初めてのSPARQL

Trouvé Antoine　（トルヴェ　アントワン）

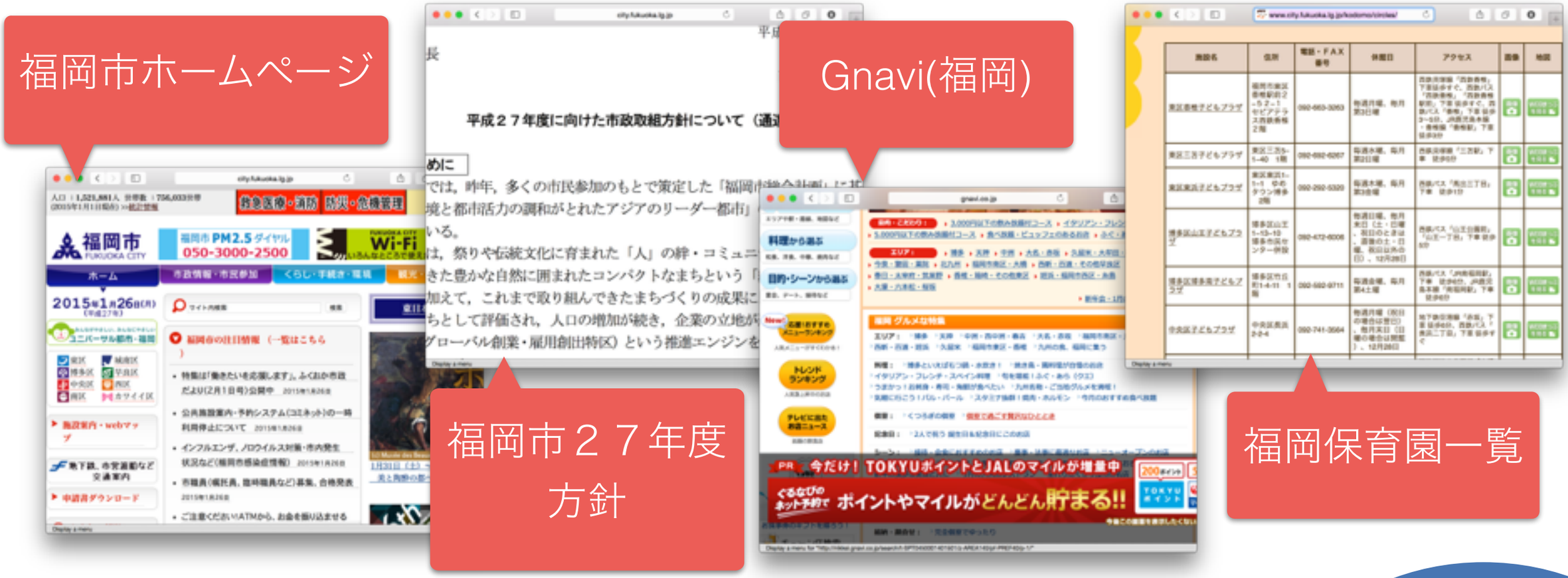
2015/08/27

九州先端科学技術研究所・九州大学

- **セミナーの目的**
  - SPARQLで基本的なクエリーを書けるようになること
  - SPARQLの背景について理解すること
- **セミナーの概要**
  - セマンティックウエブについて（約１５分）
  - RDFデータモデルについて（約３０分）
  - SPARQL演習（約２時間）

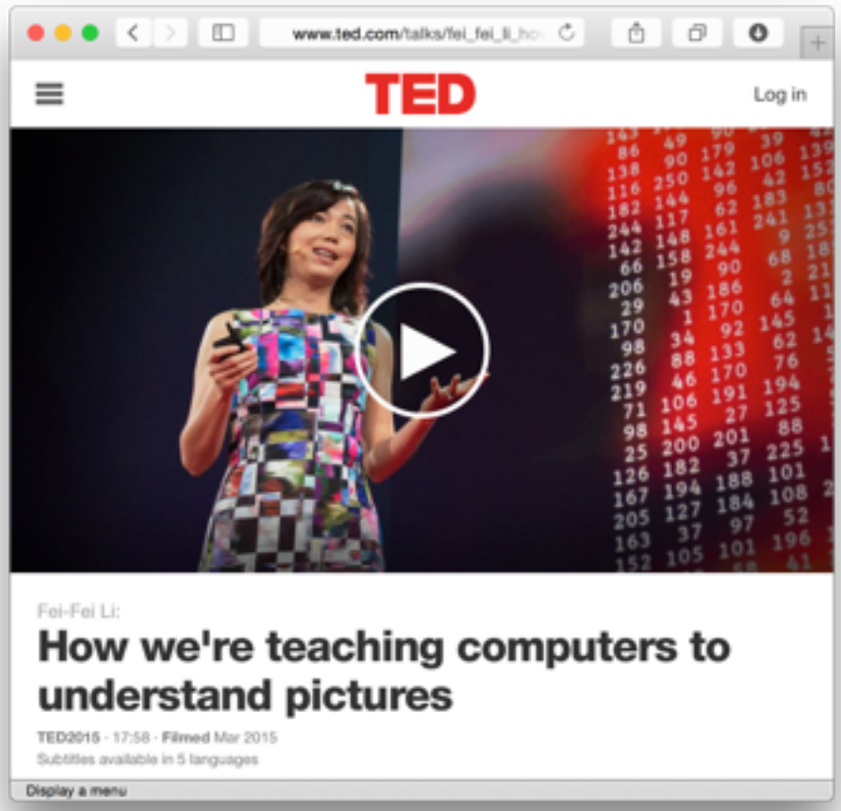
The RDF Data Model

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？



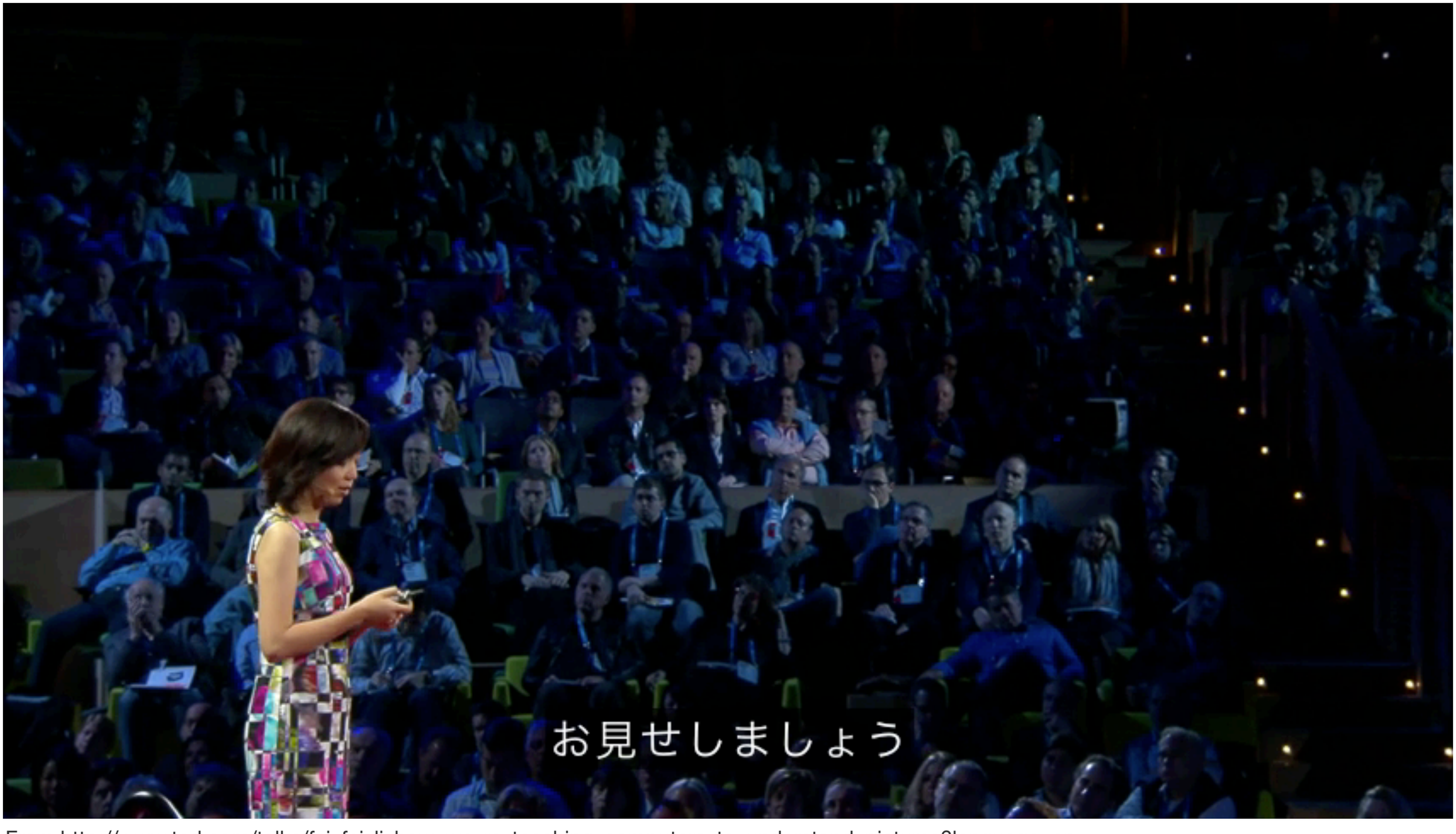
解決策１

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

解決策１について：先端研究の現状

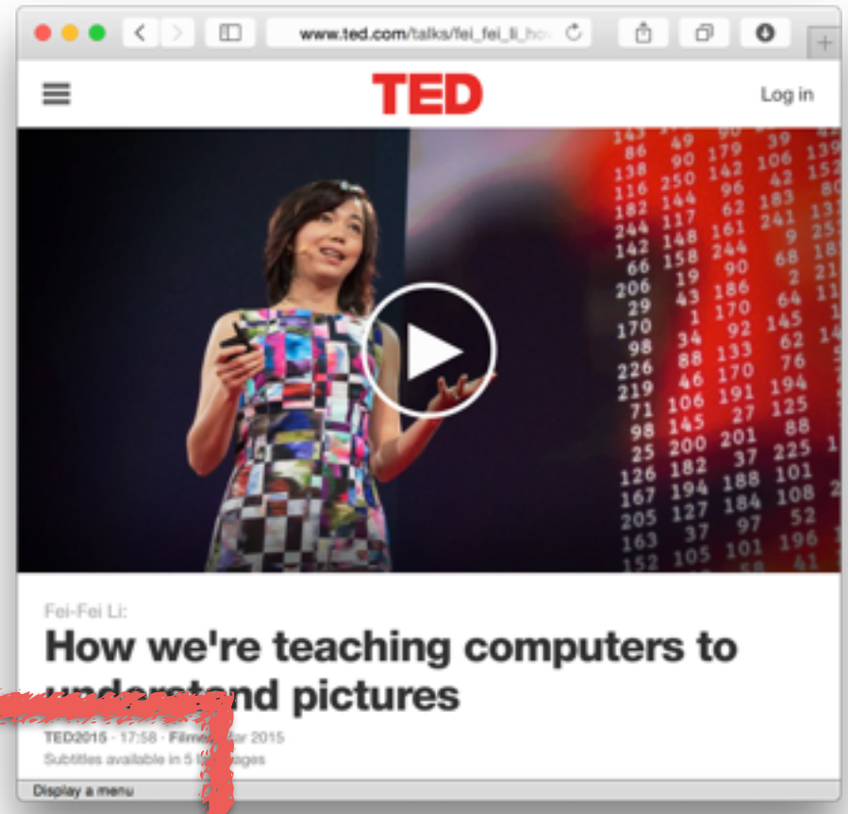


From http://www.ted.com/talks/fei_fei_li_how_we_re_teaching_computers_to_understand_pictures?language=en
Fei-Fei Li
Stanford’s Artificial Intelligence Lab and Vision Lab

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)

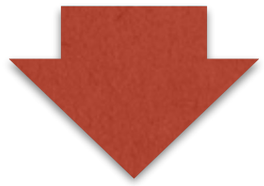
QnameとCURIE: IRIが読み

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	15-2-1”	xsd:date
http://city.fukuoka.lg.jp			en
http://city.fukuoka.lg.jp			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	
	ec	“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

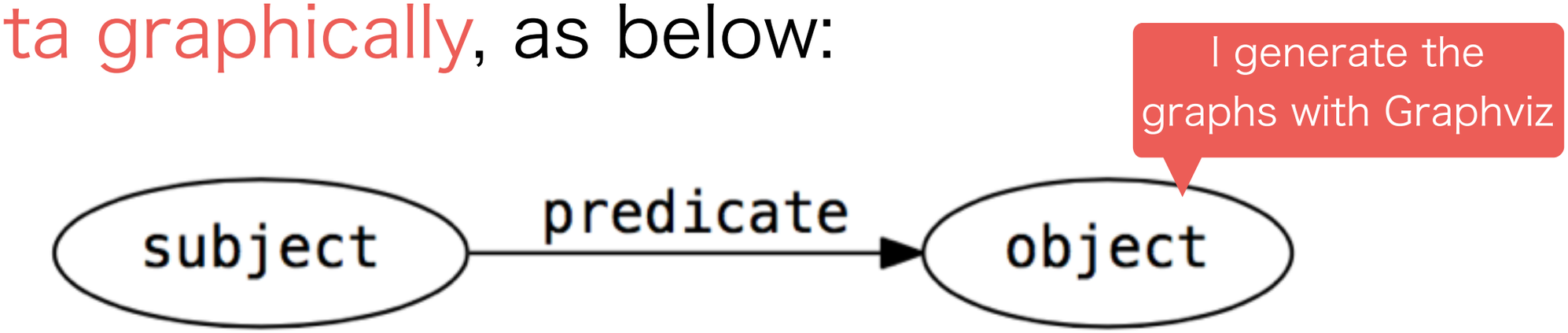
言語の例

・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

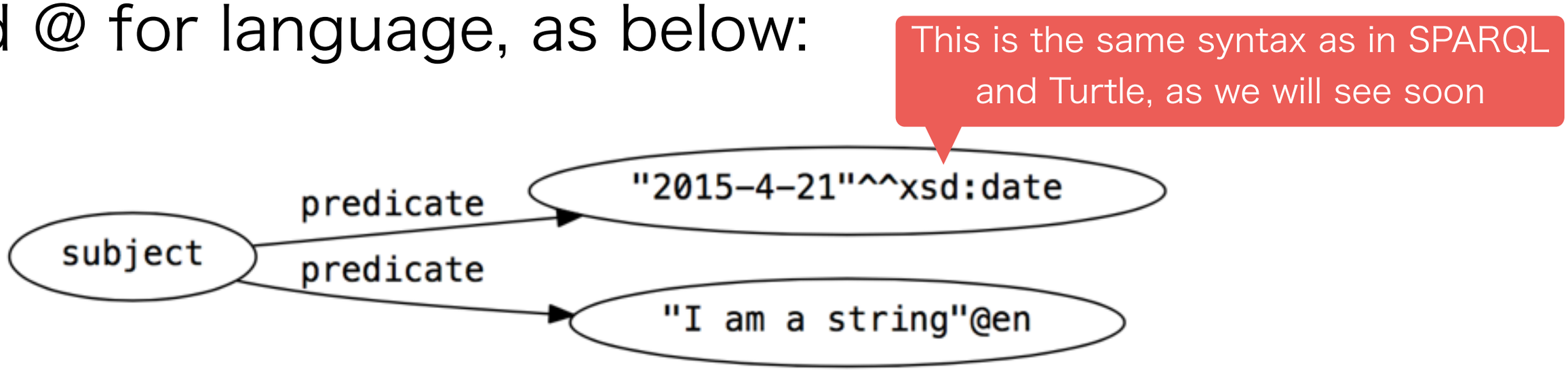
- RDF規格はXML標準タイプを含む:
  - xsd:integer, xsd:decimal, xsd:float, xsd:double, xsd:string, xsd:boolean, xsd:dateTime, xsd:date, xsd:time

RDFグラフ

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

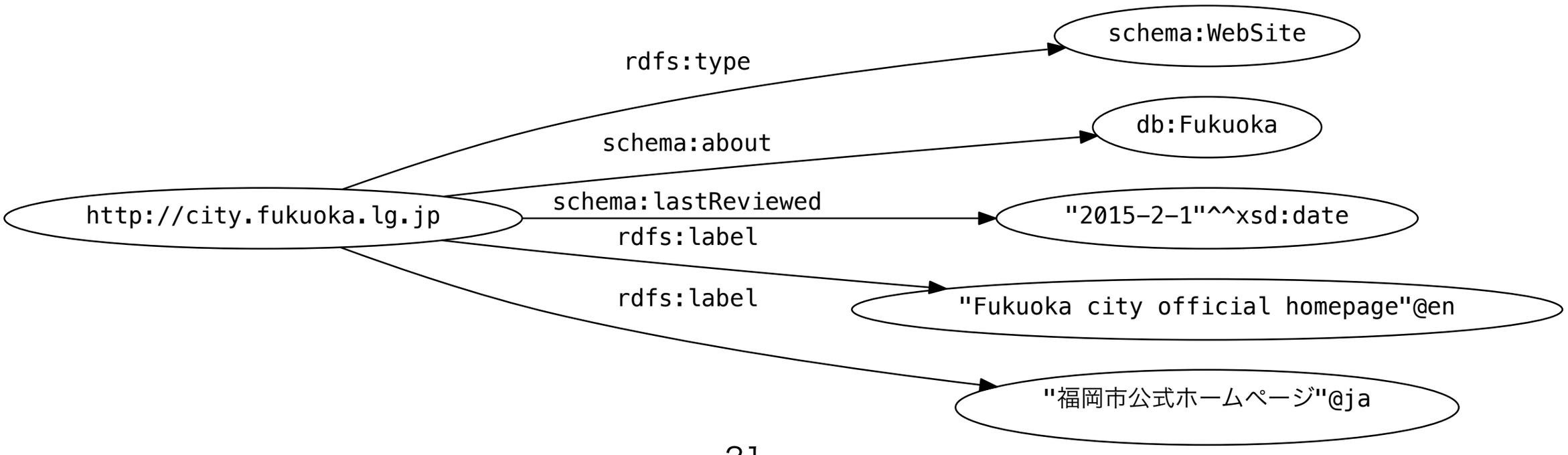


Literals' tags are represented with "^^" for datatypes, and @ for language, as below:



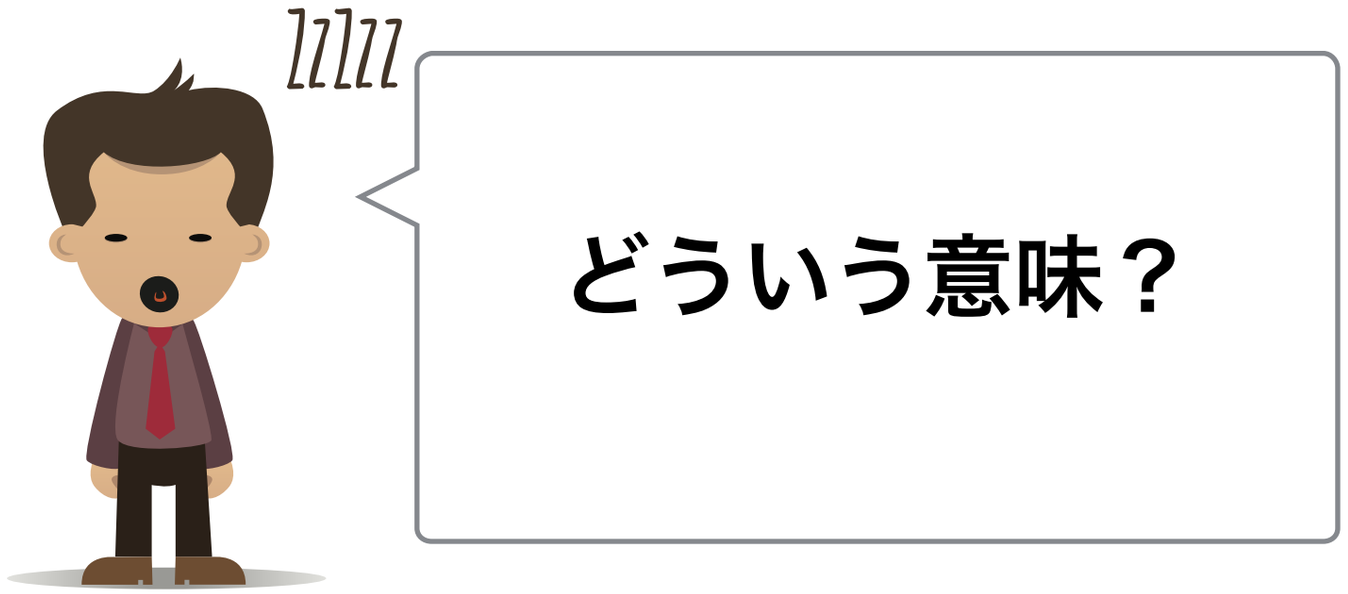
An Example of Graph

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



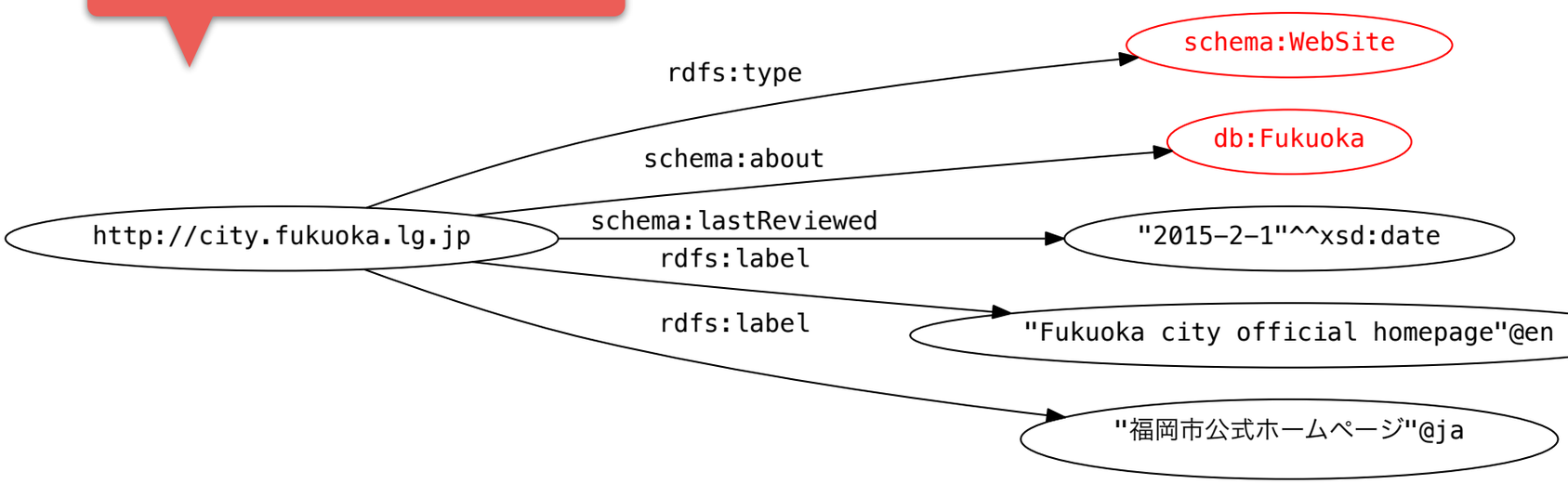
LODについて

- 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

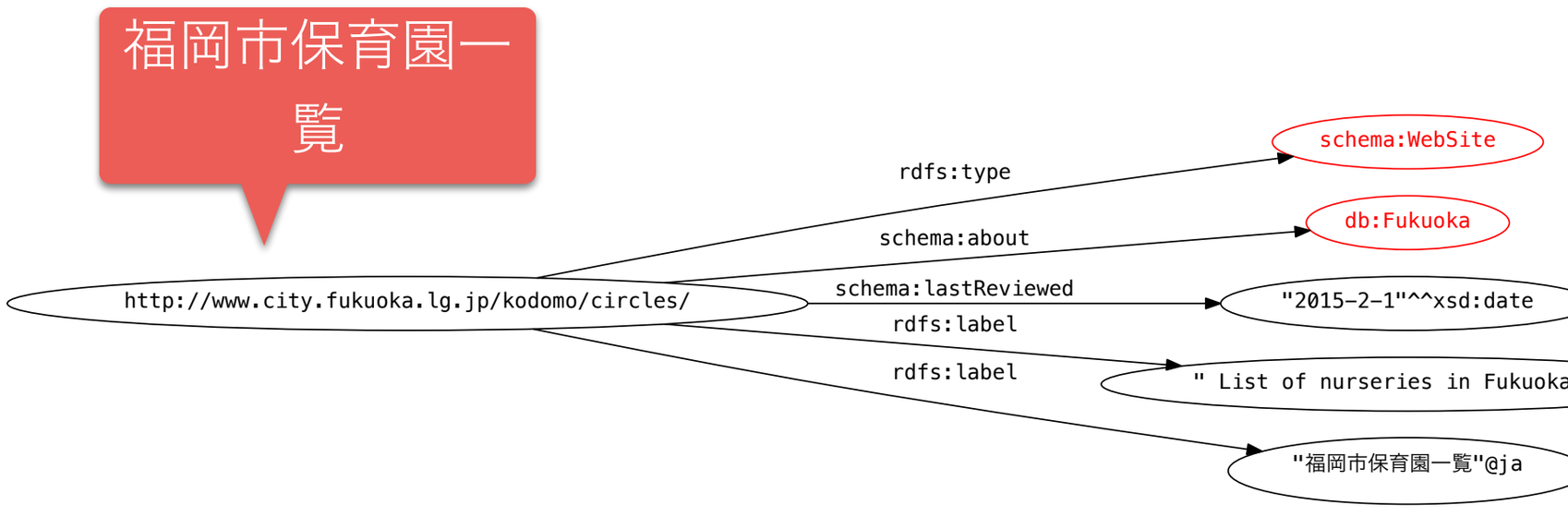


LODグラフの例（1）

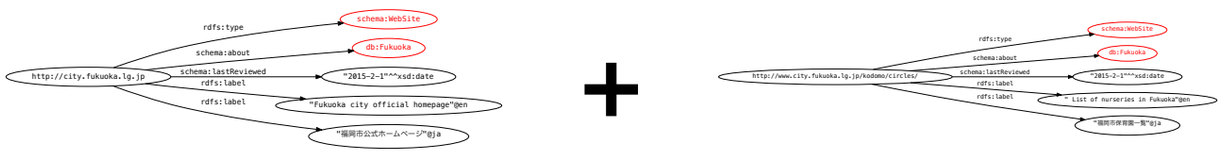
福岡市ホームページ

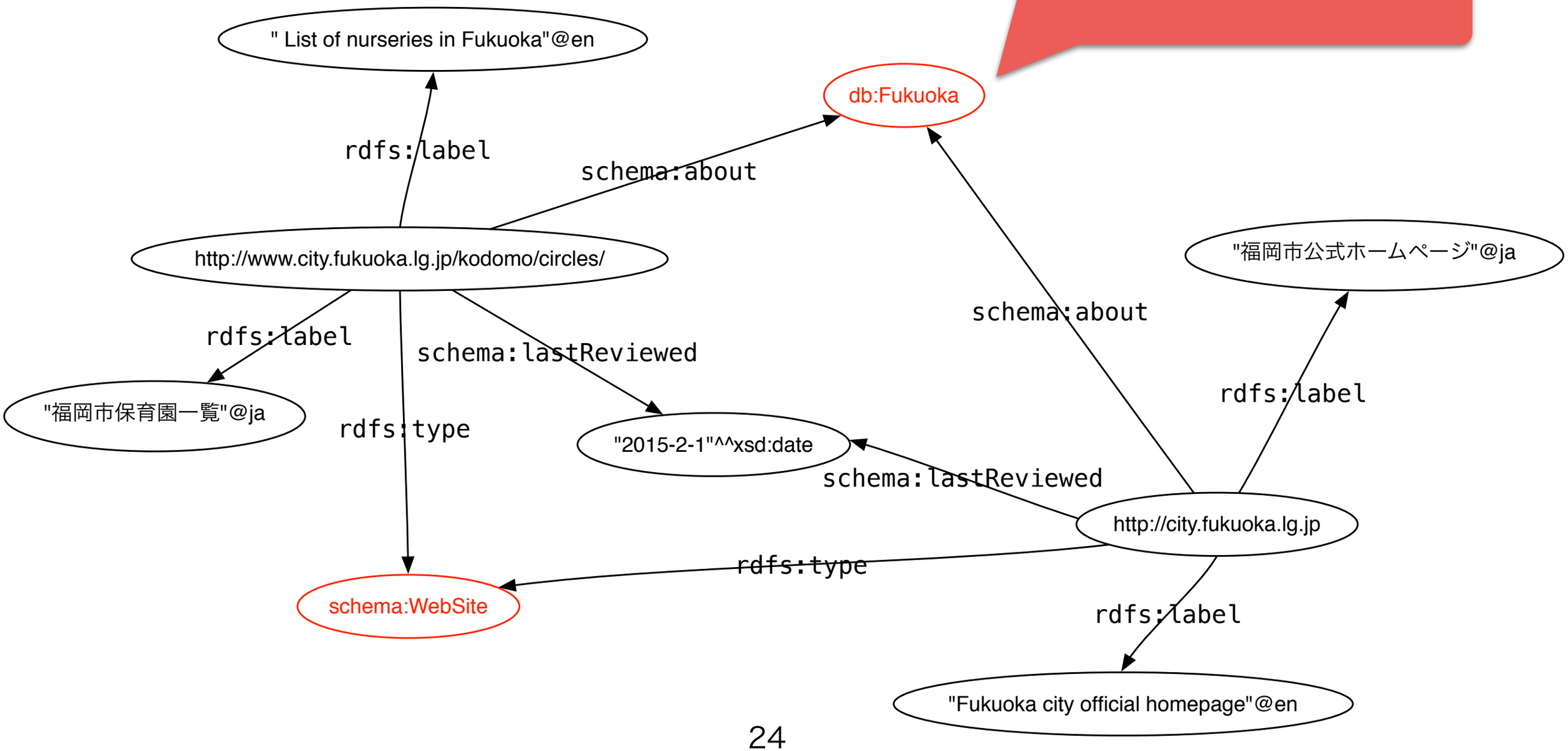


福岡市保育園一覧

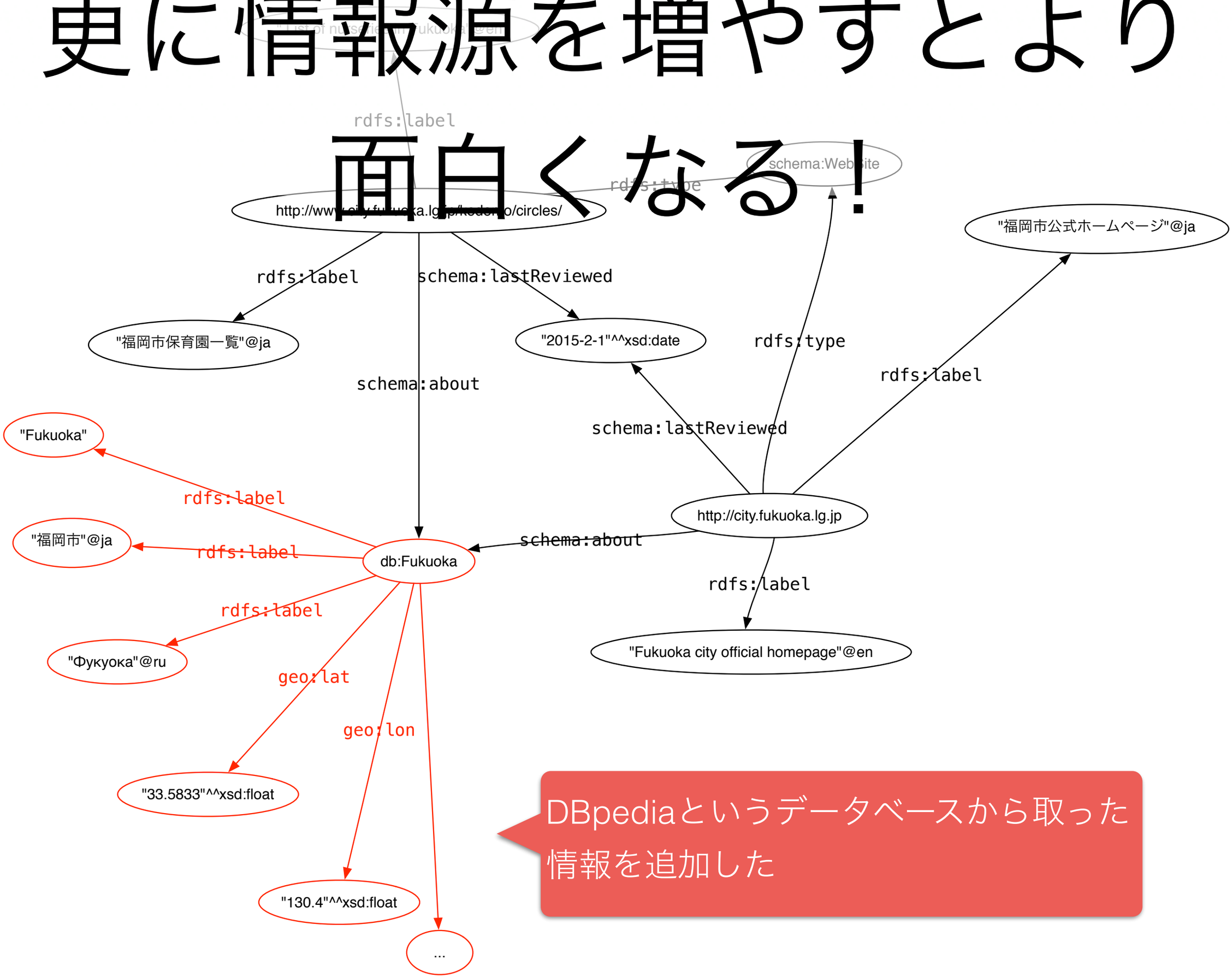


LODグラフの例（1）


rdfs:type
schema:about
schema:lastReviewed
rdfs:label
rdfs:label
schema:WebSite
db:Fukuoka
"2015-2-1"^^xsd:date
"Fukuoka city official homepage"@en
"福岡市公式ホームページ"@ja
http://city.fukuoka.lg.jp
+
rdfs:type
schema:about
schema:lastReviewed
rdfs:label
rdfs:label
schema:WebSite
db:Fukuoka
"2015-2-1"^^xsd:date
" List of nurseries in Fukuoka"@en
"福岡市保育園一覧"@ja
http://www.city.fukuoka.lg.jp/kodomo/circles/
=

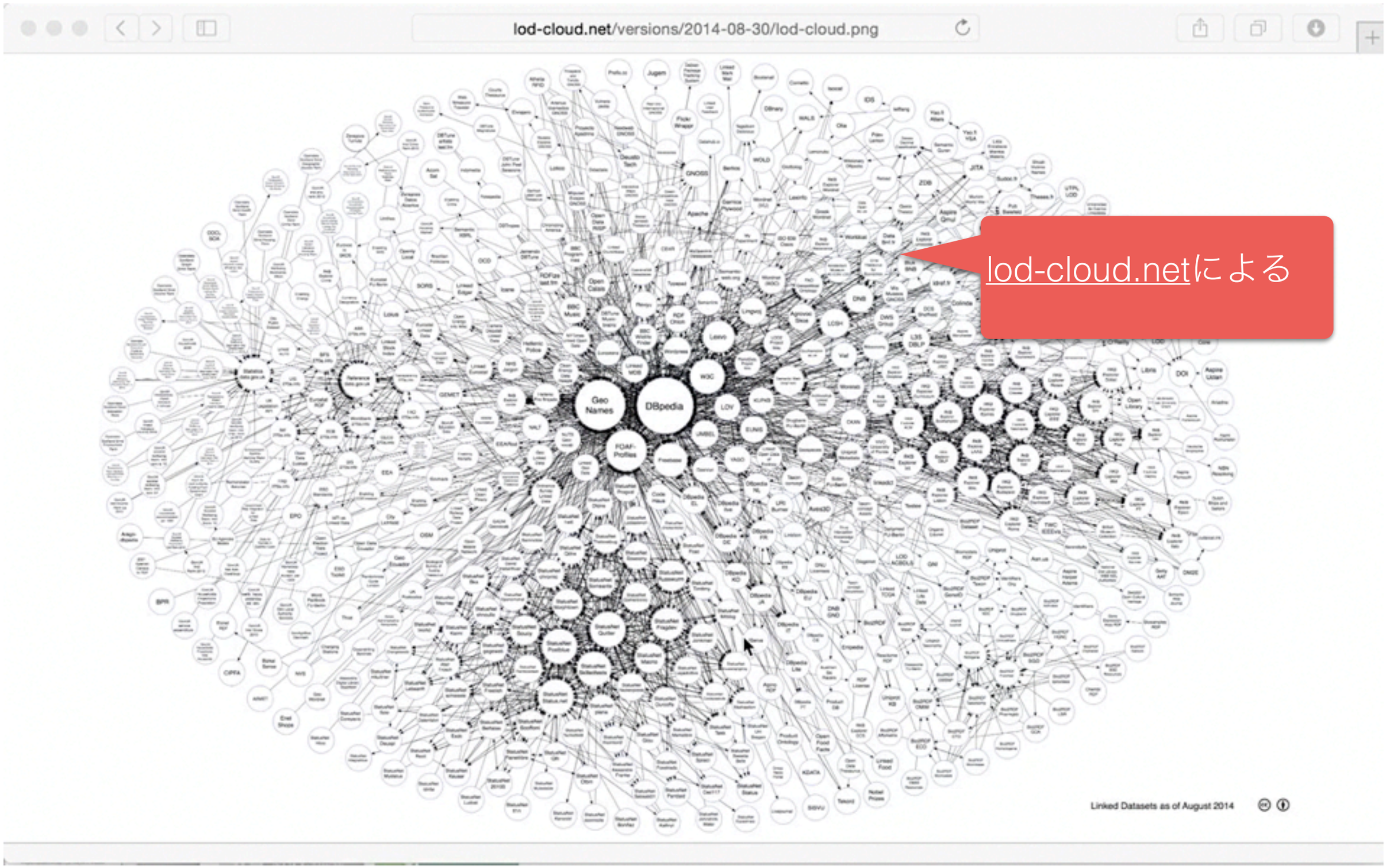

両方福岡市について
" List of nurseries in Fukuoka"@en
rdfs:label
schema:about
db:Fukuoka
http://www.city.fukuoka.lg.jp/kodomo/circles/
"福岡市公式ホームページ"@ja
schema:about
rdfs:label
rdfs:label
schema:lastReviewed
"福岡市保育園一覧"@ja
rdfs:type
"2015-2-1"^^xsd:date
schema:lastReviewed
http://city.fukuoka.lg.jp
rdfs:type
rdfs:label
schema:WebSite
"Fukuoka city official homepage"@en

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

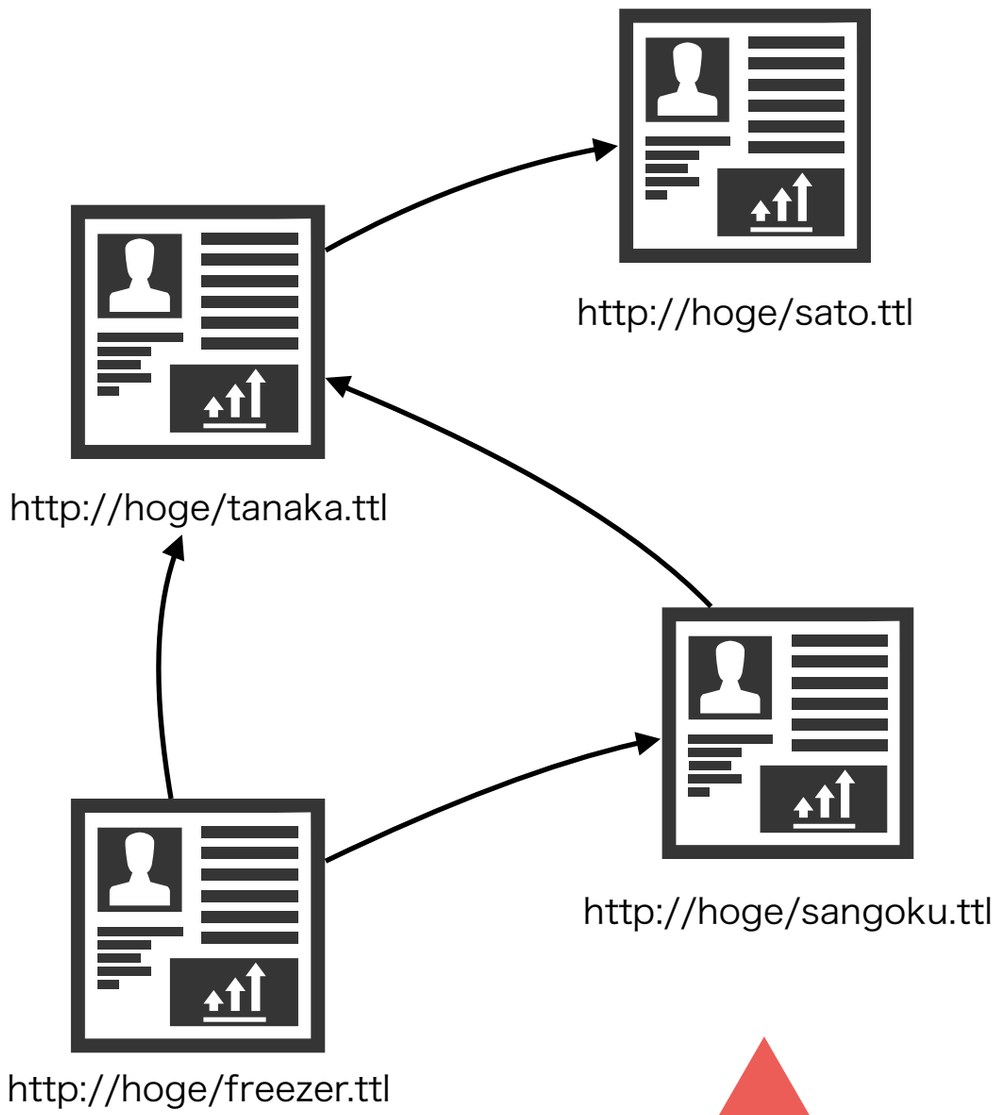

lod-cloud.net/versions/2014-08-30/lod-cloud.png
lod-cloud.netによる
Geo Names
DBpedia
FOAF-Profiles
Linked Datasets as of August 2014

RDFデータ

保管について

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**という
  - NoSQLデータベースの１種類

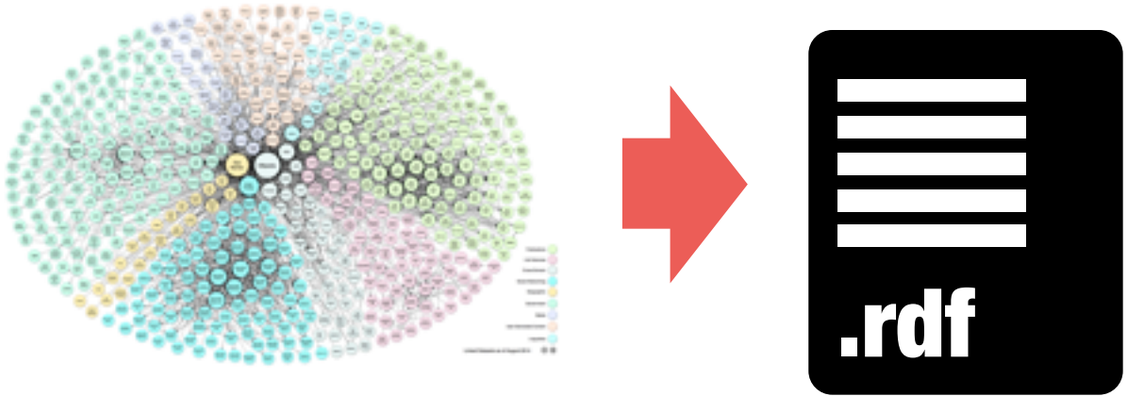


Friend of Friend (FOAF)プロジェクトについて

・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！

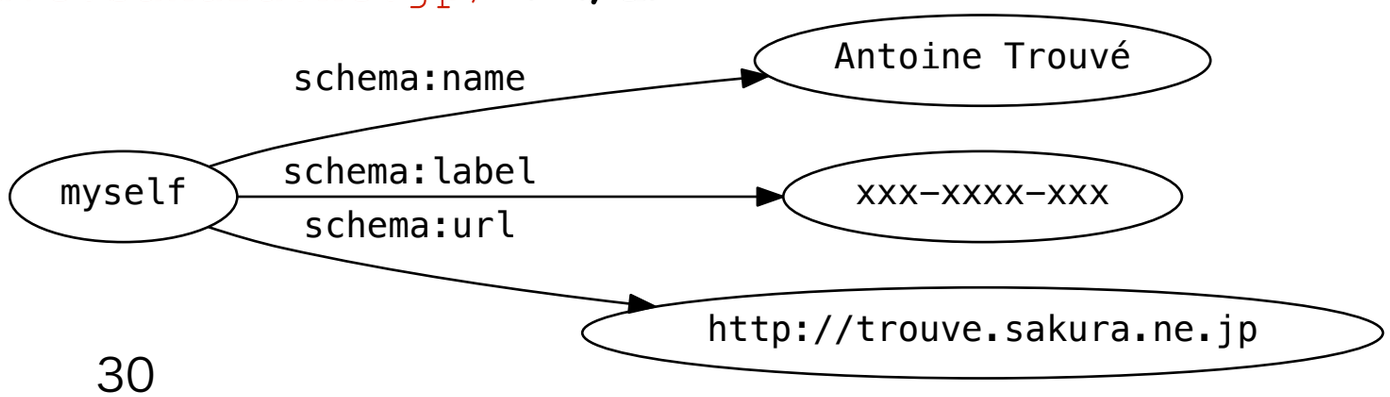
Microdata と RDFa

- 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：

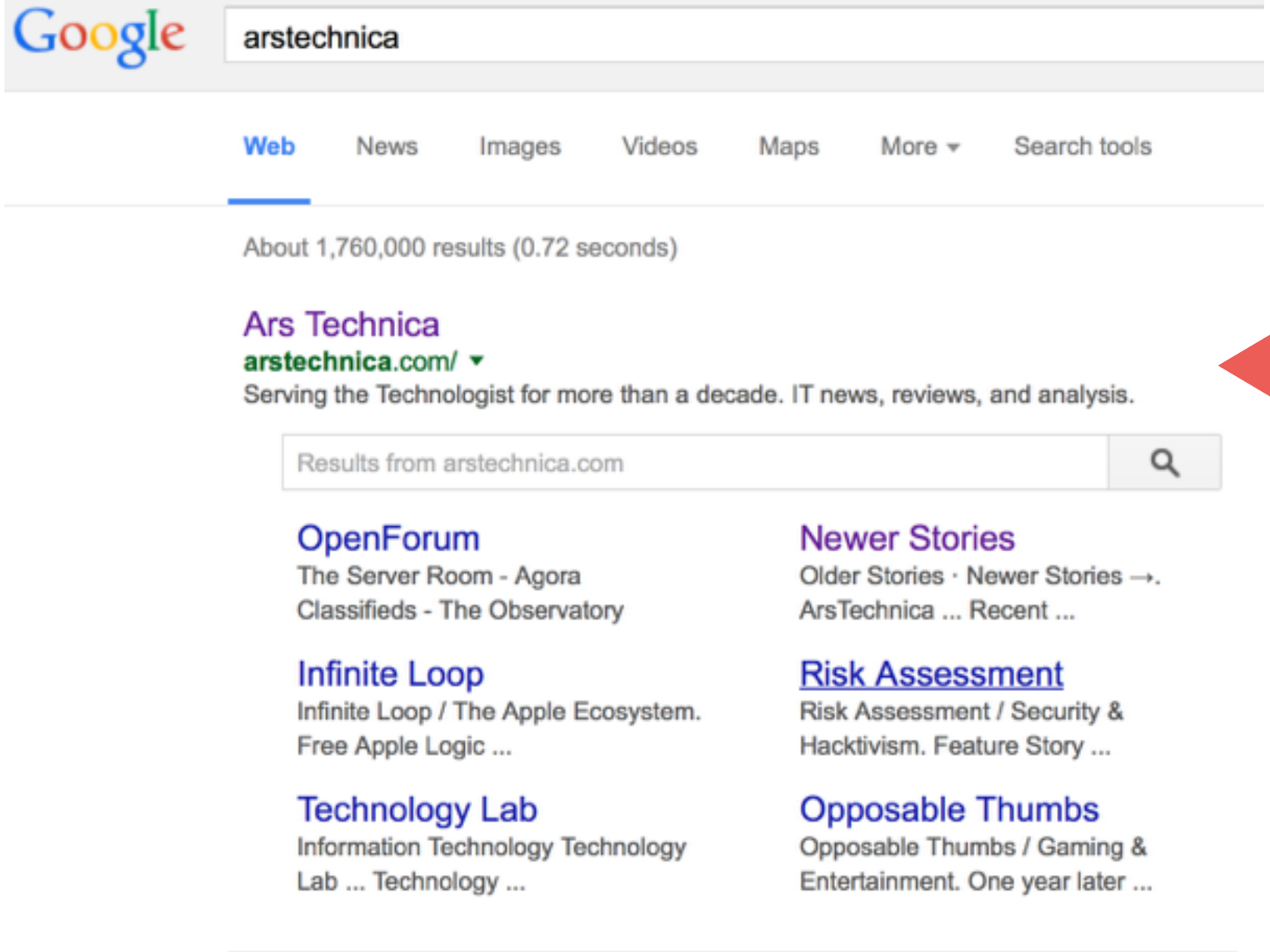
```
<p about="myself" vocab="http://schema.org/" typeof="Person">
  My name is
  <span property="name">Antoine Trouvé</span>,
  my phone number is
  <span property="telephone">xxx-xxxx-xxx</span>
  and my homepage is
  <a property="url" href="http://trouve.sakura.ne.jp/"></a>
</p>
```

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！

ハンズオン：SPARQL クエリー書きましょう！

SPARQLにおける背景

SPARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage

- SPARQLはW3C規格である
  - SPARQL 1.0 (15/1/2008)
  - SPARQL 1.1 (21/3/2013)
- 現在のRDFデータベースは大体対応している

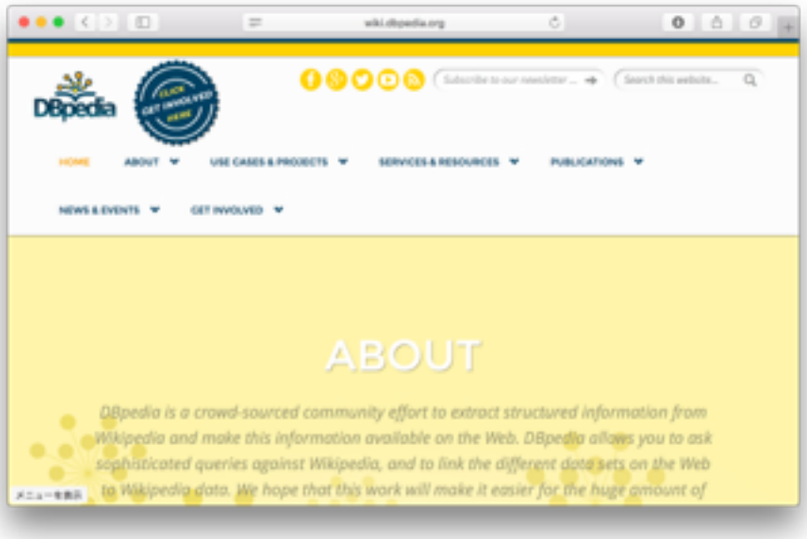
RDFは2000年３月に
発表されたので、
SPARQLの方が大分若い

SPARQLはRDFのSQLである

	関係データベース	RDF
クエリー言語	SQL	SPARQL
データベース種	関係データベース	RDFデータベース
データモデル	関係モデル	RDF 1.1

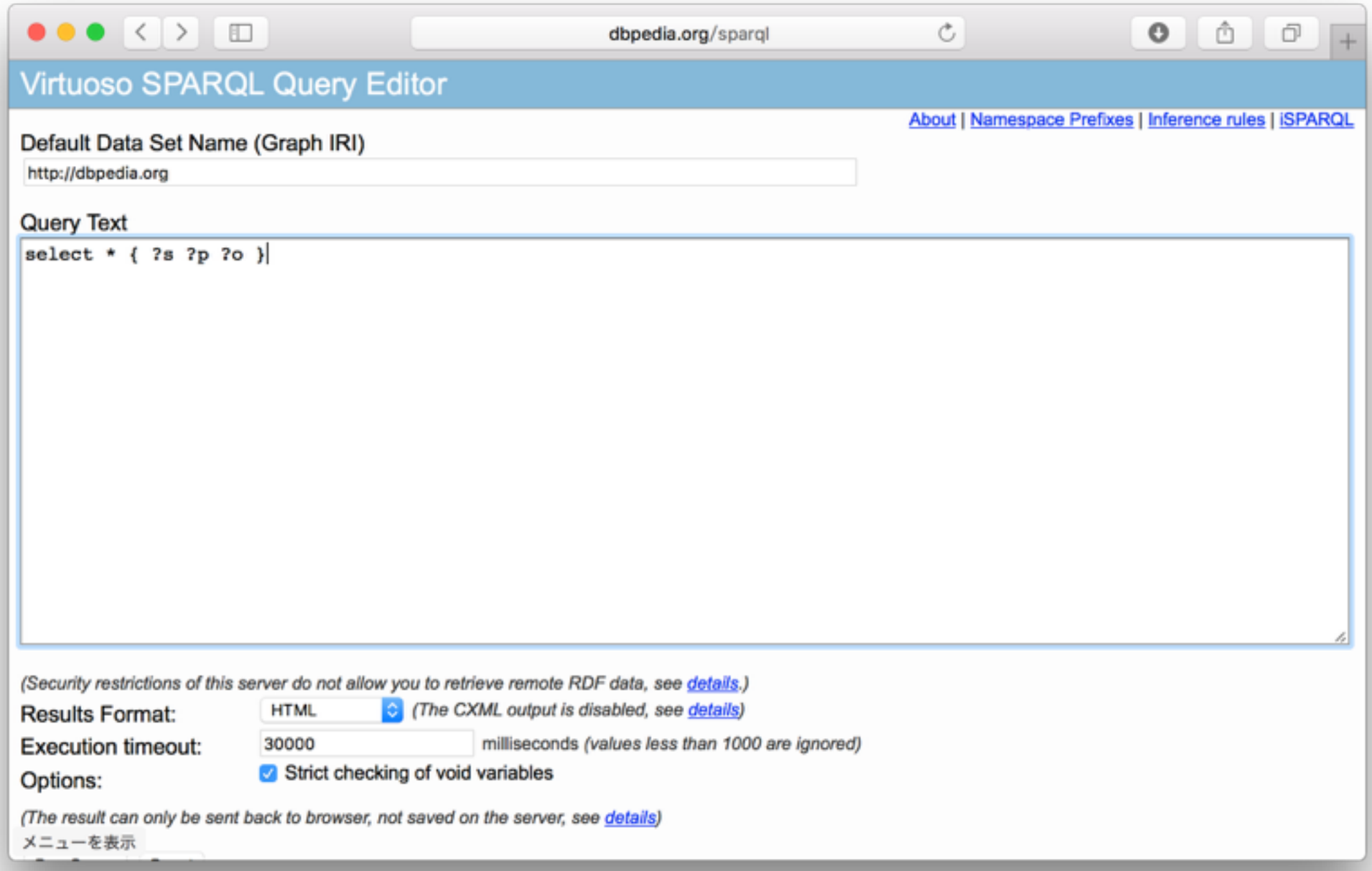
DBpediaについて

- DBpediaプロジェクトの目的
  - Wikipediaにある情報を構造化して、公開しましょう。
- その成果は
  - dbpedia.orgというサイト
- そのデータはRDFで公開し、SPARQL検索が可能

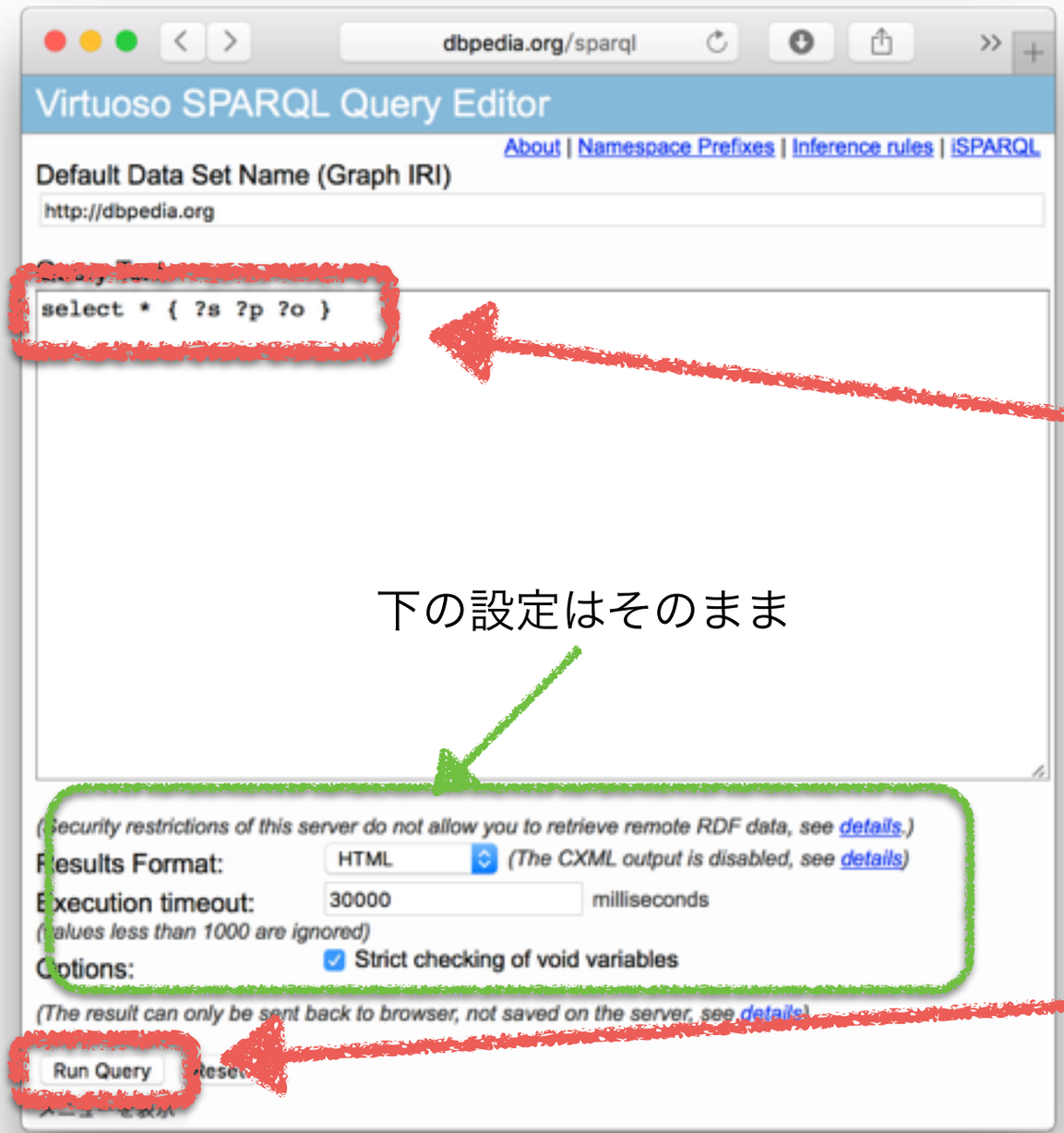


DBpedia SPARQL インタフェース

http://dbpedia.org/sparql



演習 1



下記のクエリーを書いてください：

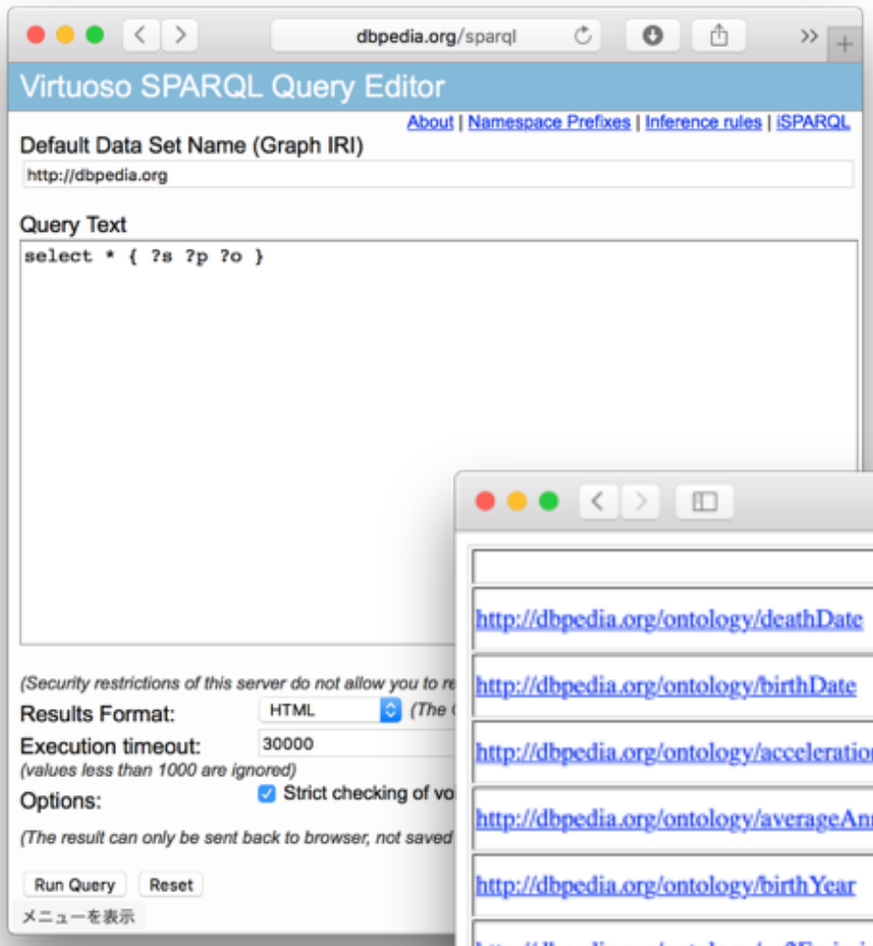
```
select * { ?s ?p ?o }
```

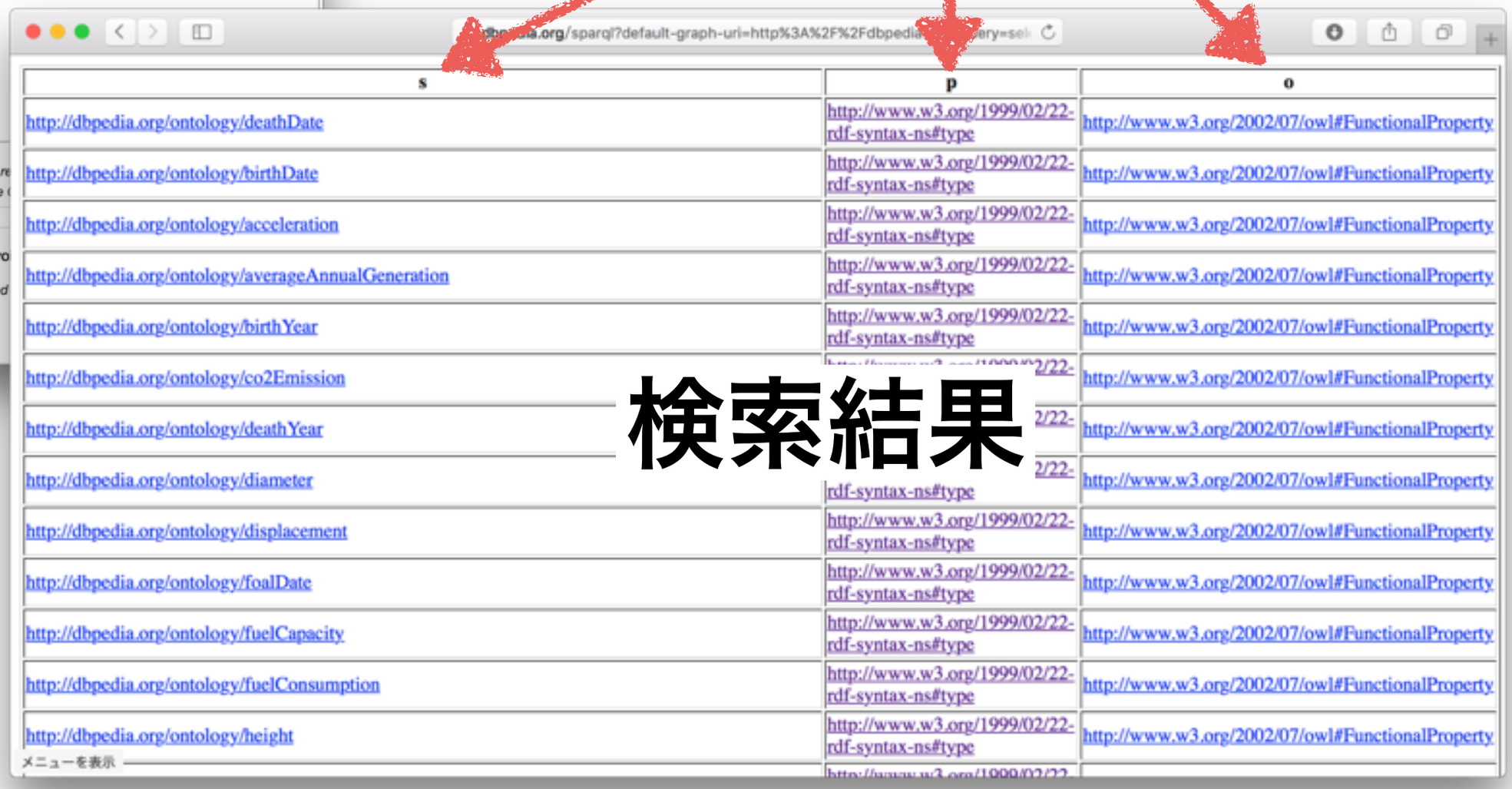
クエリーを記入したら「Run Query」を押す。

演習 1

変数s,p,oはそれぞれ全ての主語,述語,目的語とする

```
select * { ?s ?p ?o }
```





検索結果

LIMIT,OFFSETについて

- 先ほどのクエリーは合致した結果が多い
- 表示側もサーバー側も大量のデータを扱うのは負担がかかる
- その負担を減らすために、「LIMIT」や「OFFSET」SPARQLキーワードを使う
  - LIMIT：検索結果の件数の上限
  - OFFSET：全ての検索結果の中から何件目から欲しいのかを指定
- 例：
  - SELECT * { ?s ?p ?o } LIMIT 10 → １件目～１０件目
  - SELECT * { ?s ?p ?o } LIMIT 10 OFFSET 5 → ５件目～１５件目
  - SELECT * { ?s ?p ?o } OFFSET 20 → ２０件目～最後まで

演習２：LIMIT/OFFSETを使おう！

下記のクエリーを

```
select * { ?s ?p ?o }
```

「LIMIT」と「OFFSET」キーワードを使って、０から１００件目までのみ戻すように変更してください。

基本的なSPARQL
シンタックスについて

クエリーの種類を指定する動詞
・SELECTはクエリー
・他にもあります：ASK、DESCRIBE、CONSTRUCT

SELECT * { <グラフパターン> }

変数リスト。「*」は「全ての変数」と意味する

グラフパターン
・Turtleに近いシンタックスを使ったトリプル。
・クエリーの条件や合致したい情報を表現するため。

Turtleシンタックスの例

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/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		

- トリプルはドット「.」で区切る
- IRIはアングルブラケット「<…>」の中に入れる
- リテラル（文字列など）は引用符の中に入れる
- 言語とタイプはそれぞれ「^^」と「@」で指定する
  - タイプ：”2”^^integer
  - 言語：”Fukuoka”@en

```
<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 .
```

条件のつけ方について

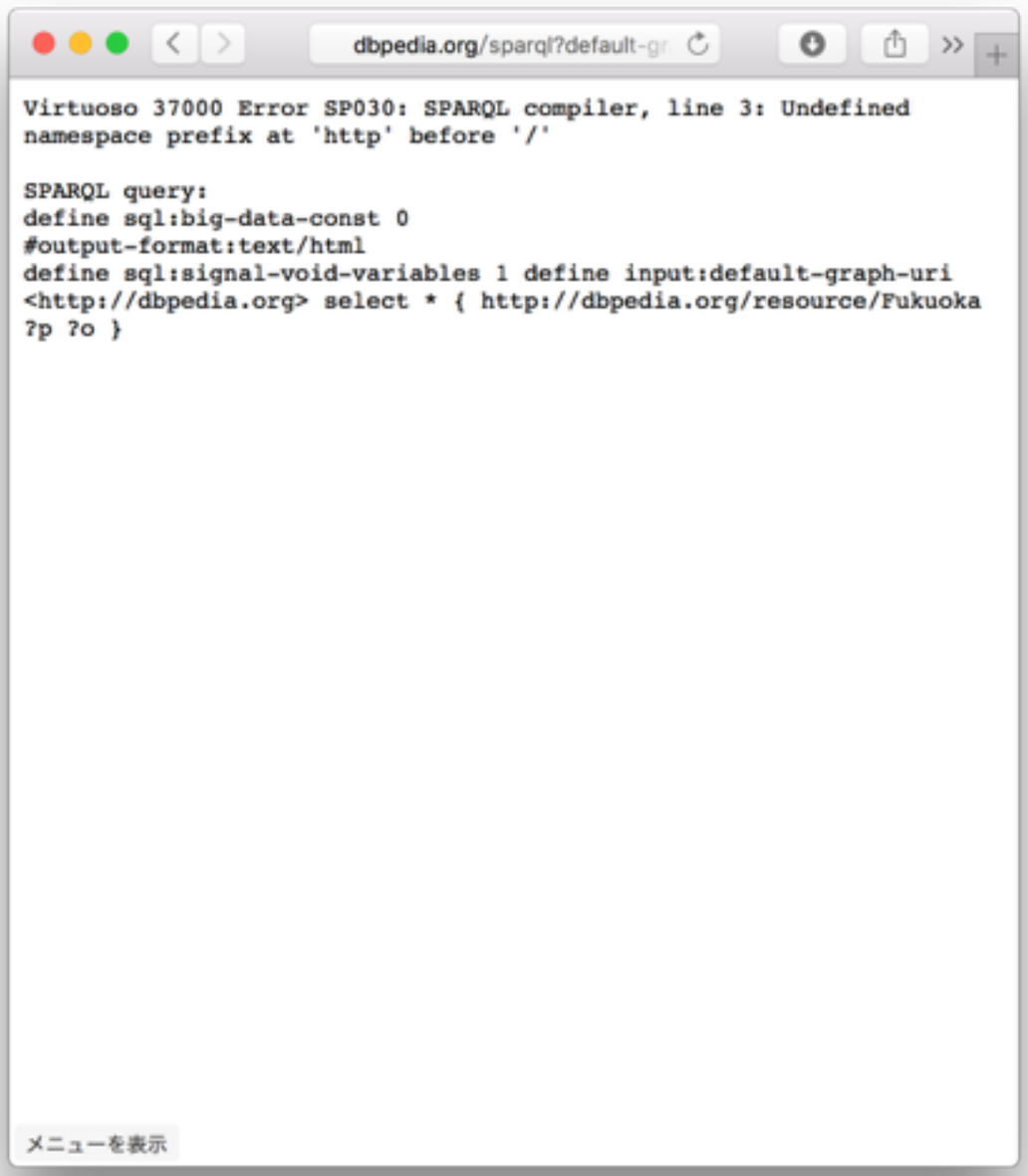
SELECT ＊ { <グラフパターン> }

- グラフパターンの中にある主語、述語、目的語は２種類があります：
  - 変数（「?」で始まる）は全てのデータを合致する
  - 定数（リテラル又はIRI）は合致トリプルに対する条件となる
- 例：
  - 「?s ?p ?o」は全てのトリプルを合致する
  - 「?s 名前 ?o」は述語が「名前」となっているトルプルのみ合致する

演習３：条件をつけてみよう

- DBpediaでは福岡市のIRIは
  - http://dbpedia.org/resource/Fukuoka
- ですが、下記のクエリー
  - `select * { ?s ?p ?o }`
- のトリプルパターンの主語に条件を追加することで、福岡市に関わる全てのトルプルを取得してください

演習３：注意点



このエラーある方はIRIの周りに「<…>」をつけてるかどうか。

演習４：述語に条件をてけてみよう！

<http://dbpedia.org/resource/Fukuoka> ?p ?o

述語

- 下記のクエリー

```
select * {
<http://dbpedia.org/resource/Fukuoka>
?p ?o }
```

- のトリプリパターンの述語に条件を追加することで、福岡市の名前を指定するトリプルを取得してください。
- DBpediaだと都市名は「http://www.w3.org/2000/01/rdf-schema#label」という述語で指定されている

演習４：答え

```
select * {
  <http://dbpedia.org/resource/Fukuoka>
  <http://www.w3.org/2000/01/rdf-schema#label>
  ?o
}
```

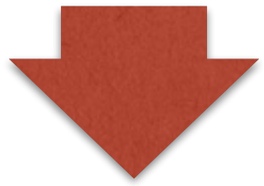
QnameとCURIE: IRIが読み

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

SPARQLでQNAMEの指定の仕方

トリプルパターンの中に
<http://www.w3.org/2000/01/rdf-schema#type>
がある場合

1. クエリーの前にプレフィックスを宣言する：
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

2. トリプルパターンの中にIRIの代わりにCURIEを使う
rdfs:type

メモ：
・CURIEは「<>」や「””」が必要ない
・プレフックス名は自由だけど、Good Practiceがあります

演習５：QNAMEを使ってみよう！

- 下記のクエリーは

```
select * {
 <http://dbpedia.org/resource/Fukuoka>
 <http://www.w3.org/2000/01/rdf-schema#label>
    ?o
}
```

- IRIをQNAMEに変更してください

「http://dbpedia.org/resource/」→「dbpedia」

「http://www.w3.org/2000/01/rdf-schema#」→「rdfs」

語彙について

- データによって諸々な語彙が必要になる
  - 自分で語彙を決めても問題ありません
  - しかし、誰かがあなたのニーズを満たす語彙を作ったことがあるかも
- W3C規格語彙があります
  - **RDF**と **RDF Schema** (RDFS)
  - **geo**：地理的情報 (e.g. 緯度 / 軽度)
  - **SKOS**：「simple knowledge organization system」
  - **XSD**, XML規格の標準語彙
- 規格ではなくても、よく使われている語彙もある
  - **Foaf** (Friend of a friend)：人間関係の表現
  - **Schema.org**：諸々な概念を表現できるような語彙（RDFaやMicrodataでよく使われている）
  - **Dublin Core**：図書データ
  - **DBPedia**, **Yago**：図鑑などデータ

語彙について

- データによって諸々な語彙が必要になる
  - 自分で語彙を決めても問題ありません
  - しかし、誰かがあなたのニーズを満たす語彙を作ったことあるかも
- W3C規格語彙があります
  - **RDF**と **RDF Sch**
  - **geo**：地理的情報 経度)
  - **SKOS**：「simple ... organizat... System」
  - **XSD**, XML規格の標準
- 規格ではなくても、よく ている語彙もある
  - **Foaf** (Friend of a fri... 間関係の表現
  - **Schema.org**：諸々な概念を表現できるような語彙（RDFaやMicrodataでよく使われている）
  - **Dublin Core**：図書データ
  - **DBPedia**, **Yago**：図鑑などデータ

語彙の探し方について

1. データベースにクエリーを投げることににによって、語彙を閲覧する（演習10と11を参考）
2. グーグルやprefix.ccで語彙の定義を確認し、関連語彙を探索する

prefix.ccを見てみましょう

prefix.ccで語彙を探索

RDFSの場合

- RDFデータのスキーマを表現するため
- W3C規格

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 .
```

語彙のTurtle式を

取得する

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 .
```

ちょっと待って、語彙は
Turtleで記述するの？

語彙のTurtle式を
取得する

Display a menu

RDFで語彙を記述する

- RDFデータのスキーマをRDFで表現できる
  - **オントロジ**という
- オントロジ自体を表現するために標準語彙があります
  - **RDFS**: W3C規格による基本的なオントロジ語彙
  - **OWL**: W3C規格による複雑なオントロジ語彙
  - **SPIN**: SPARQLでオントロジ条件を表現できる語彙
- オントロジ語彙では複雑な概念が記述可能！
  - モデルドリブン開発へ
- オントロジはRDFデータに組み込むと誰でもSPARQLでデータの構成を探索できるようになる！

プログラムの理論はオントロジ語彙で表現できる箇所が多い

演習５の結果を詳しく見る：言語について

0
"Fukuoka"@en
"فوكوكا ، فوكوكا"@ar
"Fukuoka"@de
"Fukuoka"@es
"Fukuoka"@fr
"Fukuoka"@it
"福岡市"@ja
"Fukuoka (stad)"@nl
"Fukuoka"@pl
"Fukuoka (cidade)"@pt
"Фукуока"@ru
"福岡市"@zh

ISO 639規格に従う言語指定文

演習6：あなたの故郷の情報を取得しよう！

- 今までは主語と述語に条件を付けましたが、今回は目的語につけてみよう！
- 「rdfs:label」の目的語としてあなたの故郷の名前を入れることのよって、あなたの故郷のIRIを検索してみよう！
- 注意点
  - グラフパタンの主語は条件の代わりに変数になる
  - 名前は言語の指定を忘れずに！

下記のような結果が欲しい

s
http://dbpedia.org/resource/Poitiers
http://www.wikidata.org/entity/Q6616

僕の故郷はフランス語で「Poitiers」ですので、リテラルは”Poitiers”@frです。

FILTERキーワード

- FILTERキーワードは合致するリテラルに更に条件を付けることができます：

```
select * {
   <グラフパターン> .
   FILTER(条件)
}
```

条件の書き方は通常のプログラミング言語に似ています：
・変数とリテラルの関係性を表現できる（=,<,>,!=など）
・SPARQLは様々な関数も定義する。例（x,y,zは変数、リテラルまたは式）：
  ・lang(x)：xの言語
  ・abs(x)：xの絶対値（数値のみ）
  ・isIRI(x)：xはIRIかどうかを確認
  ・IF(x,y,z)：分岐。
  ・IRI(x)：IRIへ変換（文字列のみ）

演習７：初めてのFILTER

- 下記のクエリー

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

select * { dbpedia:Fukuoka rdfs:label ?o }
```

- を変更し、FILTER文を使って日本語の名前のみ取得する

o
"Fukuoka"@en
"فوكوكا ، فوكوكا"@ar
"Fukuoka"@de
"Fukuoka"@es
"Fukuoka"@fr
"Fukuoka"@it

下記の結果を求める

o
"福岡市"@ja

演習８：より複雑なグラフパタンを使ってみよう！

- グラフパタンでは複数のトリプルを指定できます。
  - 「.」で区切る。例：{ s1 p1 o1 . s2 p2 o2 }
- 例えば、下記のグラフパタンは福岡の名前と人口を合致する：

```
dbpedia:Fukuoka rdfs:label ?name .
dbpedia:Fukuoka http://dbpedia.org/property/populationTotal ?pop .
```

- 問題：上のグラフパタン使ったクエリーを実行してみよう

演習８の結果

name	pop
"Fukuoka"@en	1483052
"فوكوكا، فوكوكا"@ar	1483052
"Fukuoka"@de	1483052
"Fukuoka"@es	1483052
"Fukuoka"@fr	1483052
"Fukuoka"@it	1483052
"福岡市"@ja	1483052
"Fukuoka (stad)"@nl	1483052
"Fukuoka"@pl	1483052
"Fukuoka (cidade)"@pt	1483052
"Фукуока"@ru	1483052
"福岡市"@zh	1483052

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

select * {
  dbpedia:Fukuoka rdfs:label ?name .
  dbpedia:Fukuoka dbpedia-p:populationTotal ?pop .
}
```

人口は何度も出てきます。
それは、「 dbpedia:Fukuoka rdfs:label ?name .」が何度も合致するから：合致した各名前に人口をクエリーしてしまう。

演習８の改善

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

select * {
  dbpedia:Fukuoka rdfs:label ?name .
  dbpedia:Fukuoka dbpedia-p:populationTotal ?pop .
}
```

name	pop
"Fukuoka"@en	1483052
"فوكوكا، فوكوكا"@ar	1483052
"Fukuoka"@de	1483052
"Fukuoka"@es	1483052
"Fukuoka"@fr	1483052
"Fukuoka"@it	1483052
"福岡市"@ja	1483052
"Fukuoka (stad)"@nl	1483052
"Fukuoka"@pl	1483052
"Fukuoka (cidade)"@pt	1483052
"Фукуока"@ru	1483052
"福岡市"@zh	1483052

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

select * {
  dbpedia:Fukuoka rdfs:label ?name .
  dbpedia:Fukuoka dbpedia-p:populationTotal ?pop .
  FILTER(lang(?name)=“ja”)
}
```

name	pop
"福岡市"@ja	1483052

グラフパタンを短くする

主語
の再利用

```
<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 .
```

セミコロンで述語・目的語を句切る

最後だけドットを付ける

述語
の再利用

```
<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 .
```

コロンで目的語を句切る

演習９：前のクエリーを短くしましょう！

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

select * {
  dbpedia:Fukuoka rdfs:label ?name .
  dbpedia:Fukuoka dbpedia-p:populationTotal ?pop .
  FILTER(lang(?name)="ja")
}
```

セミコロンを使って主語を再利用しましょう

DISTINCTキーワード

- 変数リストの中に「DISTINCT」を指定すると、何度も同じ結果が合致しても一回しか戻さない
- 例：下記のクエリーはDBpediaのデータベースが使っているの全ての述語を返す

```
select DISTINCT ?p {
  ?s ?p ?o
}
```

演習10：DBpediaの語彙を探索してみよう（1）

- DISTINCTキーワードを使って、福岡市のデータに使っている全ての述語を取得しましょう！

演習11：DBpediaの語彙を探索してみよう（2）

- 今までは下記の述語を使いました：
  - rdfs:label：都市の名前
  - dbpedia-p:PopulationTotal：人口
- 都市の国を表すプロパティを見つけてください
- ヒント：福岡市のデータを見て探索するのはおすすめ
  - 「福岡は日本に位置する」という概念を表すトリプルを見つける

rdf:typeについて

- 「http://www.w3.org/1999/02/22-rdf-syntax-ns#」はW3CのRDF規格の標準的な語彙
  - 普段は「rdf」というプレフィックスを使う
- 「rdf:type」はリソースのタイプを表すような述語
- タイプの概念はオブジェクト指向言語に似ている
  - オントロジーを使ってオブジェクトのメンバーなどを定義できる
- ただし、RDFでは、あるリソースは複数のタイプを持つことができる
- SPARQLで「http://www.w3.org/1999/02/22-rdf-syntax-ns#/type」のかわりに「a」キーワードを利用できる
  - 例：?s a <http://dbpedia.org/ontology/City>

演習12：タイプを見てみよう

- 福岡市のリソースのすべてのタイプを取得する
- リマインダー：福岡市のIRIは「http://dbpedia.org/resource/Fukuoka」

t
http://www.w3.org/2002/07/owl#Thing
http://dbpedia.org/ontology/Place
http://dbpedia.org/ontology/Location
http://www.wikidata.org/entity/Q486972
http://www.wikidata.org/entity/Q515
http://dbpedia.org/ontology/City
http://dbpedia.org/ontology/PopulatedPlace
http://dbpedia.org/ontology/Settlement
http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing
http://schema.org/City
http://schema.org/Place
http://umbel.org/umbel/rc/City
http://umbel.org/umbel/rc/Location_Underspecified

このような結果が欲しい！

演習13：より複雑なグラフパタンを使ってみよう

- 下記の述語を使って：

  `http://dbpedia.org/ontology/country`
- 日本のすべての都市の日本語の名前と人口を取得する
- ヒント、注意点：
  - 日本のIRIは「dbpedia:Japan」
  - 都市以外にも「http://dbpedia.org/ontology/country」を使うリソースがあるので、都市（英語：”city”）のみ合致するように工夫する必要がある

name	pop
"我孫子市"@ja	132999
"昭島市"@ja	112521
"秋田市"@ja	323232
"奄美市"@ja	44561
"青森市"@ja	293442
"旭市"@ja	68265
"浅口市"@ja	36985

このような結果が欲しい！

演習14：数値にFILTERをかけよう！

- 演習12のクエリーを変更して、FILTERキーワードを使って人口が１０万を超えてる都市だけ取得する
- ヒント：FILTERを複数使いたい場合、やり方が２つあります
    - FILTERキーワードを複数使って、ドットで区切る
        - 例：FILTER(条件1) . FILTER(条件2)
    - FILTERの条件はブール理論を使う
        - 例：FILTER(条件1 && 条件2)

演習15：ソートする

- 「ORDER BY」キーワードをクエリーの後ろにつけると結果をソートできる。例（?popは人口とする場合）：
  - SELECT * { … } ORDER BY ?pop：人口の少ない都市から
  - SELECT * { … } ORDER BY DESC(?pop)：人口の多い都市から
- 演習13のクエリーを人口の多い都市から結果を出すように変更してください

演習16：GROUP BYとCOUNTとは？

下記のクエリーは何を検索しているのかを当ててみてください：

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

select ?country COUNT(DISTINCT ?s) {
  ?s a dbpedia-o:City ;
  dbpedia-o:country ?country ;
  dbpedia-p:populationTotal ?pop .
  FILTER(?pop > 100000)
} GROUP BY ?country
```

演習17：「as」キーワード

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

select ?country (group_concat(?name, ",") as ?names) {
  ?s dbpedia-o:country ?country ;
  rdfs:label ?name ;
  dbpedia-p:populationTotal ?pop .
  FILTER(lang(?name)="ja" && ?pop > 100000) .
} GROUP BY ?country
```

SERVICEを使った結合クエリー

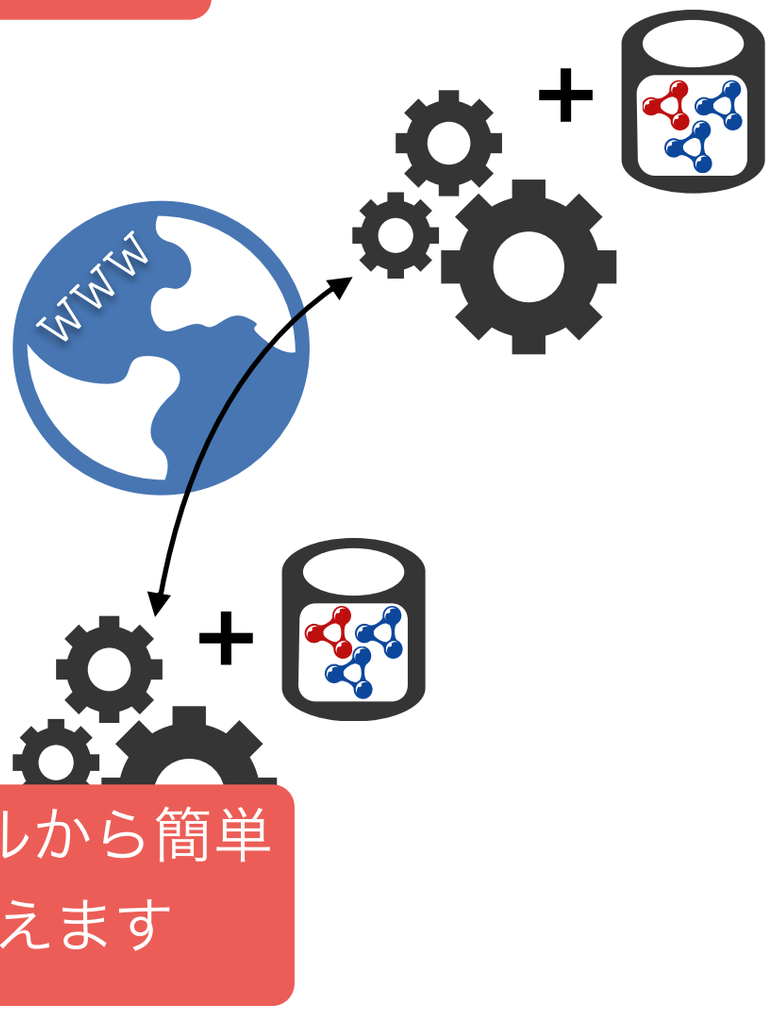
同時複数のSPARQLエンドポイントにクエリーを投げる

```
SELECT ?facility_name ?city_name
WHERE {
  ?s a bodic:PublicFacility ;
    rdfs:label ?facility_name ;
    bodic:locatedIn ?city_iri .
  SERVICE <http://dbpedia.org/sparql> {
    ?city_iri rdfs:label ?city_name
  }
}
```

dbpediaのSPARQLエンドポイントのURL

・SPARQLクエリーの中から**マッシュアップできる**

・しかし、SPARQLエンドポイントに対して負担が大きいため、公開エンドポイントはSERVICEを禁止することが多い

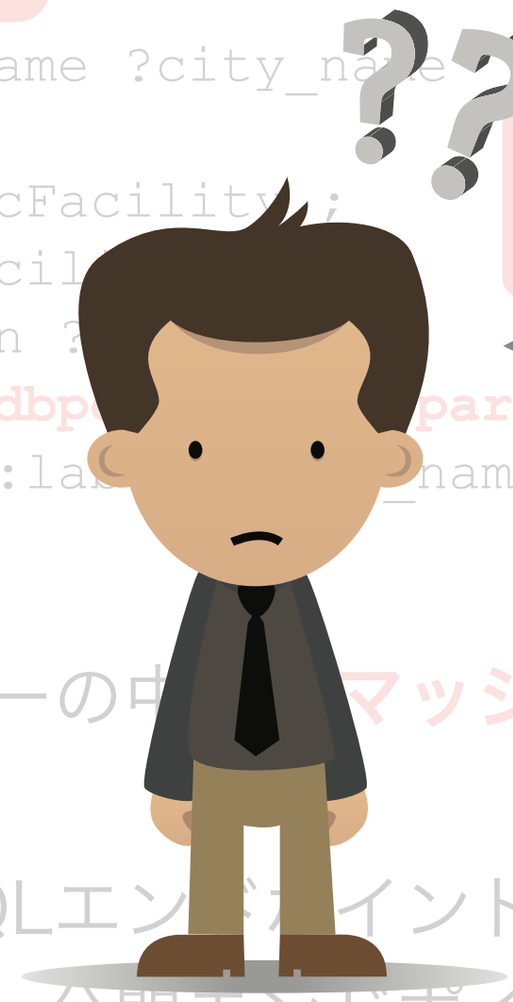


Jenaのようなツールから簡単にローカルで使えます

SERVICEを使った結合クエリー

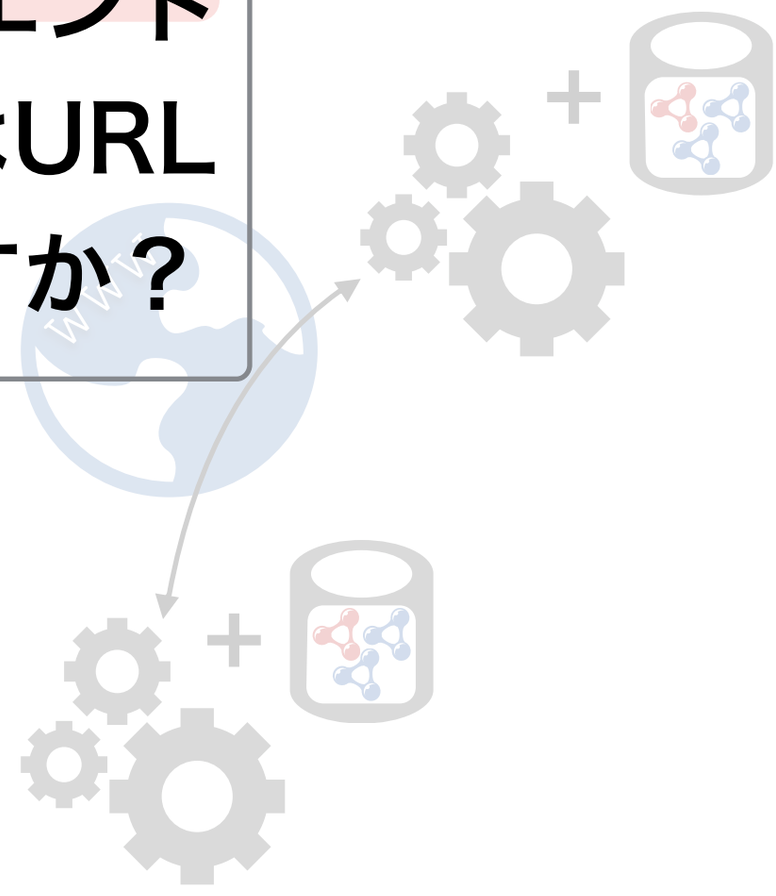
同時複数のSPARQLエンドポイントにクエリーを投げる

```
SELECT ?facility_name ?city_name
WHERE {
  ?s a bodic:PublicFacility;
    rdfs:label ?facil
    bodic:locatedIn ?
  SERVICE <http://dbp          parql>
    ?city_iri rdfs:la          name
  }
}
```



dbpediaのSPARQLエンドポイントのURL

SPARQLエンドポイントはURLがありますか？



・SPARQLクエリーの中で**マッシュアップ**できる

・しかし、SPARQLエンドポイントに対して負担が大きいため、公開エンドポイントはSERVICEを禁止する

Jenaのようなツールから簡単にローカルで使えます

SPARQLはクエリー言語
だけではない

SPARQL Protocol And RDF Query Language

- SPARQL 1.1では２つのプロトコルを定義している
  - SPARQLクエリーを投げるためHTTP REST API
  - SPARQLエンドポイント間で通信するためのプロトコル
- 簡単にHTTP REST APIを見ましょう

SPARQL HTTP REST APIについて

- SPARQLエンドポイントサーバーAPIがhttp://endpoint.orgとすると
- SPARQL検索
  - http://endpoint.org**/sparql**
    - 読み込み専用クエリーを投げるため (SELECT/CONSTRUCT/ASK/DESCRIBE)
  - http://endpoint.org**/update**
    - SPARQL UPDATE クエリーを投げるため (LOAD/INSERT/DELETE/DROP/COPY/MOVE)
- 直接RDFグラフを扱うAPI(http://endpoint.org**?graph=graph_name**)
  - **GET** グラフを取得
  - **PUT** グラフを書き換える
  - **POST** グラフを更新
  - **DELETE** グラフを削除

グラフとは

・RDFストアはトリプルをグラフ単位で管理する（関係データベースのテーブルのように）

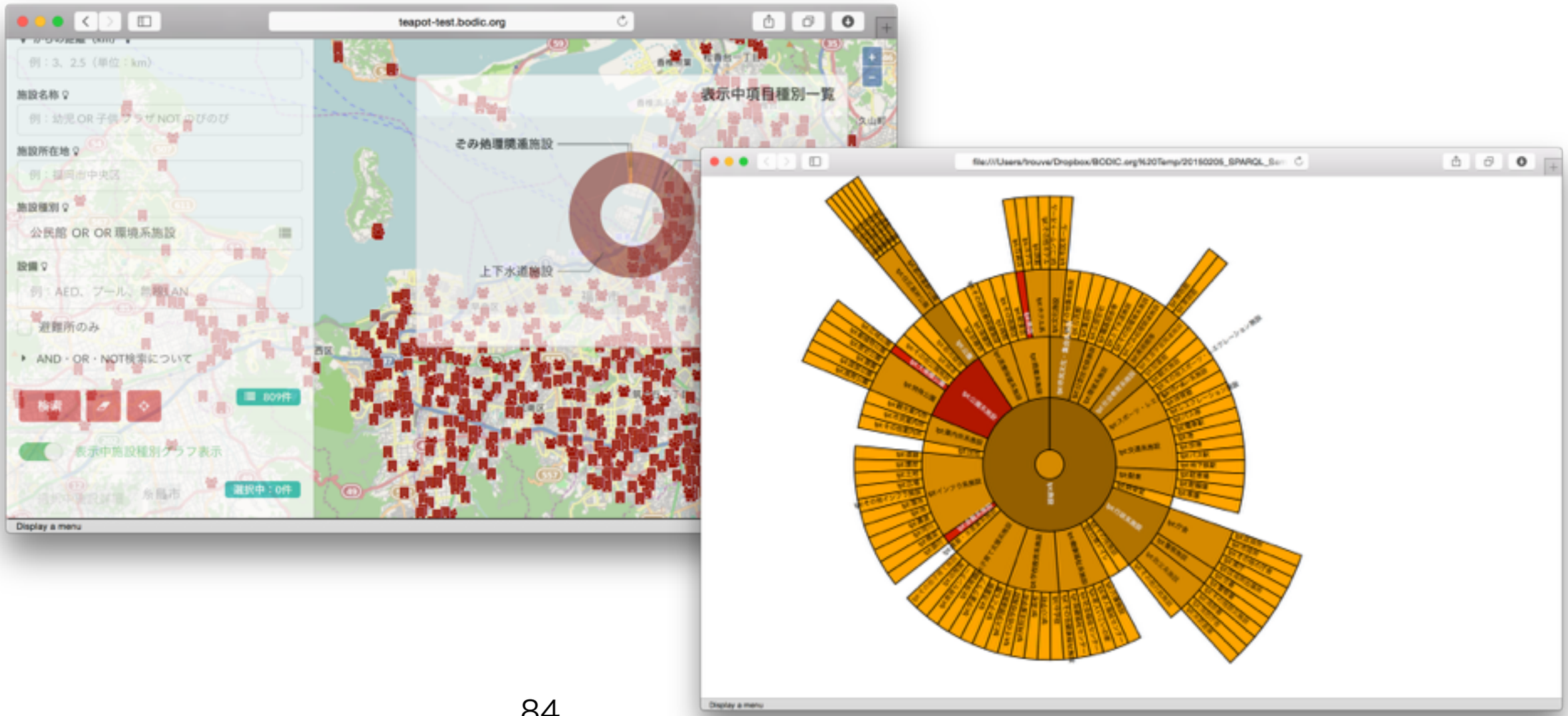
・SPARQLでGRAPHを指定しないと「default」というグラフから検索する

まとめ

- 本日は基本的なSPARQLクエリーを勉強しました
- まだ勉強できなかった機能が沢山あります：
  - 結合クエリー演習
  - キーワード：BIND、UNION、MINUSなど
  - その他の関数
  - クエリー種：CONSTRUCT、ASK、DESCRIBE、SPARQL UPDATEクエリー群
  - オントロジ
- SPARQLはセマティックウエブ以外でもよく使われている
  - NoSQLデータベースの1種類として
  - 向き不向きがありますが、ソフト工学者のツールボックスにあると便利

次回（予定）

- 地図とグラフでSPARQL結果を可視化する
- クライエント側（AJAX）のウエブ開発に関する演習



お疲れさまでした！