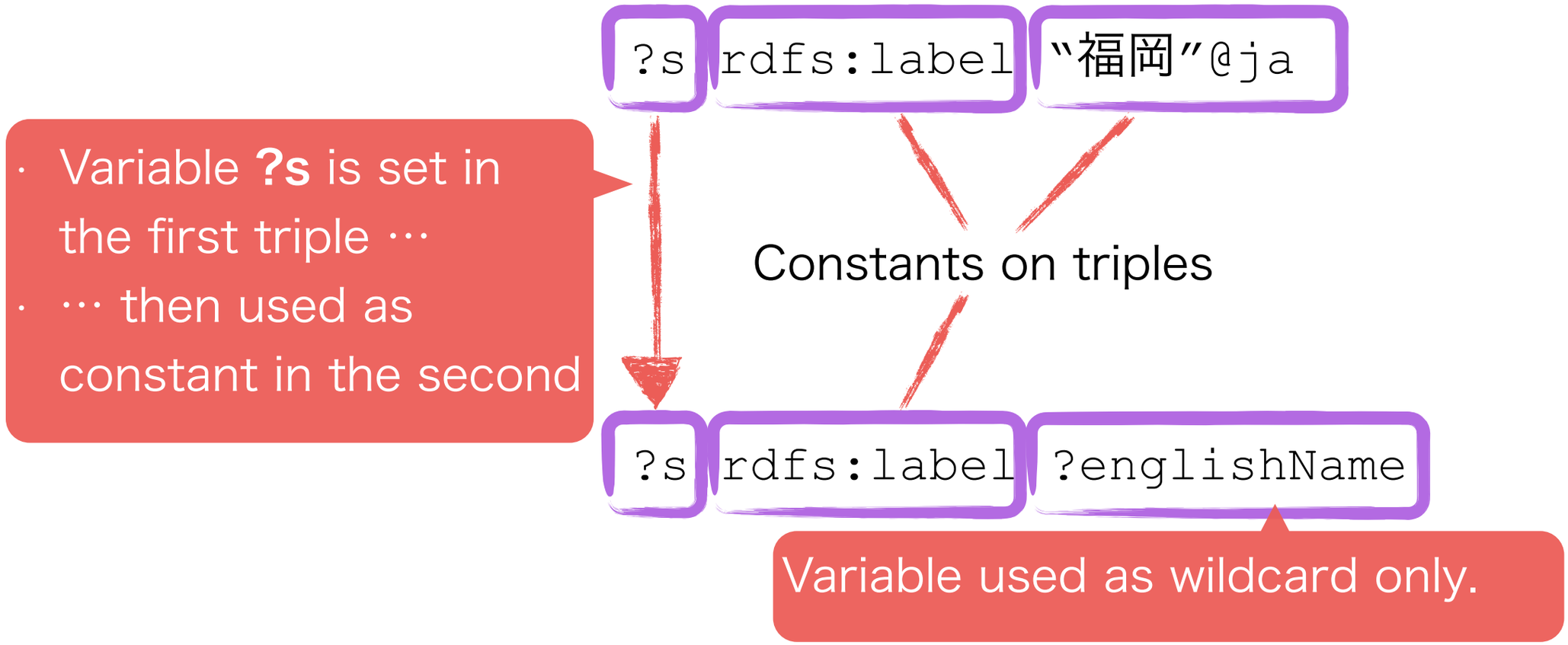
An Example of a SPARQL Query (Variables)

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix bodic: <http://www.bodic.org/datasets/>

SELECT ?englishName
WHERE {
   GRAPH bodic:dataset1 {
     ?s rdfs:label “福岡"@ja ; rdfs:label ?englishName
   }
   FILTER ( lang(?englishName) == “en” )
}
```

Variables
(selection, match, filtering)

An Example of a SPARQL Query (Graph Pattern)

- A graph pattern is a list of triples, Evaluated in order of apparition
- Constants are constraints on triples
- Variables act as both wildcard (when they first appear) and constraints (once they are set)



An Example of a SPARQL Query (Graph Pattern)

```
?s rdfs:label “福岡”@ja
```

Selects all the triples which predicate is rdfs:label and object is the string “福岡” with the language “ja”.
Stores the subjects of all the matching triples in ?s.

```
?s rdfs:label ?englishName
```

Selects all the triples which subject, stored in ?s, is as selected in the previous triple, and which object is rdfs:label.
Stores the objects of all the matching triples in ?englishName.

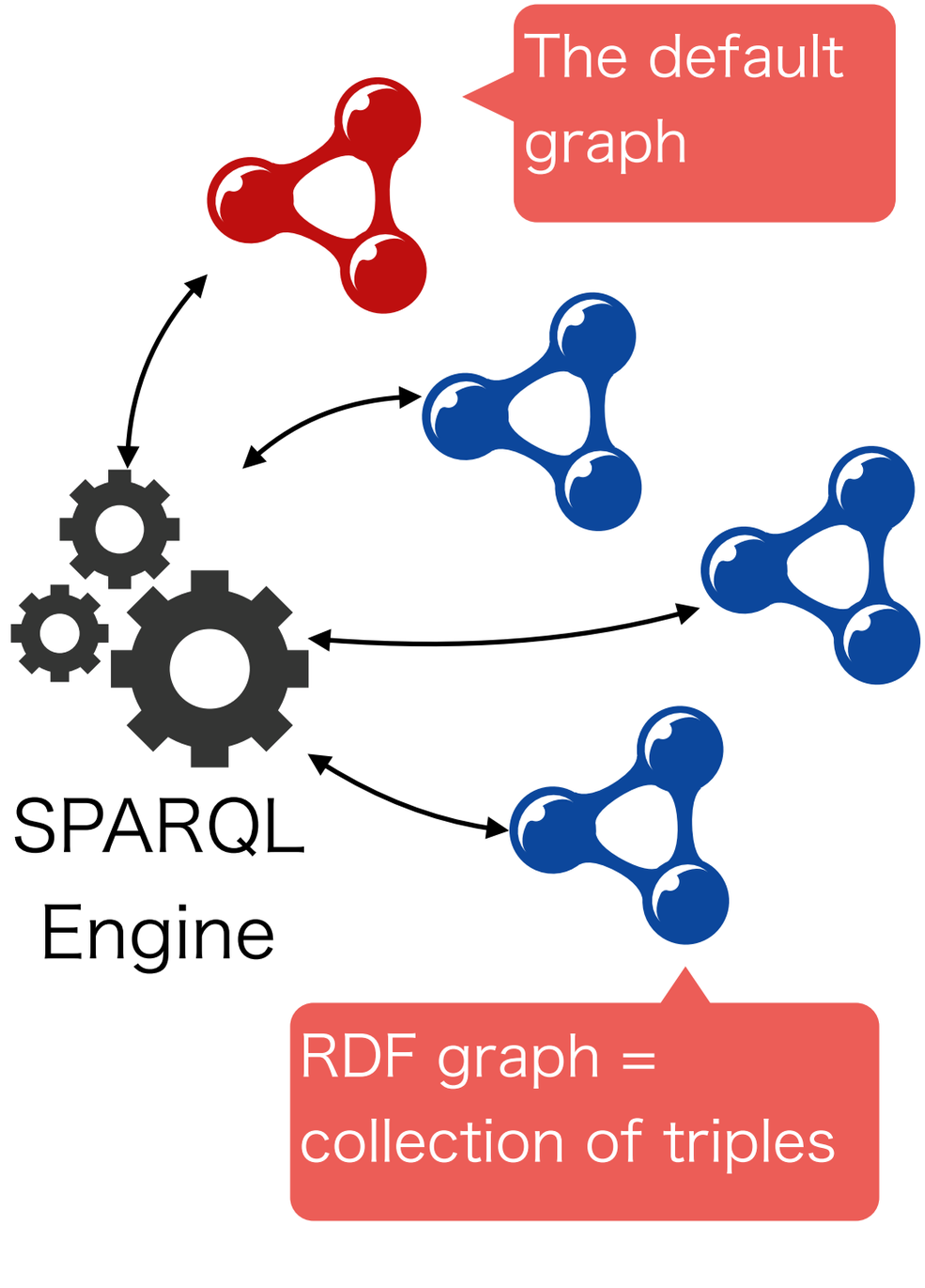
An Example of a SPARQL Query (Graph Pattern)

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix bodic: <http://www.bodic.org/datasets/>

SELECT ?englishName
WHERE {
  GRAPH bodic:dataset1 {
    ?s rdfs:label “福岡"@ja ; rdfs:label ?englishName

  }
  FILTER ( lang(?englishName) == “en” )
}
```

- Among all the objects stored in ?englishName, only keeps the ones which language is “en”
- “lang” is a special function defined in the SPARQL recommendation

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

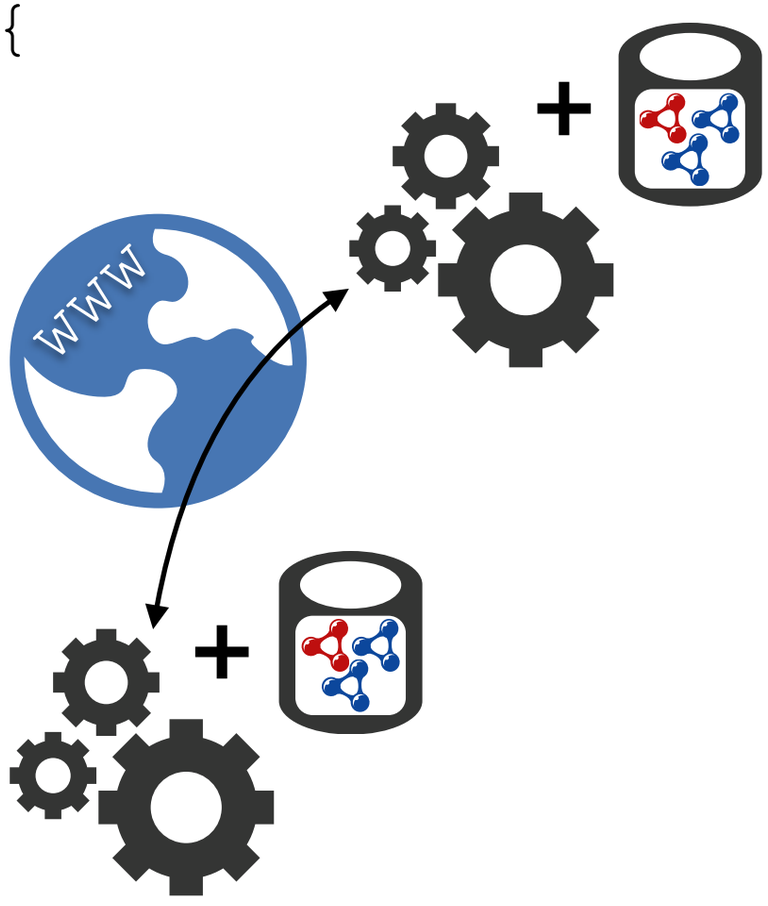


Federated Query with SERVICE

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
  }
}
```

Wait, SPARQL engines have URL ?

・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

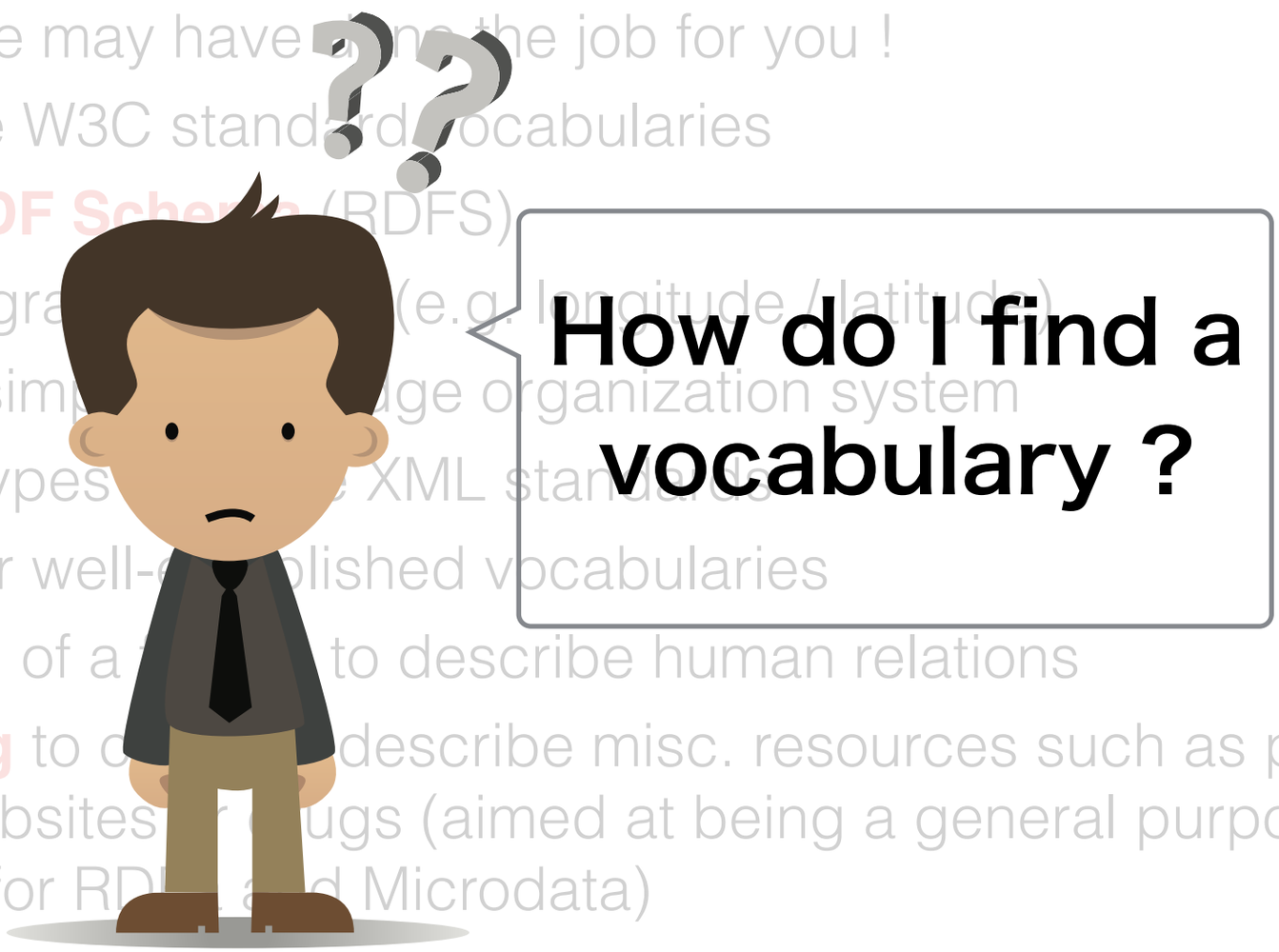
Vocabulary, RDF Schema and Ontology

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

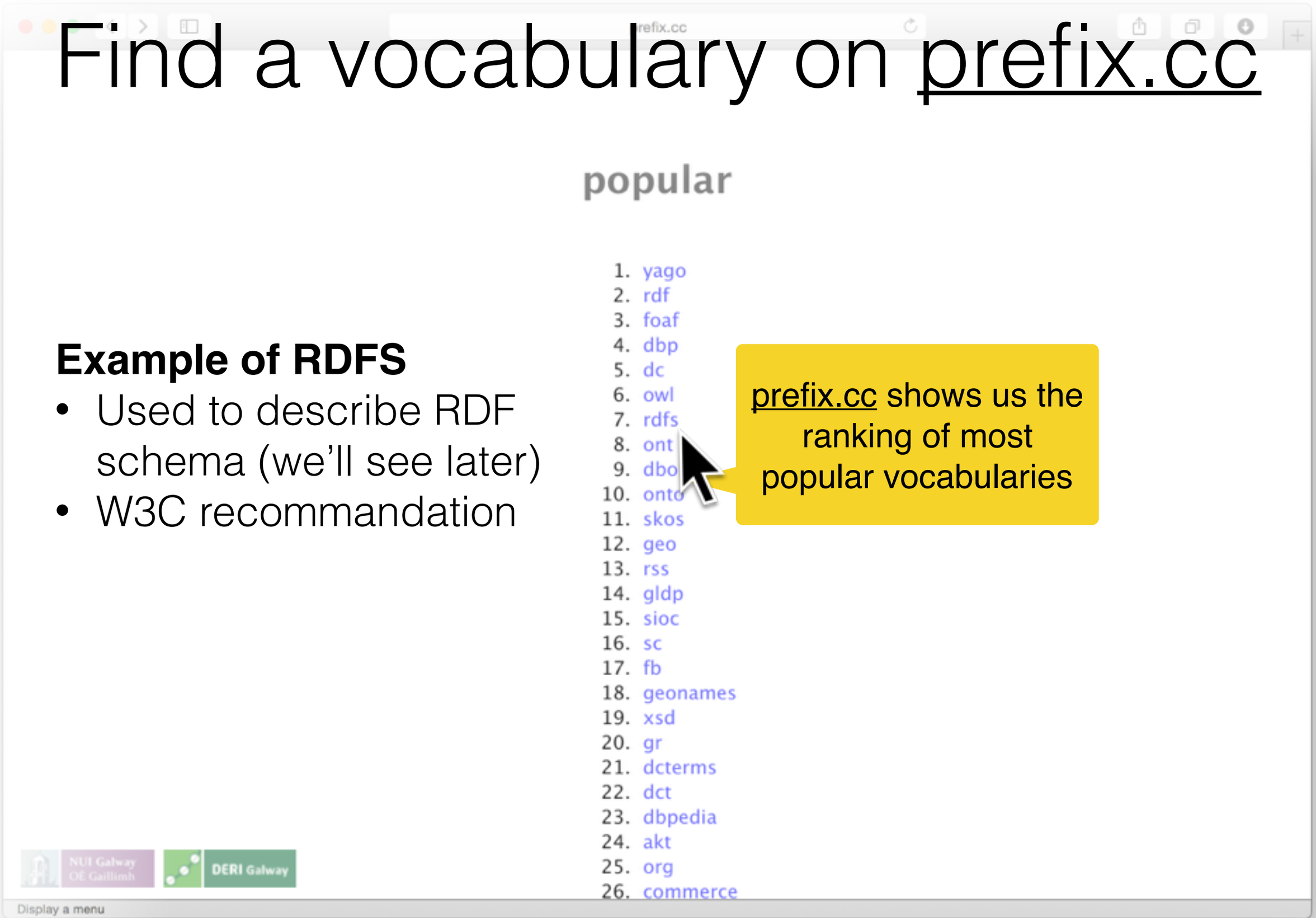
- Depending on the data you hold, you may need various vocabulary
  - You may create your own
  - But someone may have done the job for you !
- There are some W3C standard vocabularies
  - **RDF** and **RDF Schema** (RDFS)
  - **geo** for geographical (e.g. longitude /latitude)
  - **SKOS**, the simple knowledge organization system
  - **XSD**, data types for 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 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



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 .
```

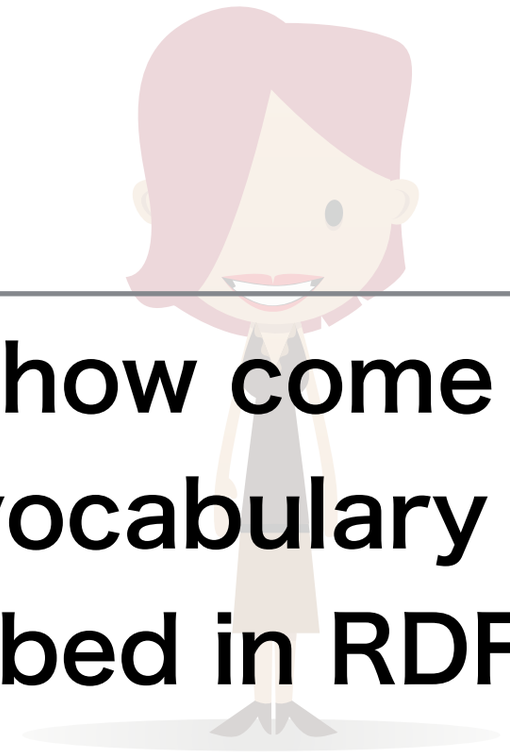
We get the turtle version of the vocabulary !

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                   hema#> ;
        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

Ontology: Schema for 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 recommendation)
  - **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**

Let's take a look at this one

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

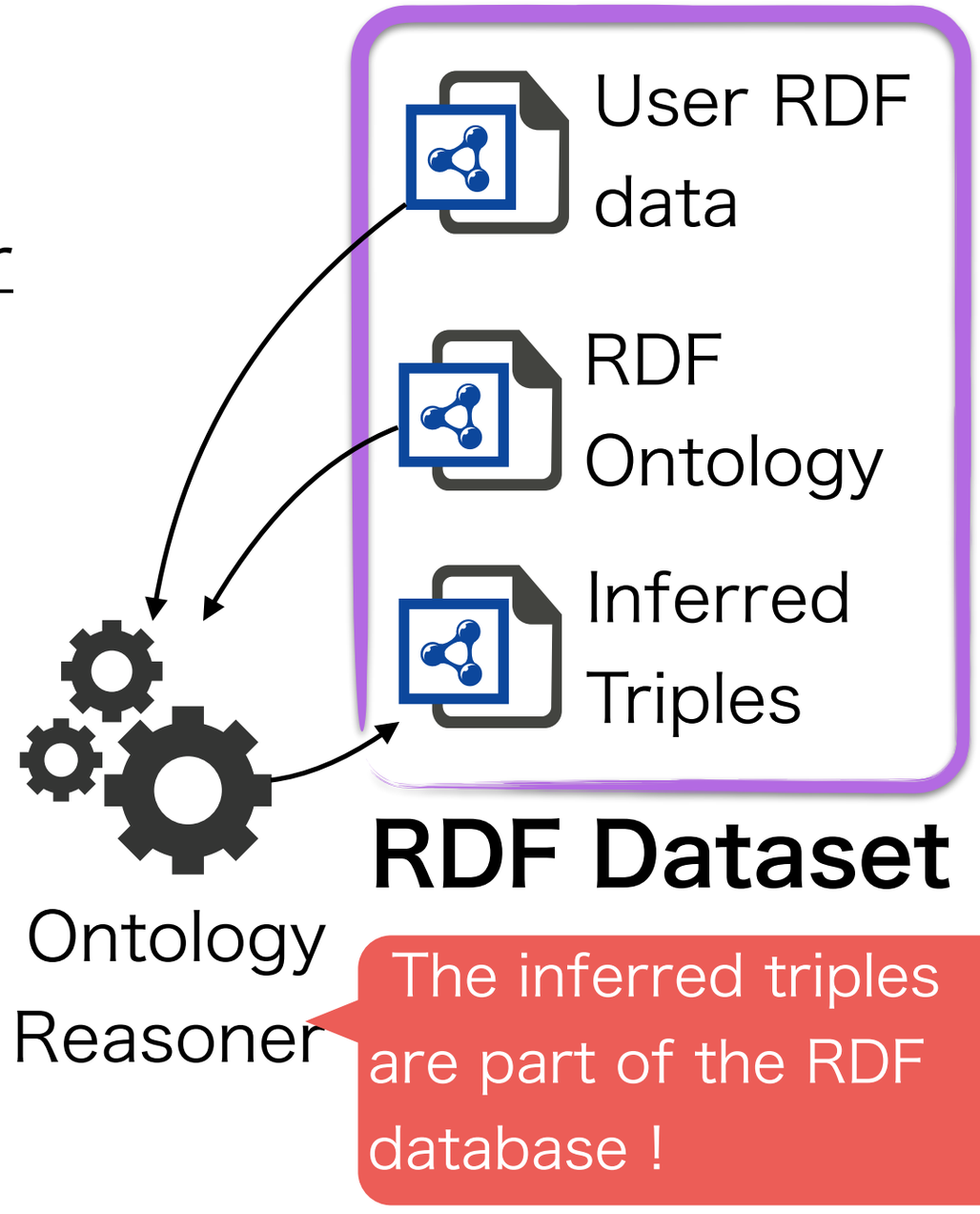
Basics of RDFS

- 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: Inference

- 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



The inferred triples are part of the RDF database !

Example of RDFS Inference

```
bodic:Vehicle a rdfs:Class.
bodic:Car a rdfs:Class ;
  rdfs:subClassOf bodic:Vehicle.
bodic:Plane a rdfs:Class ;
  rdfs:subClassOf bodic:Vehicle.
```

Schema part

```
data:myCar a bodic:car.
```

Data part

```
data:myCar a bodic:vehicle.
```

This triple is generated by a RDFS reasoner by inference from the two triples above.

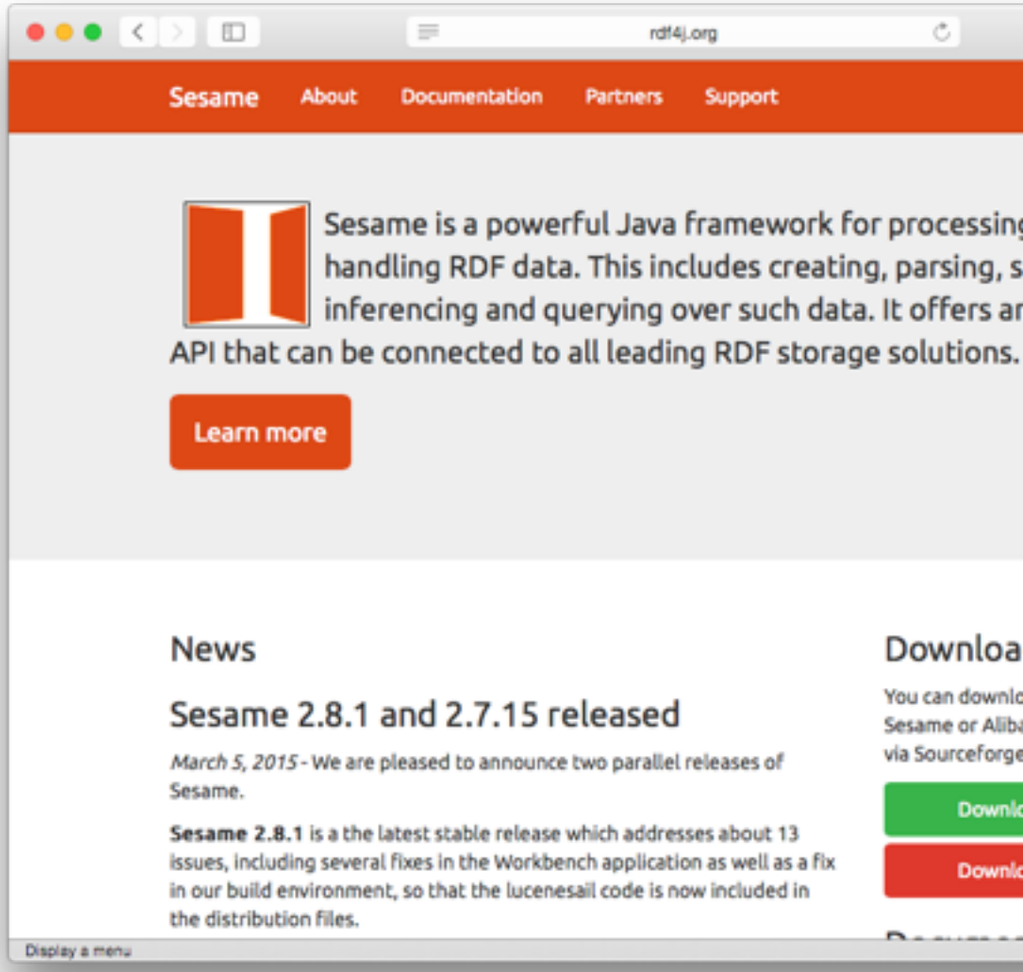
About Triple Stores

What is a triple store ?

- We know how to
  - Serialize RDF data with Turtle
  - Query RDF data with SPARQL
- But wait …
  - How do you make a SPARQL endpoint ?
  - A SPARQL should be very slow if it has to read multiple RDF files (e.g. Turtle / RDFa)
- **Triple store** are **database** that provide both
  - SPARQL endpoint
  - Quick access to RDF data

Sesame (rdf4j.org)

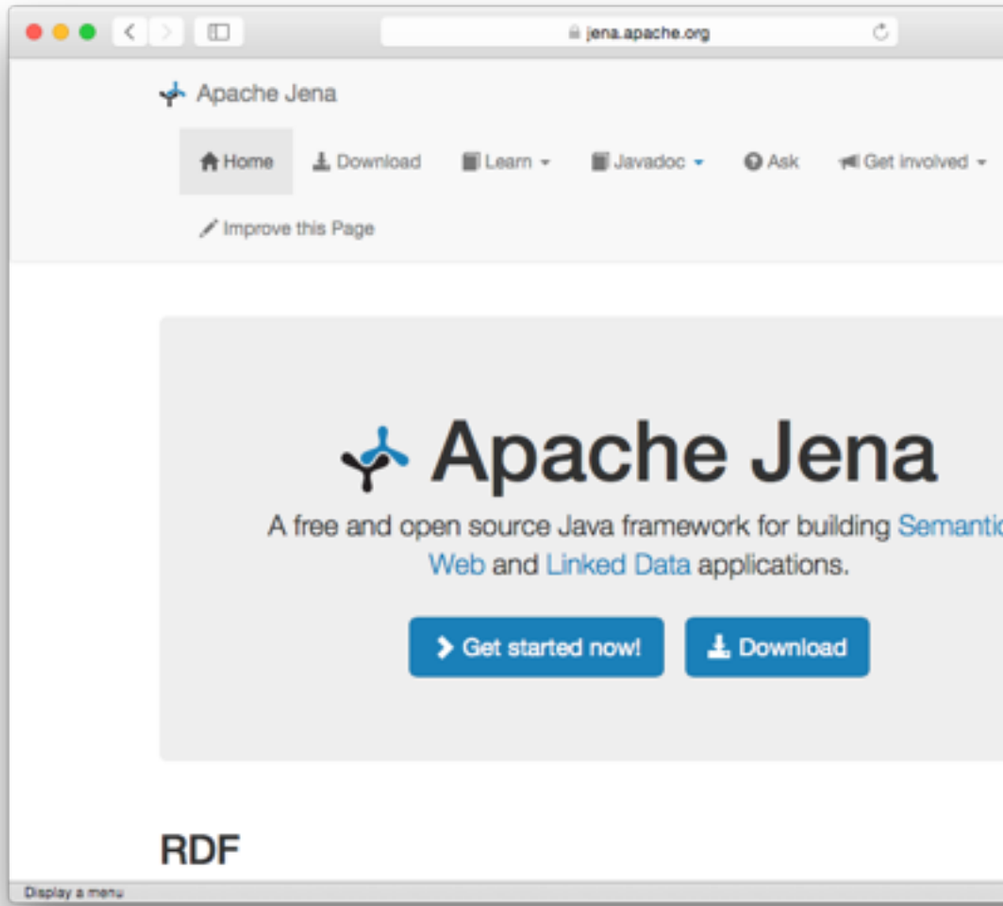
- Open-source, written in Java
- Supports plugins
- Several functionalities
  - Java RDF framework to programmatically work with RDF data
  - Triple Store Server (Java weblet for servers such as Tomcat or Jetty)
  - Inference in RDFS (not OWL)
- Originally developed as a research project
  - European 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++)

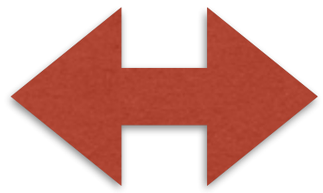
Virtuoso (virtuoso.openlinksw.com)

- 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

Weaknesses of RDF

Wait, Isn't there a contradiction ?

Semantic Web
Distributed data



Triple Store
Centralized data

- There is not no contradiction, but let's face it, you need a database for high query performance
- Yet, SPARQL endpoint can collaborate (federated queries)
  - But those are slow, and often turned on by publicly available endpoints

Is The RDF Toolchain too Disruptive ?

- In order to make your website RDF-ready you typically need
  - A triple store
  - A SPARQL engine
  - Some RDF libraries (client and serve side)
  - An ontology reasoner
- This is a lot ! And most people are not familiar with these technologies
- RDF libraries are often buggy and slow
- Moreover performance are often poor
  - Triple store are often slower that RDBMS or other NoSQL (e.g. MongoDB) counterparts
  - Ontology reasoners are very slow to execute
  - RDF text formats, even Turtle, are very verbose

Not a good fir for large tabular data

Is RDF 1.1 a Good Data Model ?

- RDF is very simple and often qualified as “elegant”
- Yet it has some weaknesses:
  - it lacks native basic data structure such as sorted lists and sets
  - it is very verbose by nature
  - it relies heavily on “blank nodes”, often used inconsistently
  - the notion of “graph” often seem as an afterthought
  - W3C recommendations are **very** hard to read (it does not have to be this way)
- JSON-LD tries to address these issues
  - This a (new) W3C recommendation too http://www.w3.org/TR/json-ld/#basic-concepts
  - The JSON-LD toolchain is much simpler too

RDF 1.1 added support through the RDF vocabulary and some syntactic sugar

it feels like they have been added for SPARQL

Some Articles on RDF Pro/Cons

- About the weaknesses of RDF (from on major designer of JSON-LD)
  - http://manu.sporny.org/2014/json-ld-origins-2/
- Successful integration of RDF
  - https://www.ibm.com/developerworks/community/blogs/c06ef551-0127-483d-a104-cdd02b1cee31/entry/february_3_2014_1_47_pm?lang=en

Conclusion

I want to Publish my Data.

Where do I start ?

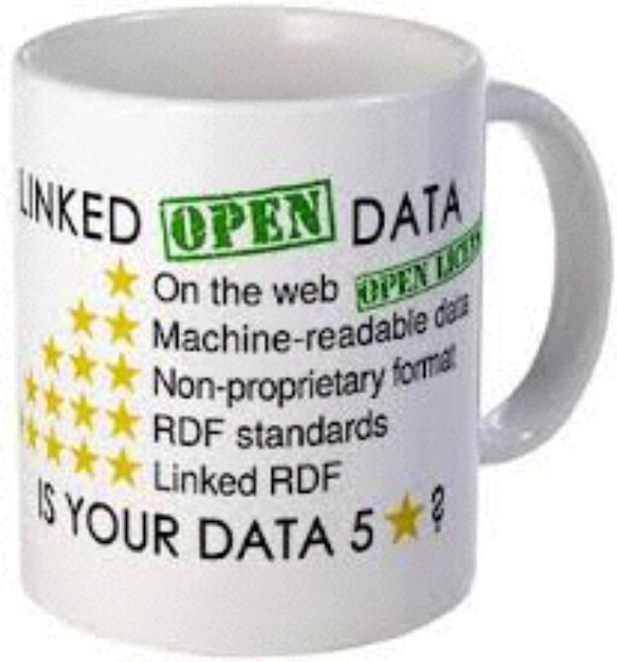
It is enough to upload your file to the Internet with a link on your Web page !

With an open license

.. yet you can choose to be kind to data consumers

The Levels of Open Data

- **1 star:** put on the web with an open license
- **2 stars:** use a machine-readable, structured format
  - **CSV** or **Excel**, not HTML or PDF
- **3 stars:** use free format
  - **CSV** or **OpenDocument** (ODF), not Excel
- **4 stars:** use RDF, or any compatible W3C recommended format
  - May not be relevant for all kinds of data
  - Recommended for meta-data like information (in this case I would recommend to embed triples in HTML pages with **RDFa** or **MicroData**)
  - You don't have to do it yourself !
- **5 stars:** link your data with other sources
  - Best for RDF-first data. It requires a lot of effort to convert legacy data to linked RDF



Useful Tools and Services

- **CKAN** Data Catalog
  - A CMS to organize and publish data to the Internet, with an open license
  - Usually self-hosted
- **BODIK's CKAN**
  - BODIK is to provide with CKAN hosting service, as well as consulting services to use it
- **BODIC.org**
  - We are proposing a service to publish easily your data as 4-star open data
  - It works hand-to-hand with CKAN data catalogs
  - It makes 3-star open data accessible via HTTP API, using W3C recommended technologies

www.bodic.org

BODIC.org　サービス詳細　データ登録　データ検索・可視化　API説明　ログイン

ビッグデータ＆オープンデータ
蓄積・分析・公開・活用プラットフォーム

BODIC.org: Big Data & Open Data in the Cloud

データ登録

あなたのデータをアップロードし,僅かの設定だけでにHTTP API公開で公開できる.

データ公開

HTTP REST API経由でテータを公開し,他のデータセットとのマッシュアップもできる.

視覚化

BODIC.org上にあるデータを可視化して検索できる.

ありがとうございました