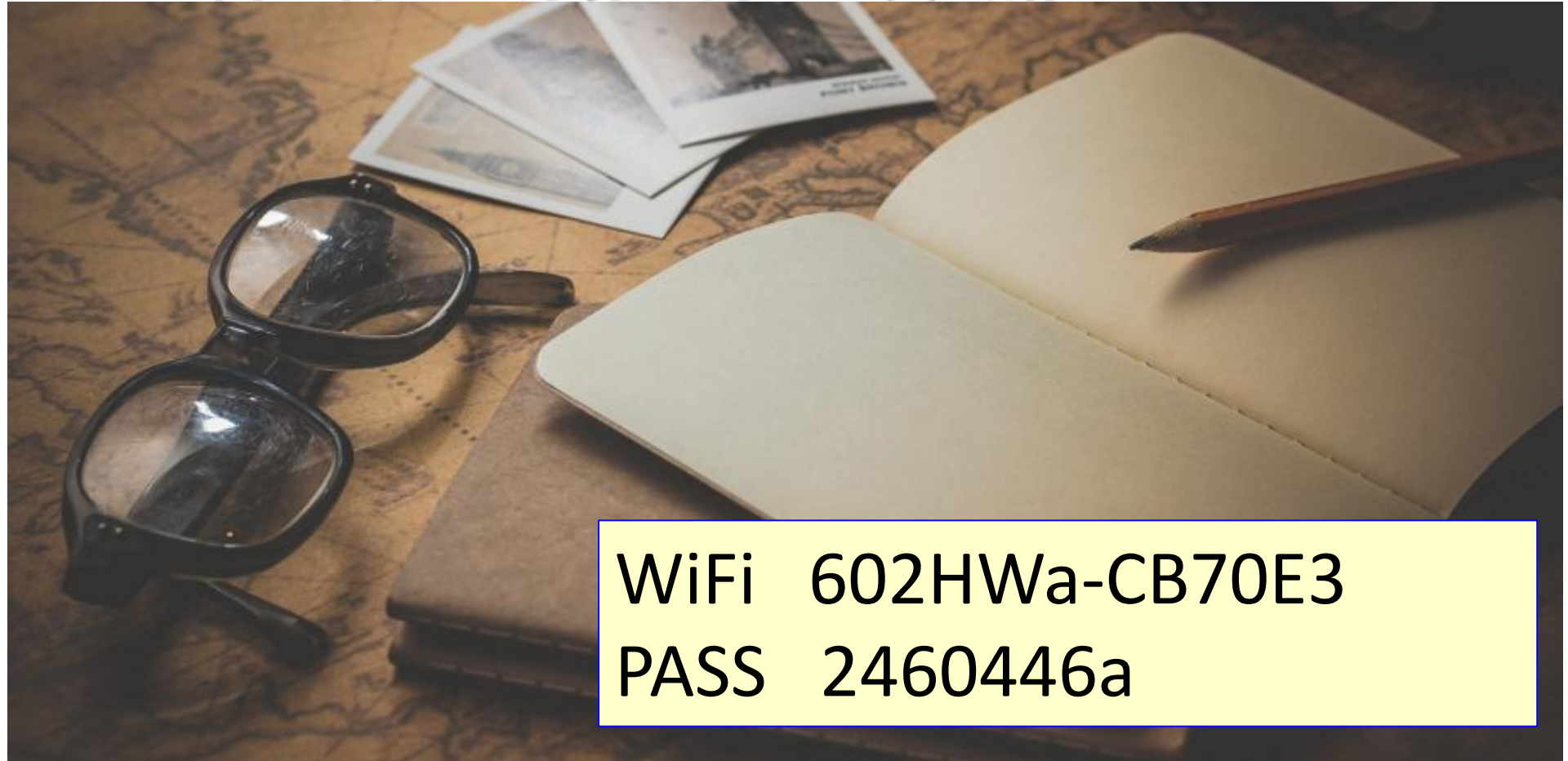


人工知能学会SWO研究会ワークショップ

「ナレッジグラフ推論チャレンジ2019技術勉強会 & 応募相談会」

：まだ間に合う！応募に向けた実践講座

10/21(月)

A photograph of a desk setup. On the left, a pair of black-rimmed glasses rests on a brown leather-bound book. To the right of the glasses, an open notebook with cream-colored pages lies flat. A wooden pencil is positioned diagonally across the right page of the notebook. In the background, several small, vintage-style photographs are scattered on a surface covered with a detailed map. The lighting is warm and focused on the notebook and glasses.

WiFi 602HWa-CB70E3
PASS 2460446a

➡ 本イベントのねらい

➤「推論チャレンジ」に応募するのに必要なナレッジグラフの基礎技術の基本を学んでいただく

➡ 本日の予定

14:30 第2回ナレッジグラフ推論チャレンジの紹介

14:45 ナレッジグラフ利用技術のハンズオン
(大阪電気通信大学・古崎晃司)

16:15 休憩

16:30 推論チャレンジ応募に向けたアプローチ例の紹介
(富士通研究所・松下京群, 鵜飼孝典)

17:15 第2回ナレッジグラフ推論チャレンジ2019への応募相談

17:30 終了

第2回

ナレッジグラフ推論チャレンジの紹介

1. 推論チャレンジとは

2. 対象とするナレッジグラフの概要

◆ シャーロック・ホームズのような

“推理”(推論)ができるAIシステムの開発
を目指した技術コンテスト

➡ チャレンジのねらい

➤ **説明可能性(解釈可能性)**を有するAI技術に関する最新技術の促進・共有と、その分析・評価、体系化を行う。

➡ チャレンジタスク

➤ **推理小説のナレッジグラフ**(ホームズの短編小説)を対象に、ホームズと同じ結論に辿り着き、その理由を説明する。

- ▶ **説明可能性(解釈可能性)**を有するAI技術に関する最新技術の促進・共有と, その分析・評価, 体系化を行う.
- ▶ 特に, 現実社会を反映したより複雑な, 例えば時間的, 因果関係的, 確率的関係性を含む問題を扱うため, **帰納的な機械学習(推定)**と**演繹的な知識活用(推論)**を融合したAI技術を対象とする.

- ➡ **推理小説のナレッジグラフ**（シャーロックホームズの短編小説）を対象に，ホームズと同じ結論に辿り着き，その理由を説明する.
 - 現実社会の複雑な関係性を含みながら仮想的にクローズな（答えがあり，それに至る制約を制御できる）タスクを設計できる
 - タスクによっては不確実情報や証拠写真など確率的な処理や機械学習を入れないと解けなかったり，陽に書かれていない常識知識を補完しなくては解けない等，推定と推論の融合を促せる
 - 読者が納得しないと小説として成立しないという人間に対する説明性を有している，
 - 小説が広く一般に知られており関心を引きやすい，など
- ➡ 但し，第4回チャレンジ以降は社会問題解決に関するベストプラクティス集なども予定



対象とする推理小説

➡ まだらのひも(第1回のKGの不具合を修正して利用)

- タスク: ヘレンを殺したのは誰か? (犯人+説明)

に加え, 新たに, 以下の4編をKG化

➡ 踊る人形 [[Wikipedia](#)] [[青空文庫](#)]

- タスク: 暗号を解け (暗号の解読)

➡ 背中の曲がった男(曲がれる者) [[Wikipedia](#)] [[青空文庫](#)]

- タスク: バークリはなぜ死んだのか? (説明)

➡ 悪魔の足 [[Wikipedia](#)] [[青空文庫](#)]

- タスク: 各人物を殺したのは誰か? (犯人+説明)

➡ 花婿失踪事件(同一事件) [[Wikipedia](#)] [[青空文庫](#)]

- タスク: 花婿はなぜ消えたか? (説明)



▶ 対象とするKG

- 5つの小説のうち、いずれの小説を対象にしてもよい
(どれか1つだけ, 2つだけ…などでもOK)
- できるだけ多くの小説が、同じシステム(仕組み)で解けるとよい
- 各小説で使用するKGの範囲を段階的に変える(昨年同様)
→ 完全(すべてのKG) / 不完全(10%) / 不完全(10%)
- ナレッジグラフの独自拡張も可能(昨年同様)

▶ 対象とするタスク

- ①本部門: 対象小説1つ以上のタスクを解くシステムを開発
- ②ツール部門: いずれかのタスクを部分的に解くツールを開発
例) 容疑者の推定, アリバイ検証, 動機説明, など
★「自然言語文をトリプル化」するKG構築支援ツールの応募も可
- アイデア部門: ①, ②の実現方法のアイデア(実装なしでOK)

第2回

ナレッジグラフ推論チャレンジの紹介

1. 推論チャレンジとは

2. 対象とするナレッジグラフの概要

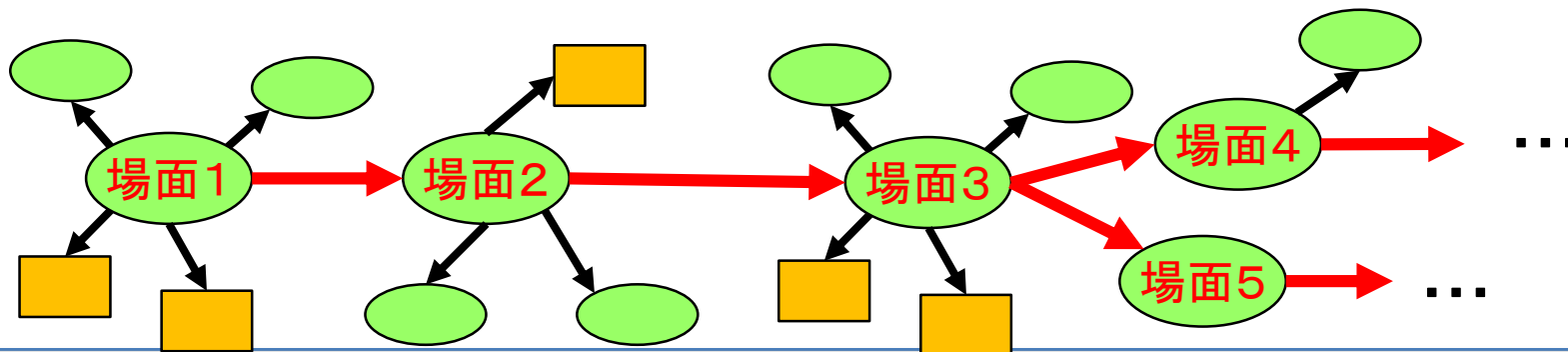


ナレッジグラフの要求仕様

- 犯人を推論(推理)するのに必要な知識を提供する
- 「推理小説」で描かれる様々な状況を, できるだけ統一的な形式で計算機処理(検索・推論・etc.)可能にする

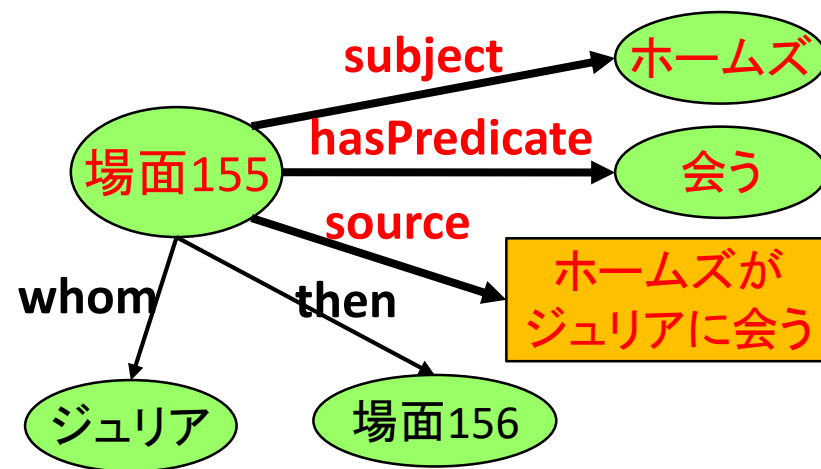
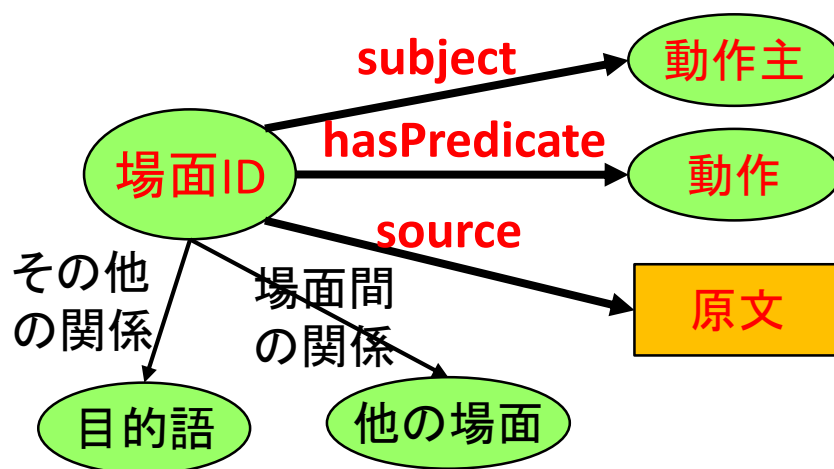
ナレッジグラフ化の基本方針

- 「推理小説」の内容を, 最小単位の「**場面(シーン)**」に分割
→ 場面ごとにID(IRI)を付与
- 「**各場面の記述内容**」および「**場面間の関係**」をグラフ化
→ グラフ化に必要なクラス・プロパティを定義

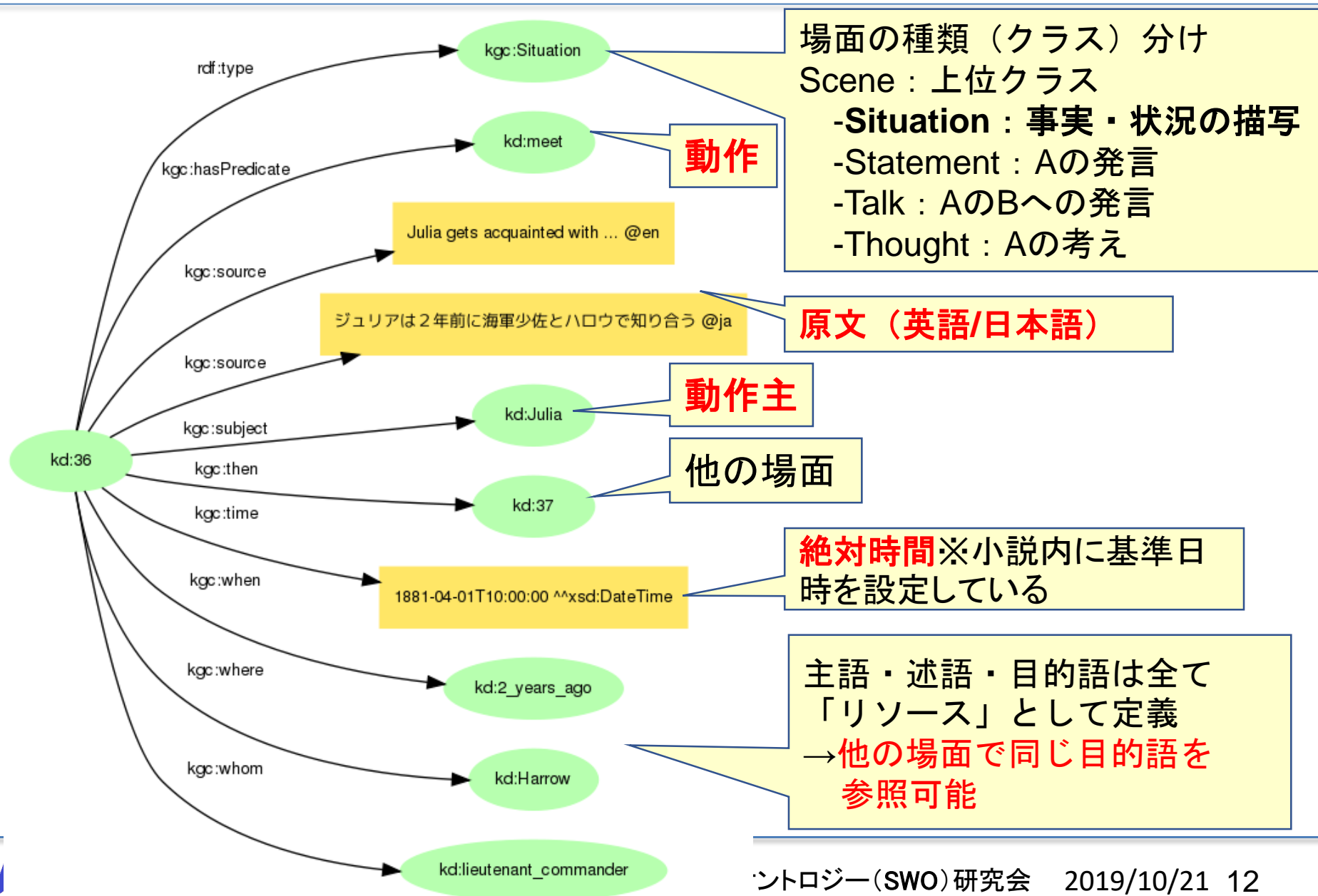


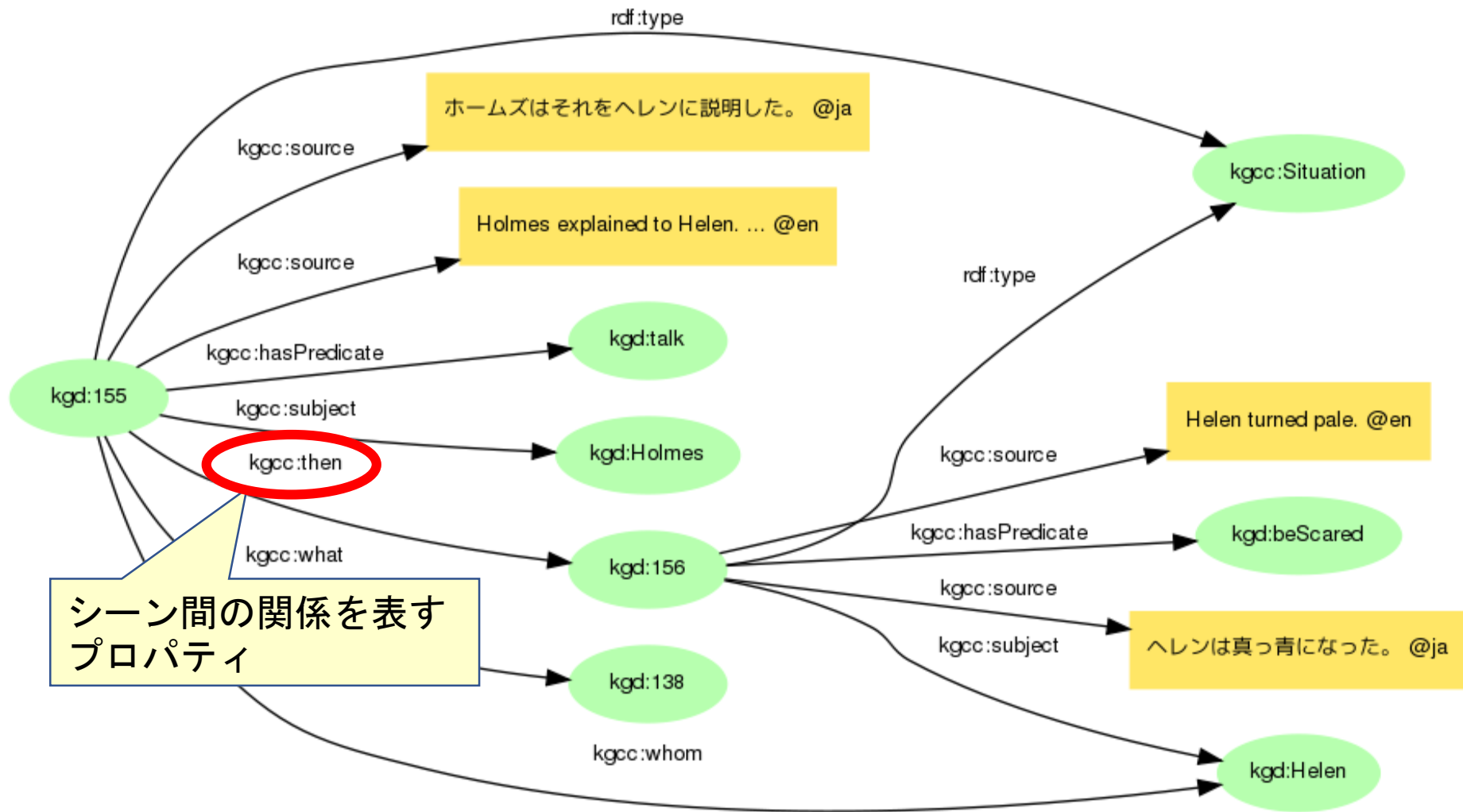
➡ 場面を表現するプロパティ

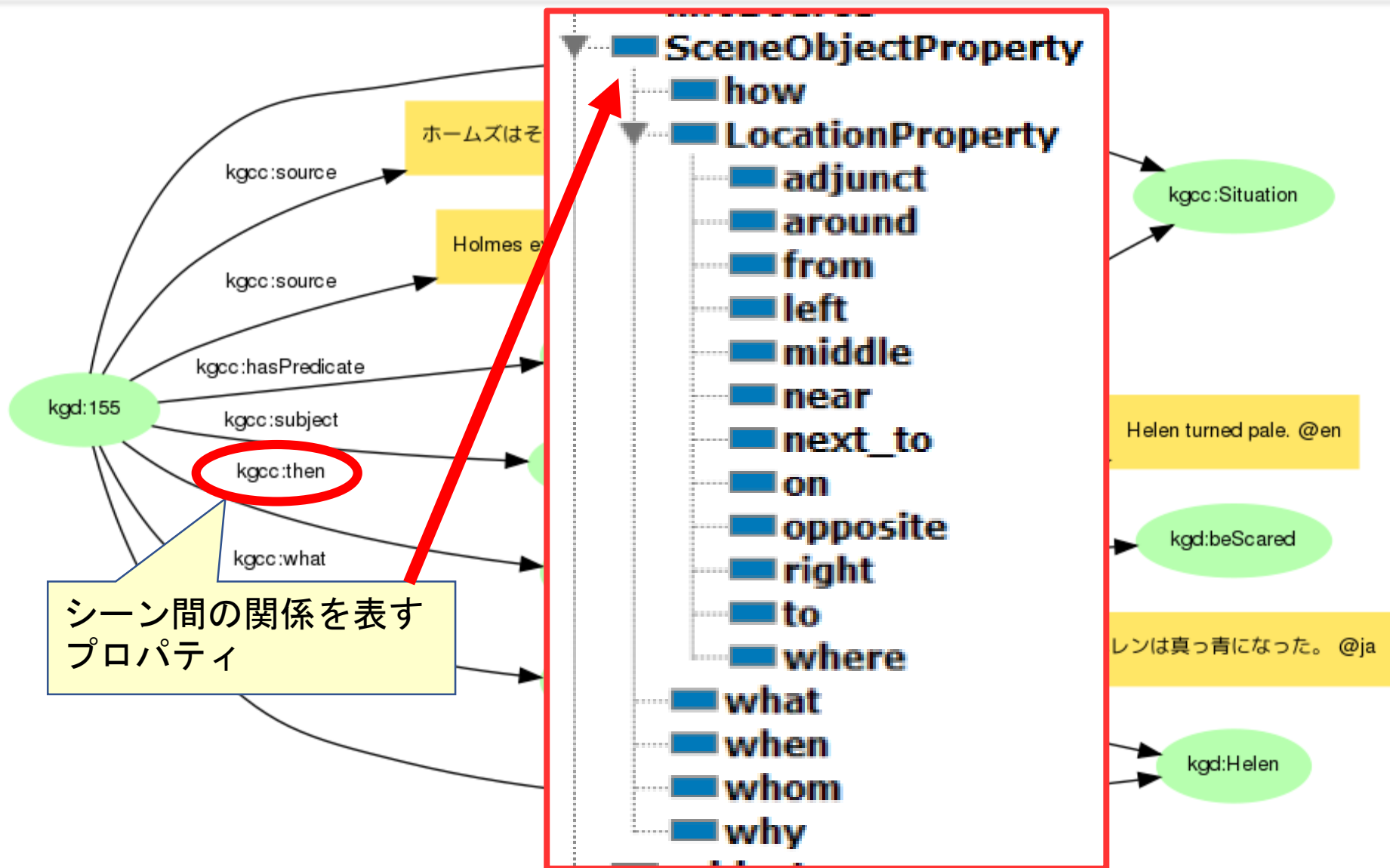
- **subject**: その場面の記述において主語となる人や物
- **hasPredicate**: その場面の内容を表す述語
- **場面の詳細を表す目的語**: whom(だれに), where(どこで), when(いつ), what(何を), how(どのように), ...etc.
- **場面間の関係**: then, if, because, ...etc.
- **time**: その場面が起こった絶対時間(xsd:DateTime)
- **source**: その場面の原文(英語/日本語のリテラル)

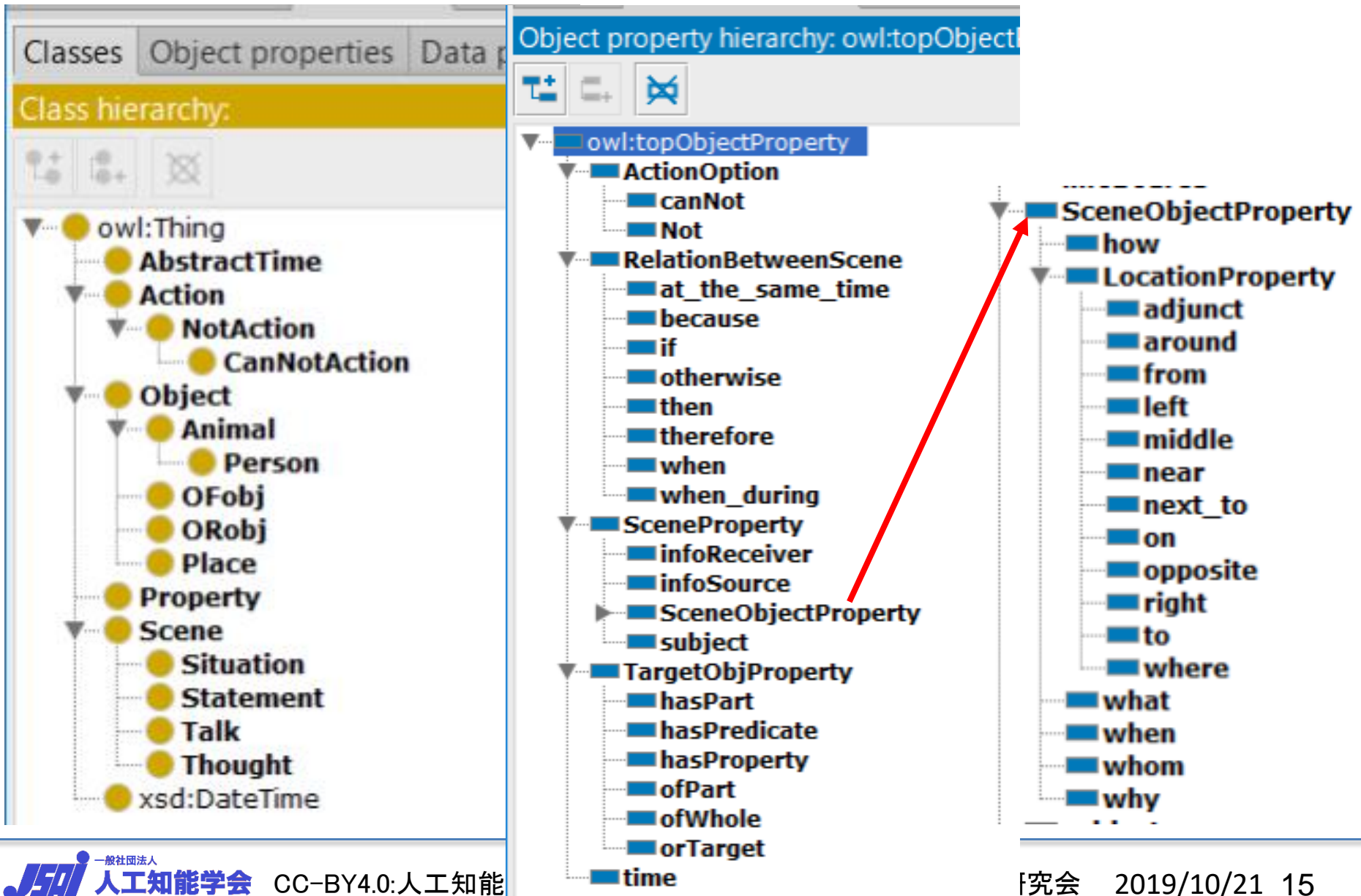


場面(シーン)スキーマ 記述例









SWO研究会・勉強会での予備的作業を経て、有志数名でナレッジグラフ化

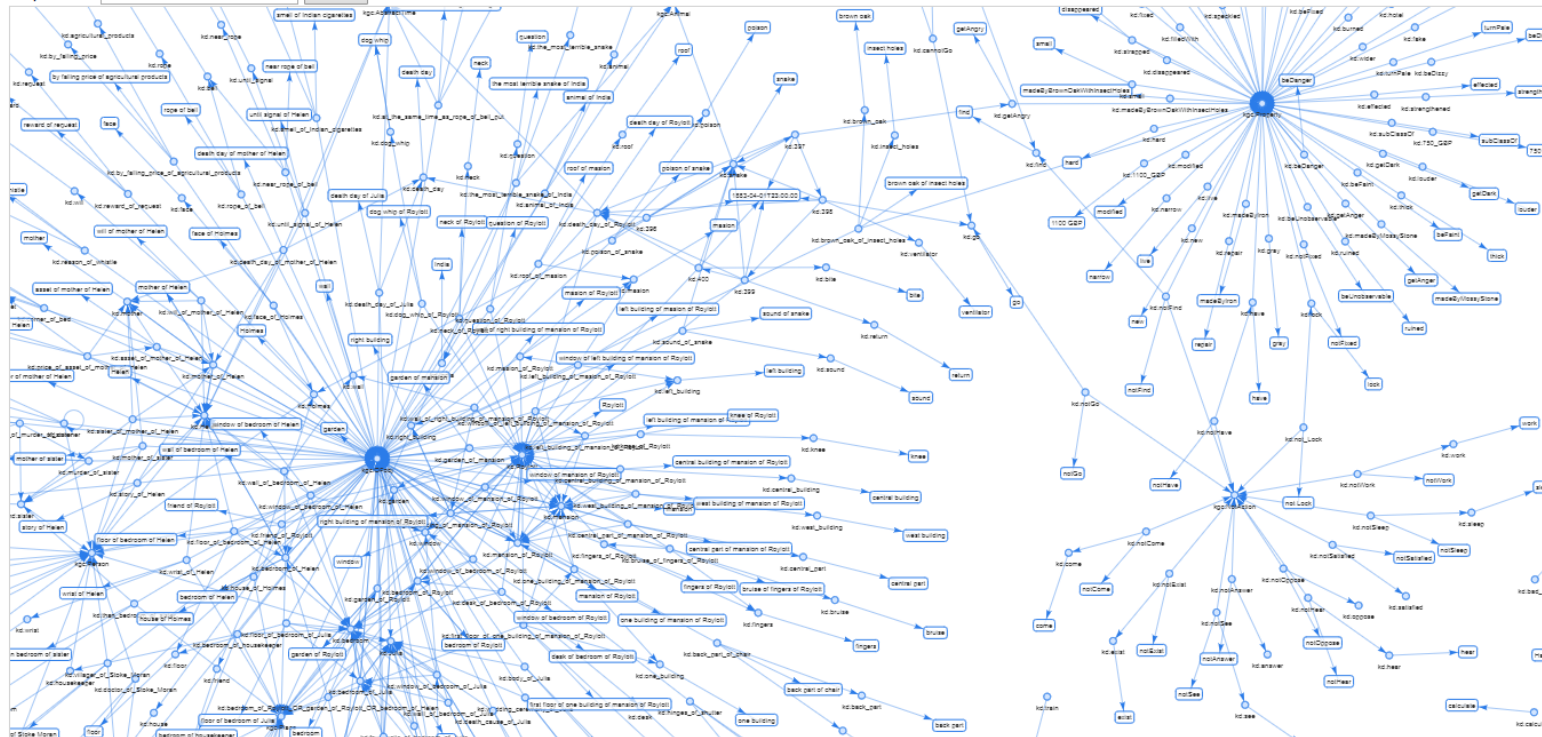
```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX kgc: <http://kgc.knowledge-graph.jp/ontology/kgc.owl#>
PREFIX kd: <http://kgc.knowledge-graph.jp/data/SpeckledBand/>
SELECT ?s ?p ?o
WHERE {
  ?s ?p ?o .
  filter(?p != rdfs:comment)
  filter(?p != kgc:source)
  filter(?o != kgc:Situation)
  filter(?o != kgc:Action)
  filter(?o != kgc:Object)
}
limit 1000
```

クエリ言語SPARQLによる検索

SPARQL Endpoint: <http://lod.hozo.jp/repositories>
Keyword:

グラフDB(キーワード検索也可)

ナレッジグラフ(RDF形式)



<http://knowledge-graph.jp/visualization/>

- ➡ 対象とする小説が5つに増えたことで、異なる小説を横断した処理ができるようにスキーマを拡充
- ➡ 各小説ごとに「ベースIRI」および「グラフIRI」を導入
 - <http://kgc.knowledge-graph.jp/data/**小説名**/>
 - 例:「踊る人形」のシーン100は
<http://kgc.knowledge-graph.jp/data/**DancingMen**/100>
- ➡ 小説をまたいで共通化した語彙
 - hasPredicateで参照する「述語」、および、hasPropertyで参照する「性質・状態」
→<http://kgc.knowledge-graph.jp/data/**predicate**/XXX>
 - 固有名詞(例:ホームズ, 地名)
→<http://kgc.knowledge-graph.jp/data/**YYY**>
は, 小説をまたいで, 共通のIRIで定義・参照

➡ 「推論チャレンジ」のサイト → ナレッジグラフの公開

➤ <https://github.com/KnowledgeGraphJapan/KGRC-RDF>

➡ SPARQLエンドポイント

➤ <http://lod.hozo.jp/repositories/kgc2019>

➡ SPARQLクエリのサンプル

➤ <https://github.com/KnowledgeGraphJapan/Challenge/blob/master/rdf/2019/SPARQLsample.md>

➡ 可視化ツール

➤ <http://knowledge-graph.jp/visualization/>

ナレッジグラフ利用技術の ハンズオン

- 0. RDF処理に利用可能な技術の概観
- 1. RDFによるナレッジグラフの表現
- 2. SPARQLクエリによるナレッジグラフの検索
- 3. Apache Jena によるRDFの処理
- 4. Apache Fuseki (RDFデータベース)の利用



▶ ナレッジグラフ(RDF)を扱うための技術・ツール

➤ RDF用検索言語: SPARQL

✓ SPARQLエンドポイント(検索用API)からWeb経由の検索が可能

➤ RDF用のライブラリ

✓ <https://github.com/KnowledgeGraphJapan/sparql-library-examples> にプログラム言語でのサンプルあり

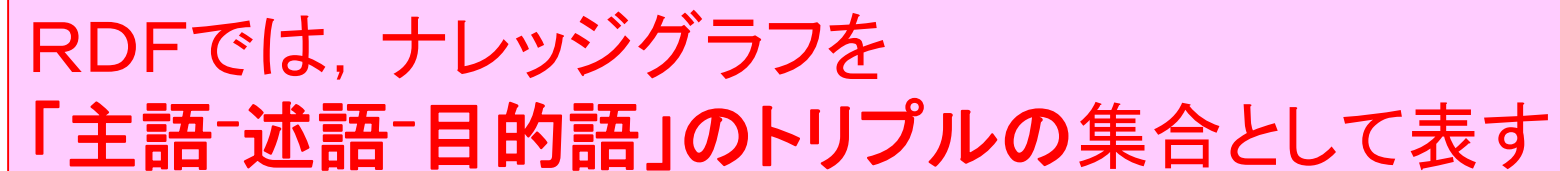
✓ Javaを使うなら, Apache Jenaがオススメ

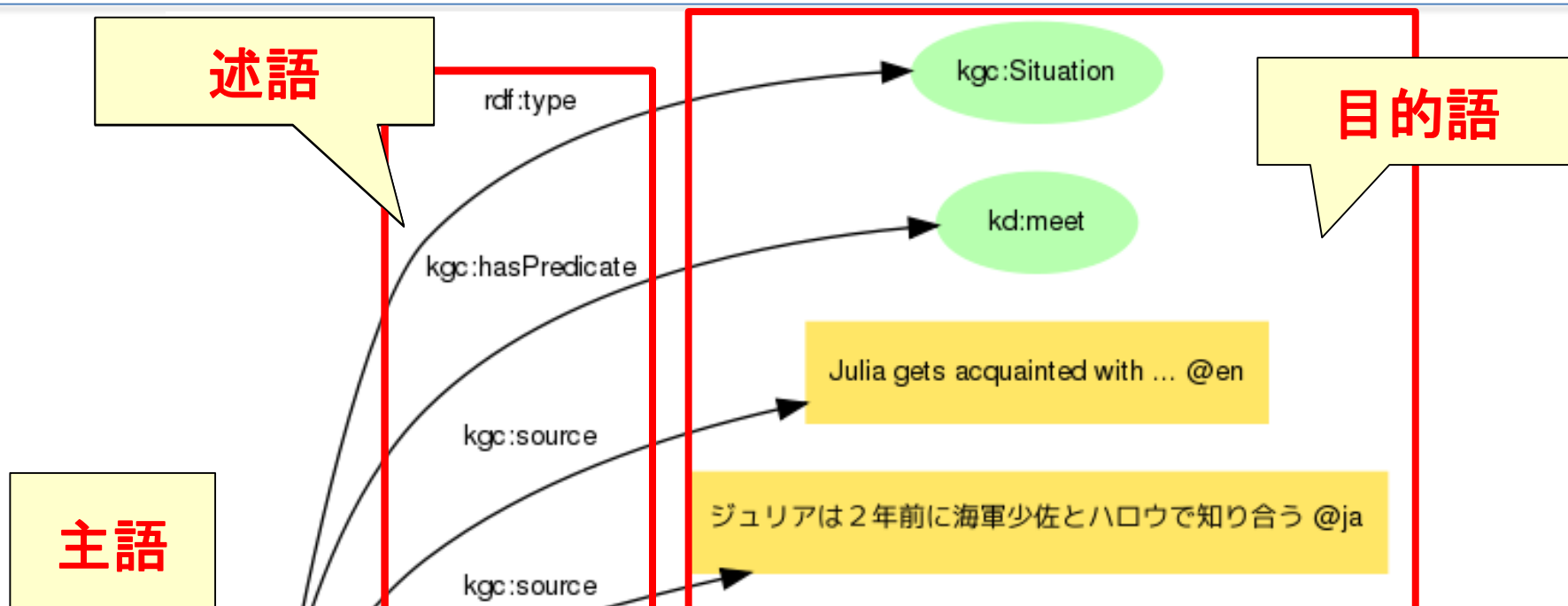
➤ OWL形式のファイルを開くには

✓ protégéなどのオントロジーエディタを使用(RDFファイルも開ける)
<https://protege.stanford.edu/>

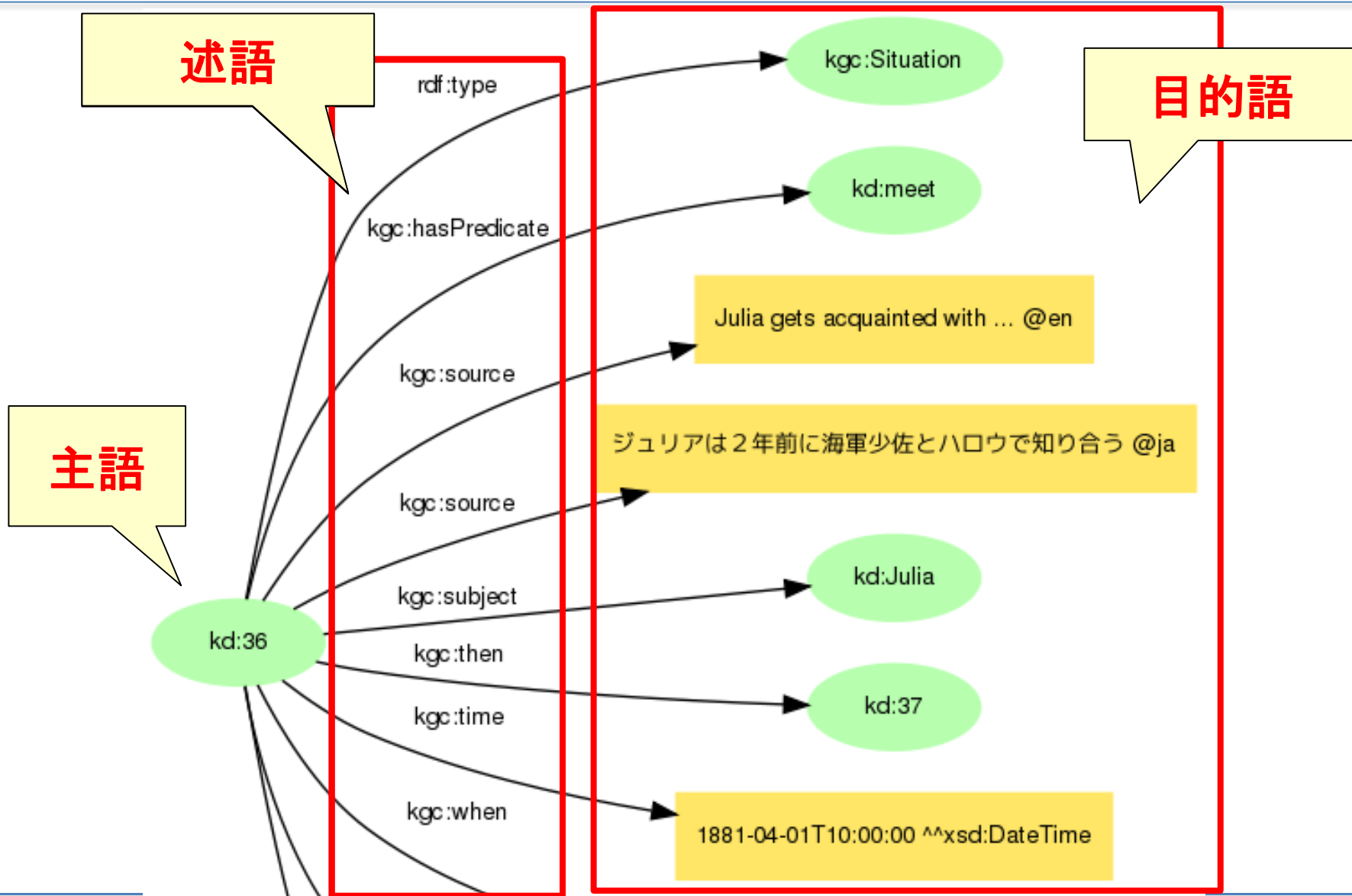
➤ RDFファイルをDBに格納して使用するには

✓ FusekiやVirtuosoなどのRDF-DBを使用

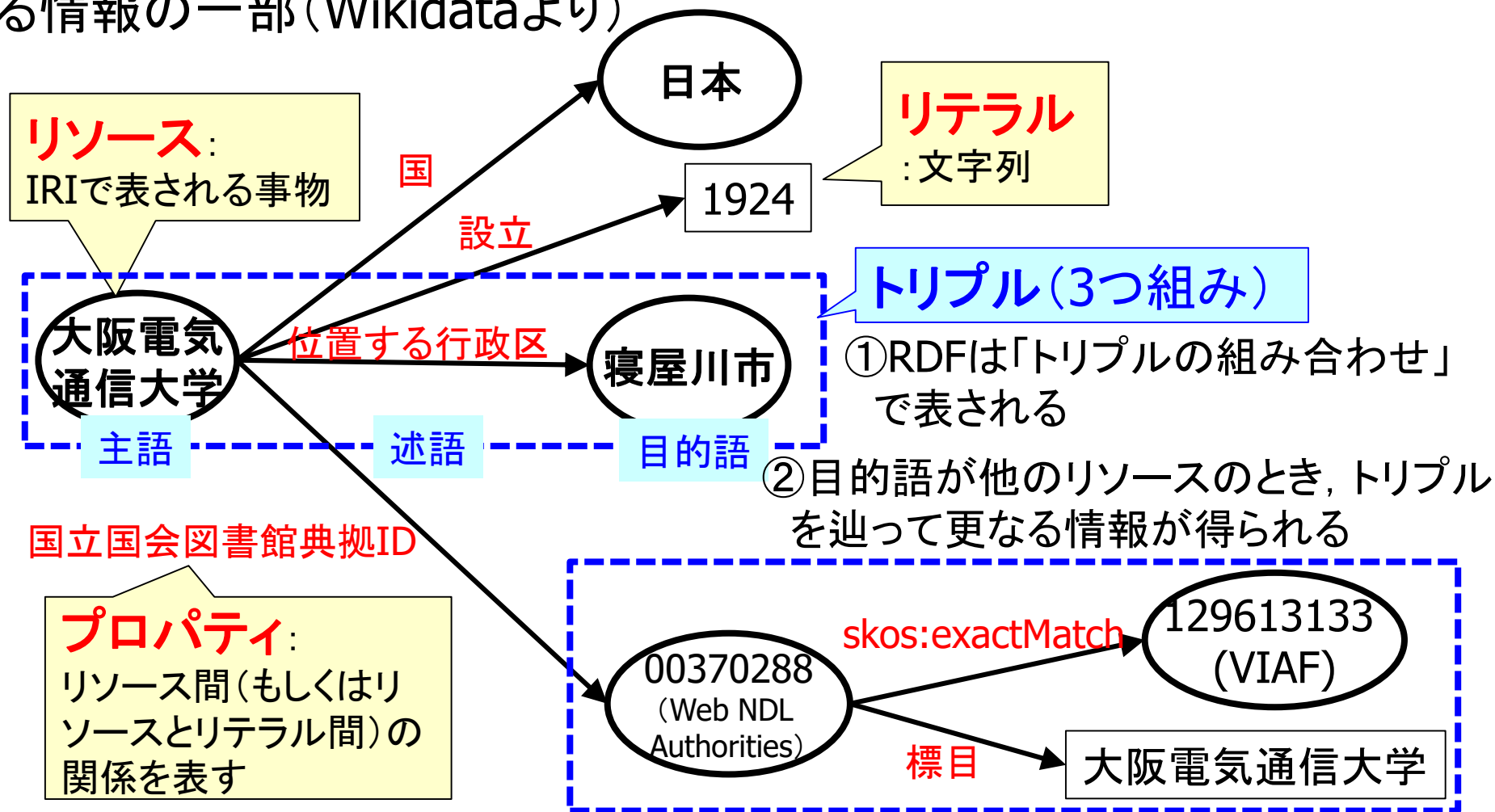




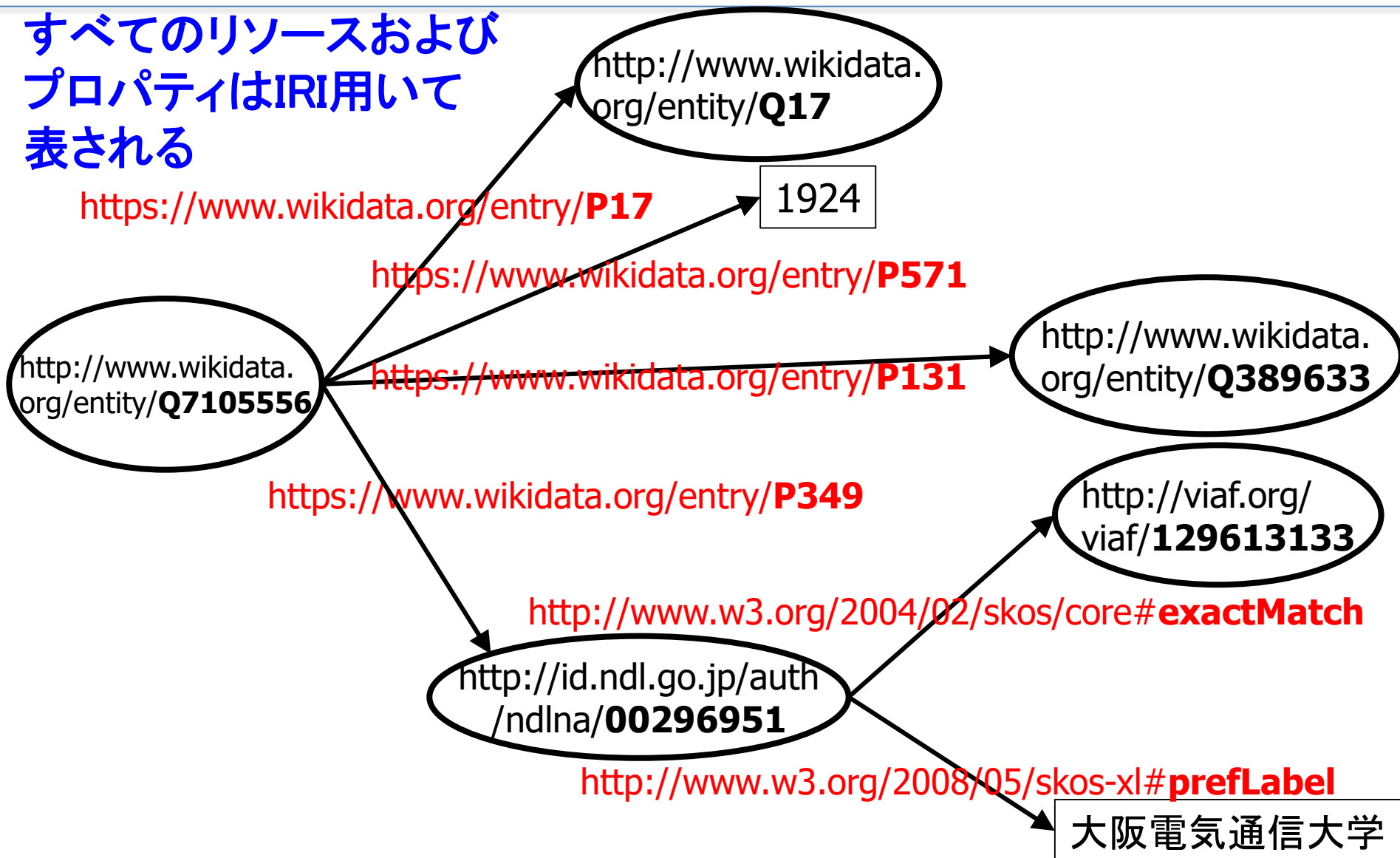
- 主語, 述語は, URL (IRI),
- 目的語は, URL (IRI), または文字列 (リテラルと呼ぶ) で表す
- 主語または目的語となるURLはリソースと呼ぶ
- リテラルには型や言語を指定できる



<http://www.wikidata.org/entity/Q7105556> というIRIから得られる情報の一部(Wikidataより)



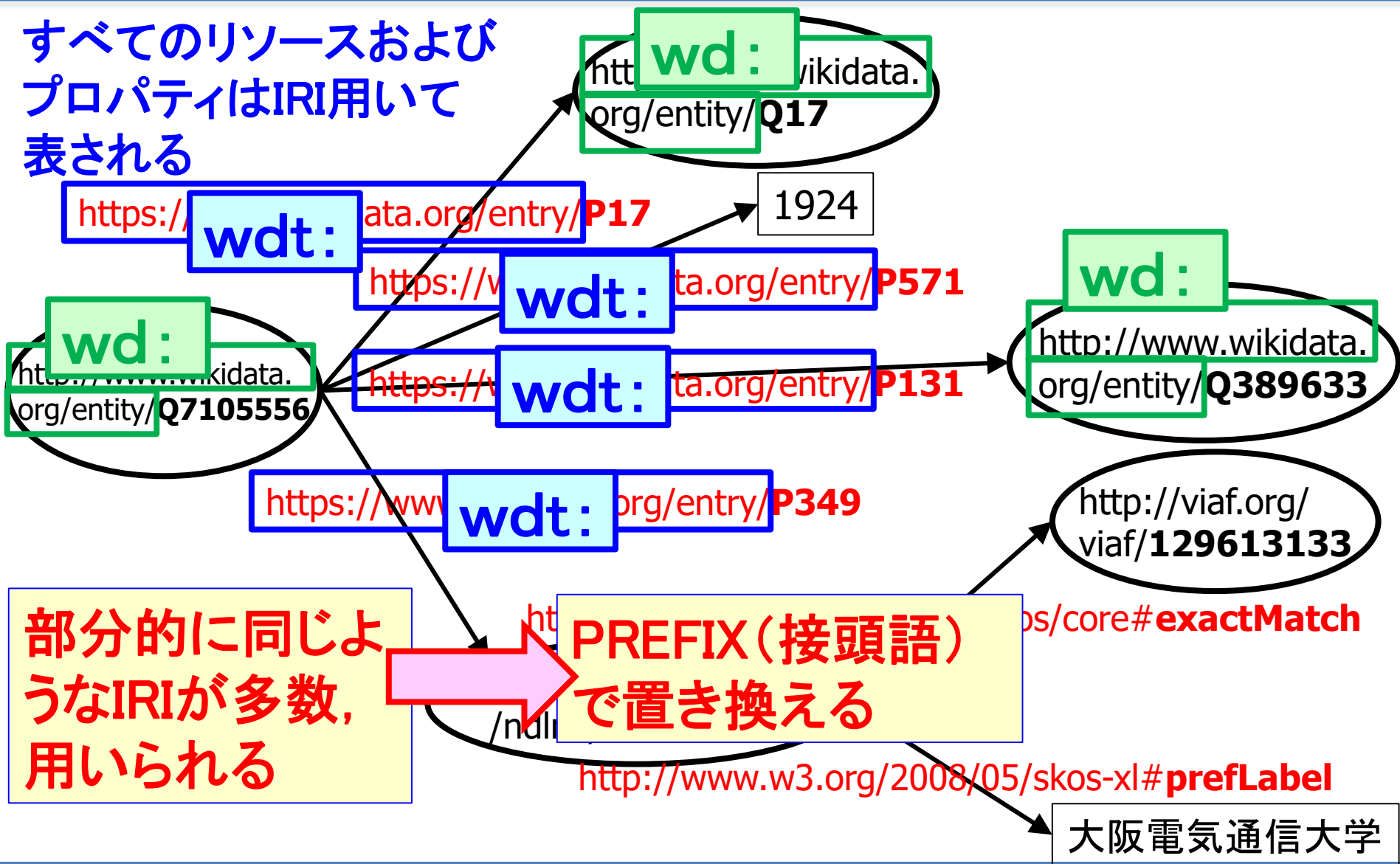
すべてのリソースおよび
プロパティはIRI用いて
表される

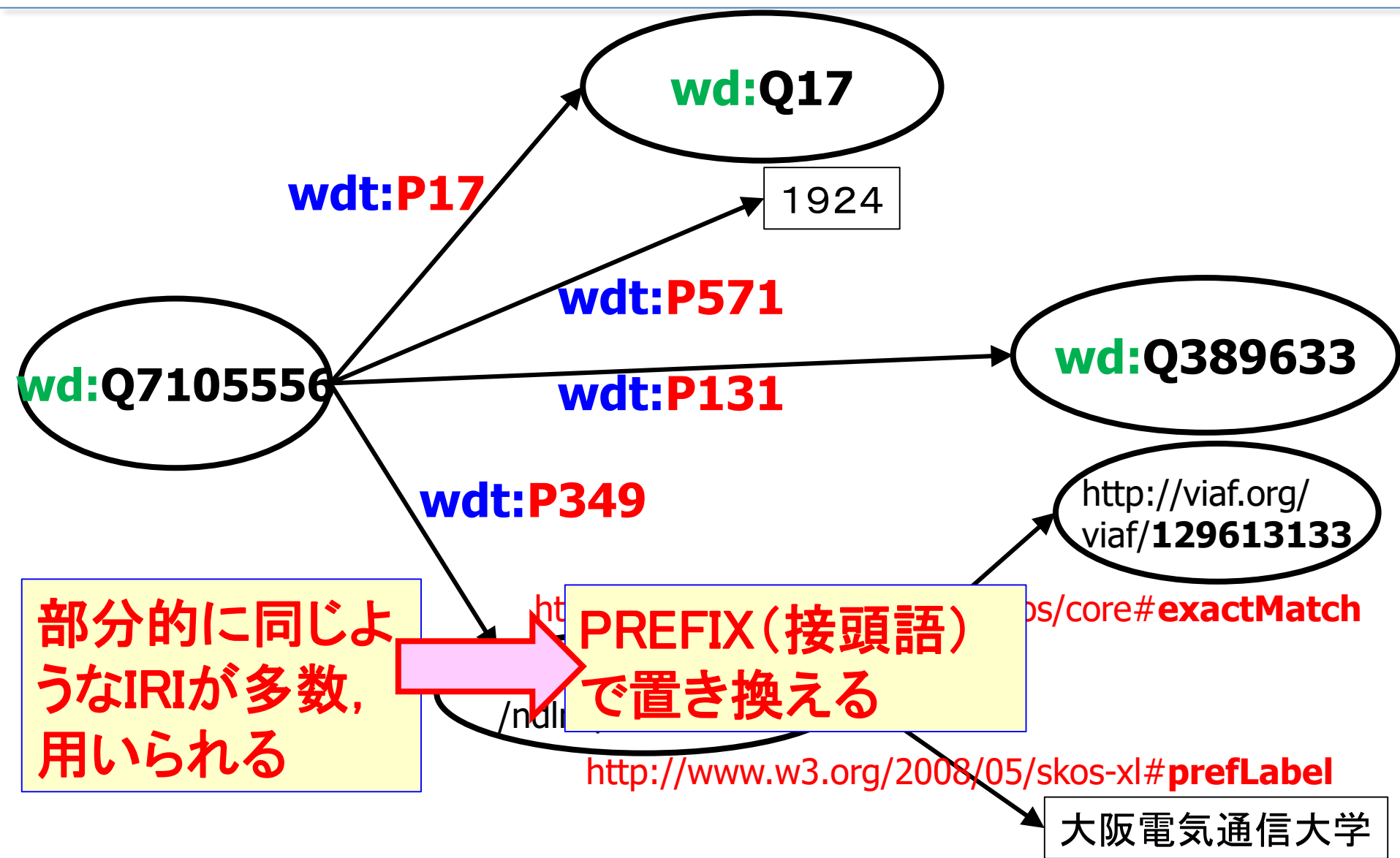




RDFの表現例

すべてのリソースおよび
プロパティはIRI用いて
表される







➡ 参照解決可能なhttp IRIs(URL,URI)を用いた公開

➤ IRIでデータにアクセスが可能

- ✓ 通常のWebページと同様に、データのURIを用いて「つながり」を辿ることが出来る

=システムによる処理(リンク解析等)が可能

➡ SPARQLエンドポイントの公開

➤ RDF用のクエリ言語SPARQLにより検索可能なAPIを公開

- ✓ クエリによるデータ検索・抽出が可能

➡ RDFファイルのダンプの公開

➤ 全データをダウンロードできる形で公開

- ✓ ダウンロードしたファイルをRDFパーサー, RDF-DBなどのツールを用いて処理可能

ナレッジグラフ利用技術の ハンズオン

0. RDF処理に利用可能な技術の概観

1. RDFによるナレッジグラフの表現

2. SPARQLクエリによるナレッジグラフの検索

3. Apache Jena によるRDFの処理

4. Apache Fuseki (RDFデータベース) の利用

■ SPARQL

- RDFデータに対するクエリ言語
- 「指定したグラフ構造」に一致するトリプルを検索する

■ 最も基本的な検索

```
select ?s ?p ?o
where {
  ?s ?p ?o .
}
LIMIT 100
```

← 返す要素

?x (x: 任意の文字列) は **変数** を表す

← 検索するグラフのパターン

「.」(ピリオド) を忘れない

↑ 取得する数の制限

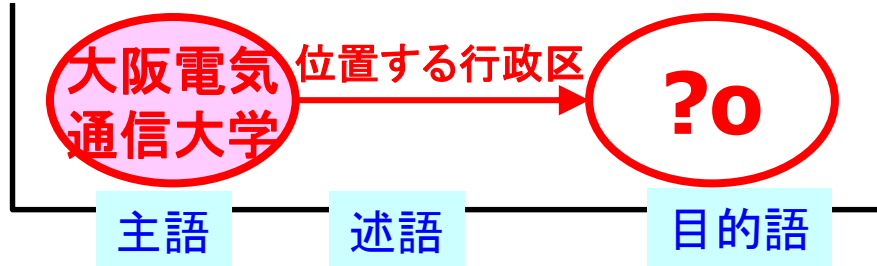
このパターンを変えることで、欲しいデータを取得する

検索例:

主語と述語を指定して「目的語」を取得

“<主語>の<述語>は何?”

検索するグラフパターン



SPARQLでの記述

<大阪電気通信大学> <位置する行政区> ?o

主語

述語

目的語

検索例1: 主語と述語を指定

- 例1)「大阪電気通信大学」(主語)の「位置する行政区」(述語)となる**目的語(?o)**を取得する

```
select ?o  
where {  
  wd:Q7105556 wdt:P131 ?o .  
}
```

大阪電気
通信大学
(主語)

位置する
行政区
(述語)

目的語
(変数)

検索例1: 主語と述語を指定

- 例1-1)「大阪電気通信大学」(主語)の「設立」(述語)となる**目的語(?o)**を取得する

```
select ?o  
where {  
  wd:Q7105556 wdt:P571 ?o .  
}
```

大阪電気
通信大学
(主語)

設立
(述語)

目的語
(変数)

※ 述語を変えるといろんな
目的語が取得できる

補足1: PREFIXの定義

- 例1)「大阪電気通信大学」(主語)の「位置する行政区」(述語)となる**目的語(?o)**を取得する

PREFIX wd: <http://www.wikidata.org/entity/>

PREFIX wdt: <http://www.wikidata.org/prop/direct/>

※ **wd:**や**wdt:**といった省略表現を使うためには、
本来は**PREFIXの定義が必要**

(WikidataのWebサービスでは省略可)

select **?o**

where {

wd:Q7105556 wdt:P131 ?o .

}

大阪電気
通信大学
(主語)

位置する
行政区
(述語)

目的語
(変数)

補足1: PREFIXを利用しない表現

■ 例1-1)をPREFIX(接頭語)を用いず書いた場合

```
select ?o
where {
  <http://www.wikidata.org/entity/Q7105556>
  <http://www.wikidata.org/prop/direct/P131> ?o .
}LIMIT 100
```

PREFIXによる省略表現

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
select ?o
where {
  wd:Q7105556 wdt:P131 ?o .
}LIMIT 100
```

PREFIXの定義

省略表現に用いる文字列は任意に設定できるが、できるだけ慣習的に利用されるものにあわせるとよい。

■ RDF一般のもの

- **rdfs:** <<http://www.w3.org/2000/01/rdf-schema#>>
RDFスキーマ(基本的な語彙定義)
- **schema:** <<http://schema.org/>>
Webのメタデータに記述される語彙
- **skos:** <<http://www.w3.org/2004/02/skos/core#>>
Web上でのシソーラス, 用語集などに用いられる語彙

■ Wikidataで使われるもの

- **wd:** <<http://www.wikidata.org/entity/>>
エンティティ(もの, コト, データ)
- **wdt:** <<http://www.wikidata.org/prop/direct/>>
プロパティ(述語/関係) ※主にSPARQL検索用の直接関係

■ PREFIXの検索サービス

- <https://prefix.cc/>

WikidataのRDFでは, 詳細情報を記述するため, 同じ内容のプロパティが3種類記述されているが, 今回は**wdt:**を使う.

- 補足例) 「大阪電気通信大学」のラベルとなる目的語(?o)を取得

```
PREFIX wd: <http://www.wikidata.org/entity/>
```

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

```
select distinct ?o
```

```
where {
```

```
  wd:Q7105556 rdfs:label ?o .
```

```
}LIMIT 100
```

**RDFで一般に
「ラベル」を表すプロパティ(述語)**

実行例 <https://w.wiki/646>

- 補足例) 「大阪電気通信大学」のラベルとなる目的語(?o)を取得
 - 「言語の種別」を合わせて取得

PREFIX wd: <http://www.wikidata.org/entity/>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

「言語種別」を取得する関数

結果を別の変数に代入

```
select ?o (lang(?o) AS ?ln)
```

```
where {
```

```
  wd:Q7105556 rdfs:label ?o .
```

```
}
```

実行結果 <https://w.wiki/648>

- 補足例) 「大阪電気通信大学」のラベルとなる目的語(?o)を取得
 - 「言語の種別=日本語(ja)」をのみ

```
PREFIX wd: <http://www.wikidata.org/entity/>
```

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

```
select ?o (lang(?o) AS ?ln)
```

```
where {
```

```
  wd:Q7105556 rdfs:label ?o .
```

```
  FILTER (lang(?o) = "ja") .
```

```
}
```

絞り込みの条件を記述(様々な条件記述できる)

実行結果 <https://w.wiki/64A>

➡ SPARQLクエリのサンプル

➡ <https://github.com/KnowledgeGraphJapan/Challenge/blob/master/rdf/2019/SPARQLs/ample.md>

➡ クエリを見ながら解説予定

ナレッジグラフ利用技術の ハンズオン

0. RDF処理に利用可能な技術の概観

1. RDFによるナレッジグラフの表現

2. SPARQLクエリによるナレッジグラフの検索

3. Apache Jena によるRDFの処理

4. Apache Fuseki (RDFデータベース) の利用



Javaの開発環境(Eclipse)のインストール

1. <http://mergedoc.osdn.jp/>

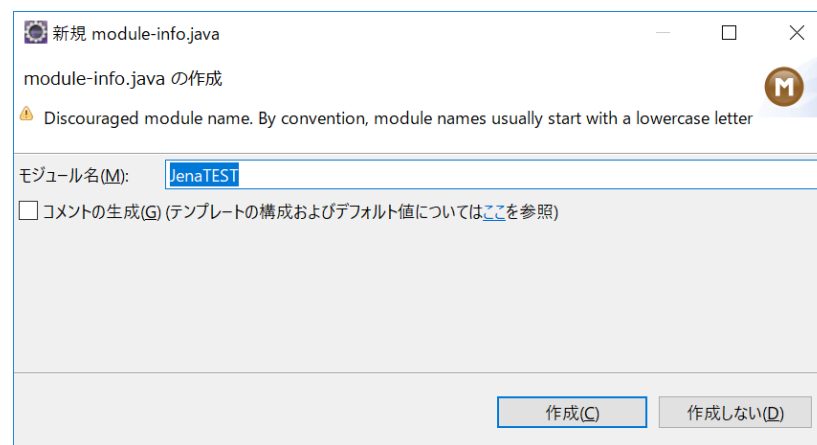
から「Pleiades All in One ダウンロード」の
「Eclipse 2019-09」を選択

2. JavaのFull EditonのZIPファイルをダウンロード

3. ZIPを解凍して、適当なフォルダにおけばインストール完了

※ZIPを解凍する際、Windowsの標準の「展開」で、
うまくいかないときは、別の解凍ソフト([7Zip](#)など)
を使えばOK

1. Eclipseを起動
2. workspace(作業フォルダ)を聞かれるのでOKする
3. ファイル>新規>Javaプロジェクトを実行
4. プロジェクト名(例: JenaTest)を入力してOK
5. プロジェクトが作成される
→srcがソースファイルを入れる
6. フォルダプロジェクト作成時に
表示される「モジュール作成」
については「作成しない」
を選択





1. ファイル＞新規＞クラスを実行
2. 名前を「HelloWorld」にして,
「public static void main (String [] args) 」にチェックを入れて「完了」
3. クラス「HelloWorld.java」が作成される
4. 作成されたクラスに以下の1行を追加して, 保存

```
public class HelloWorld {  
    public static void main(String[] args) {  
        // TODO 自動生成されたメソッド・スタブ  
        System.out.println("Hello World!");  
    }  
}
```



1. 「HelloWorld.java」を選択して,
右クリック>実行>Javaアプリケーション
を選択
2. プログラムが実行され, 結果がコンソールに
表示される

1. <https://jena.apache.org/> からダウンロード
apache-jena-3.13.0.zip を選択してダウンロード
2. 解凍してできる「lib」フォルダをプロジェクトにコピー
3. srcを選択して右クリック>ビルド・パス>ビルド・パスの構成
4. ライブラリ>JARの追加で「lib」フォルダ内のJARファイルを選択・追加

備考

- ➡ JenaはRDFを処理するためのライブラリ
- ➡ 使い方の詳細は, Java DOCを参照

➤ <https://jena.apache.org/documentation/javadoc/jena/>



1. <https://github.com/koujikozaiki/JenaSample>
からサンプルプログラムをダウンロード
→Clone or download > Download ZIP
2. ZIPファイルを展開し, Psrcフォルダの中身を自分のプロジェクトにコピーする
→エクスプローラーで選択>コピーの後,
Eclipseのパッケージエクスプローラのsrcに
「貼り付け」
※以下, プロジェクトへのコピーは同様の操作で行う
3. inputフォルダを自分のプロジェクトにコピーする
4. 自分のプロジェクトにoutputというフォルダを作る



それぞれのプログラムを選択し、
右クリック>実行>Javaアプリケーション
で実行してみる

➡ **readRDF.java**

RDFファイルを読み込み、別の形式で保存する。

➡ **searchRDF.java**

読み込んだRDFに対して、検索を行う

➡ **searchRDFusingSPARQL.java**

読み込んだRDFに対して、SPARQLで検索を行う。

➡ **searchRDFfromEndpoint.java**

SPARQLエンドポイントに対してSPARQLで検索する

ナレッジグラフ利用技術の ハンズオン

0. RDF処理に利用可能な技術の概観

1. RDFによるナレッジグラフの表現

2. SPARQLクエリによるナレッジグラフの検索

3. Apache Jena によるRDFの処理

4. Apache Fuseki (RDFデータベース) の利用

▶ Apache Fuseki(RDFデータベース)のインストール

➤ 下記の記事を参考にインストールしてください。

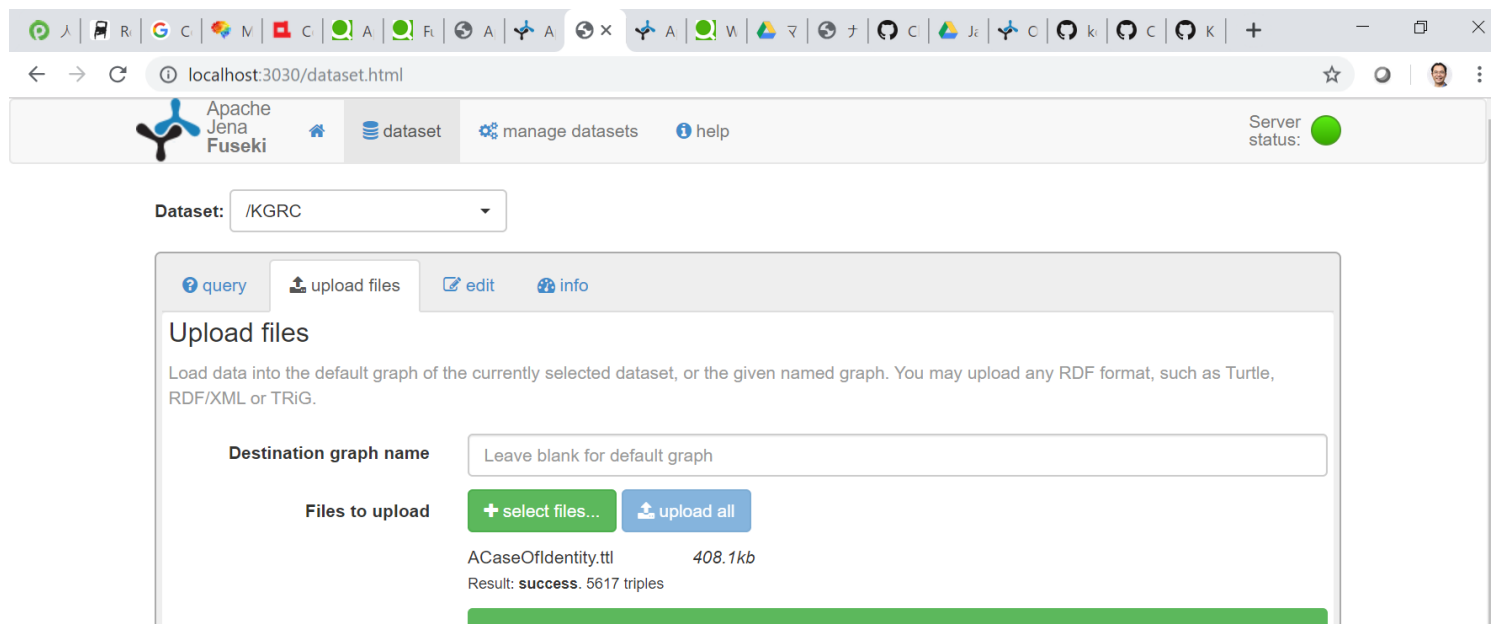
- ✓ [Fusekiをダウンロードして立ち上げるまで。Windows 10編](#)
- ✓ [Fusekiをダウンロードして立ち上げるまで。Mac OS X編](#)
- ✓ apache-jena-fuseki-3.13.1.zip で動作確認しました(Windows10)

▶ 推論チャレンジのナレッジグラフ(RDF)をダウンロード

- <https://github.com/KnowledgeGraphJapan/KGRC-RDF>からダウンロード(第1回, 第2回分を含む)
- 複数のファイル形式でダウンロードできるが, RDFファイルとしてはTurtle形式(拡張子は.ttl)の利用を

➡ FusekiにRDFファイルをアップロード

- Fusekiを起動した状態で<http://localhost:3030/>にアクセス
- manage datasets > add new dataset でDBを作成
 - ✓ Fusekiの再起動後もDBを残したいときは**Persistent**を選択する
- 作成したdatasetを選択し, Upload filesでアップロード



➡ queryを選択すると、ブラウザ上でSPARQL検索ができる

The screenshot shows the Apache Jena Fuseki web interface in a browser. The address bar shows 'localhost:3030/dataset.html'. The page has a header with the Apache Jena Fuseki logo and navigation links: 'dataset', 'manage datasets', and 'help'. A 'Server status' indicator is in the top right. Below the header, there's a 'Dataset:' dropdown menu set to '/KGRC'. A yellow callout box points to the 'info' tab, which is selected among 'query', 'upload files', 'edit', and 'info'. The 'info' tab displays various information about the dataset, including 'EXAMPLE QUERIES' (with buttons for 'Selection of triples' and 'Selection of classes'), 'PREFIXES' (with buttons for 'rdf', 'rdfs', 'owl', 'xsd', and a plus icon), and 'SPARQL ENDPOINT' (set to '/KGRC/query'). There are also dropdowns for 'CONTENT TYPE (SELECT)' and 'CONTENT TYPE (GRAPH)' (set to 'Turtle'). At the bottom, a code editor shows a SPARQL query: 'SELECT ?subject ?predicate ?object WHERE { ?subject ?predicate ?object }'. A yellow callout box points to the 'query' tab, stating that selecting it allows for SPARQL search in the browser. Another yellow callout box points to the 'info' tab, stating that selecting it allows for checking various information. A third yellow callout box points to the 'query' tab, stating that when searching from a program, the URL 'http://localhost:3030/dataset名/sparql' should be used.

Infoを選ぶと
諸々の情報を確認できる

プログラムから検索するときは
<http://localhost:3030/dataset名/sparql>
を使用する

➡ SPARQL仕様(W3Cのドキュメント)

- **SPARQL 1.1 Query Language**
<https://www.w3.org/TR/sparql11-query/>

➡ SPARQLの解説本

- **オープンデータ時代の標準Web API SPARQL**
<http://sparqlbook.jp/>

➡ SPARQL入門スライド(by古崎)

- **DBpedia Japaneseを例にした解説**
<https://www.slideshare.net/KoujiKozaki/4lod>
- **大阪市のオープンデータを例にした解説**
<https://www.slideshare.net/KoujiKozaki/apisparql>

➡ 解説記事

- **DBpediaを使った都道府県別ランキング**
<http://bit.ly/2oDPI0Q>
- **Wikidataを使った日本の政治家の出身大学ランキング**
<http://bit.ly/2PBt8fn>



<http://sparqlbook.jp/>より



➡ Jena のサイト <https://jena.apache.org/>

➡ Jenaのハンズオン資料

<https://github.com/KnowledgeGraphJapan/LODws2nd>

にある「ApacheJenaハンズオン.pdf」を見るとよい

→ハンズオンのソース

<https://github.com/KnowledgeGraphJapan/Apache-Jena-Sample-Programs>