Knowledge Graph Reasoning Challenge
ナレッジグラフ推論チャレンジ
生活行動における安心・安全を目指して
実社会版 2022

技術勉強会 2022/11/15 オンライン

ナレッジグラフを探索する SPARQL

江上周作

産業技術総合研究所
人工知能研究センター
データ知識融合研究チーム

SPARQL

ナレッジグラフを自由に探索できる標準Web API

SPARQLとは？

- RDFデータを検索するためのクエリ言語
  （RDBMSにおけるSQLに相当）
- Web標準化団体（W3C）により正式勧告
  - ver. 1.0（2008年1月にW3C勧告）
    - http://www.w3.org/TR/rdf-sparql-query/
    - http://www.asahi-net.or.jp/~ax2s-kmtn/internet/rdf/rdf-sparql-query.html
      （日本語訳）
  - ver. 1.1（2013年3月にW3C勧告）
    - http://www.w3.org/TR/sparql11-overview
    - http://www.asahi-net.or.jp/~ax2s-kmtn/internet/rdf/REC-sparql11-query-20130321.html
      （日本語訳）

グラフデータベース（トリプルストア）

- **RDF等のグラフデータを格納し，SPARQLによる操作を可能にするデータベース**
  - トリプルストア，RDFストアとも呼ばれる
- オープンソース，商用ともに様々なトリプルストアが存在

GraphDB
neo4j
OPENLINK VIRTUOSO UNIVERSAL SERVER
TigerGraph
aws Amazon Neptune
RDFOX
IBM Graph™
KATANA GRAPH
JanusGraph
ORACLE DATABASE 12c
MarkLogic®
blazegraph™
AllegroGraph Franz Inc.
Stardog

詳しくはこちら https://en.wikipedia.org/wiki/Comparison_of_triplestores

SPARQL基本パターン

```
SELECT  ?s ?p ?o
WHERE {
        ?s ?p ?o .
}
```

SELECT句

以降のWHERE句内に記述するトリプルパターンから結果として返す変数を直後で指定

```
SELECT  ?s ?p ?o

WHERE {

        ?s ?p ?o .

}
```

変数

変数は先頭に「?」をつける。
SELECT句の直後に指定した変数を結果として返す

```
SELECT  ?s ?p ?o

WHERE {

        ?s ?p ?o .

}
```

WHERE句

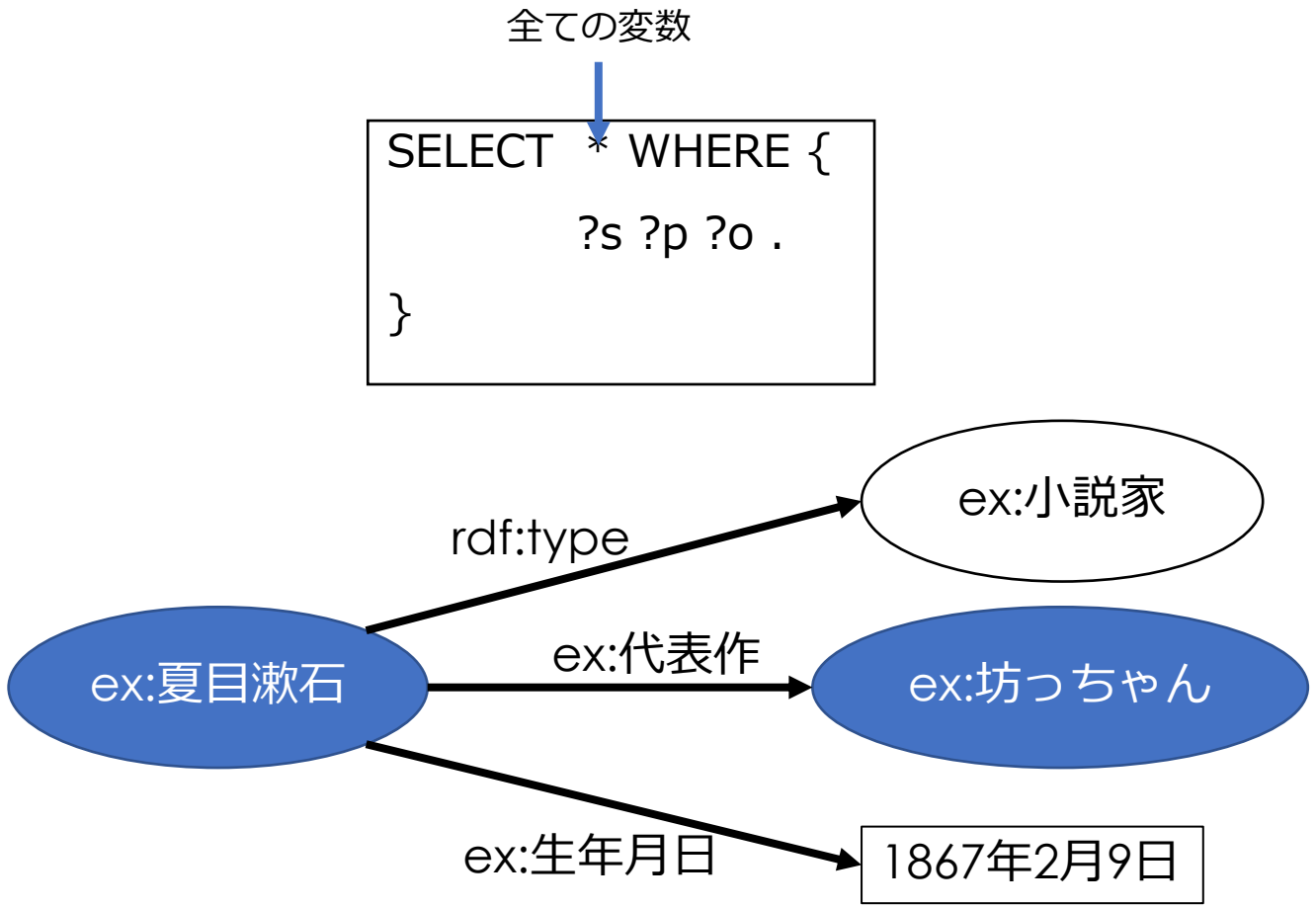
必ず {} で括る。
{} 内に検索するトリプルパターンを記述する.
{} 内のトリプルはTurtleと記法が同じ.

```
SELECT  ?s ?p ?o

WHERE {

        ?s ?p ?o .

}
```

この場合，主語・述語・目的語が全て変数なので，全てのトリプルパターンを検索している

検索イメージ

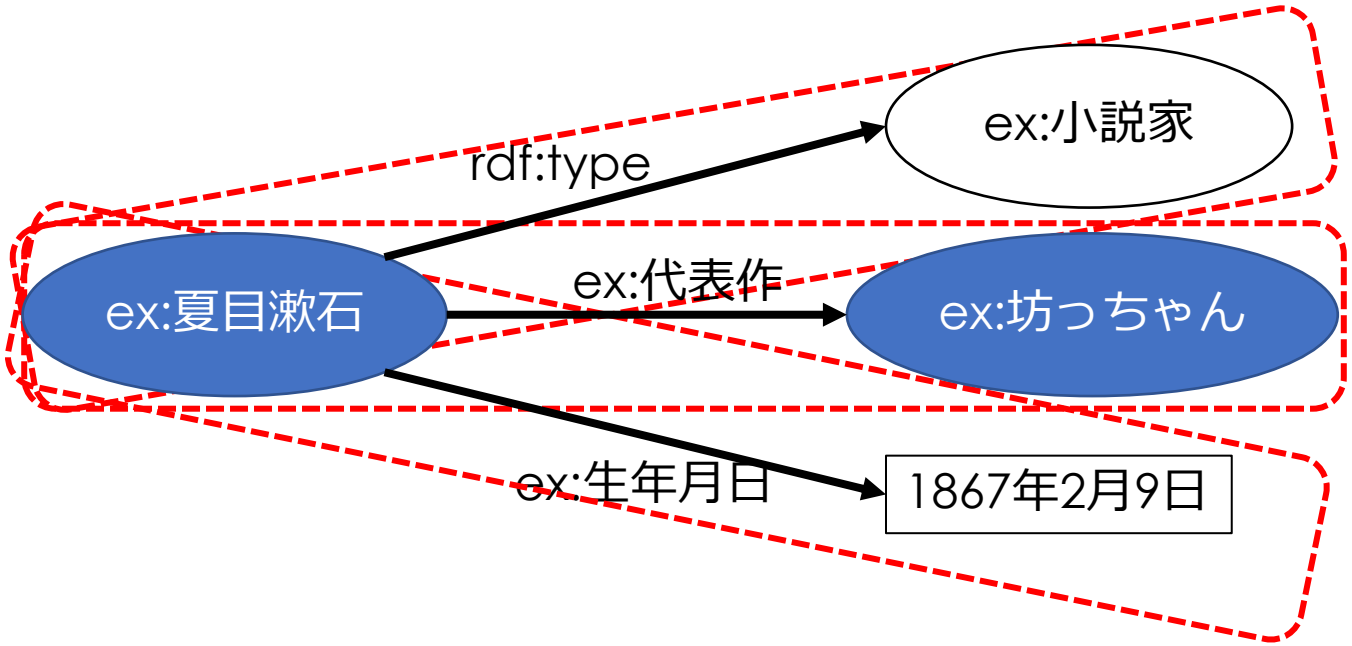
- 下図のグラフに対して次のSPARQLを実行する場合



検索イメージ

- WHERE句内のトリプルパターンに赤枠部分が該当

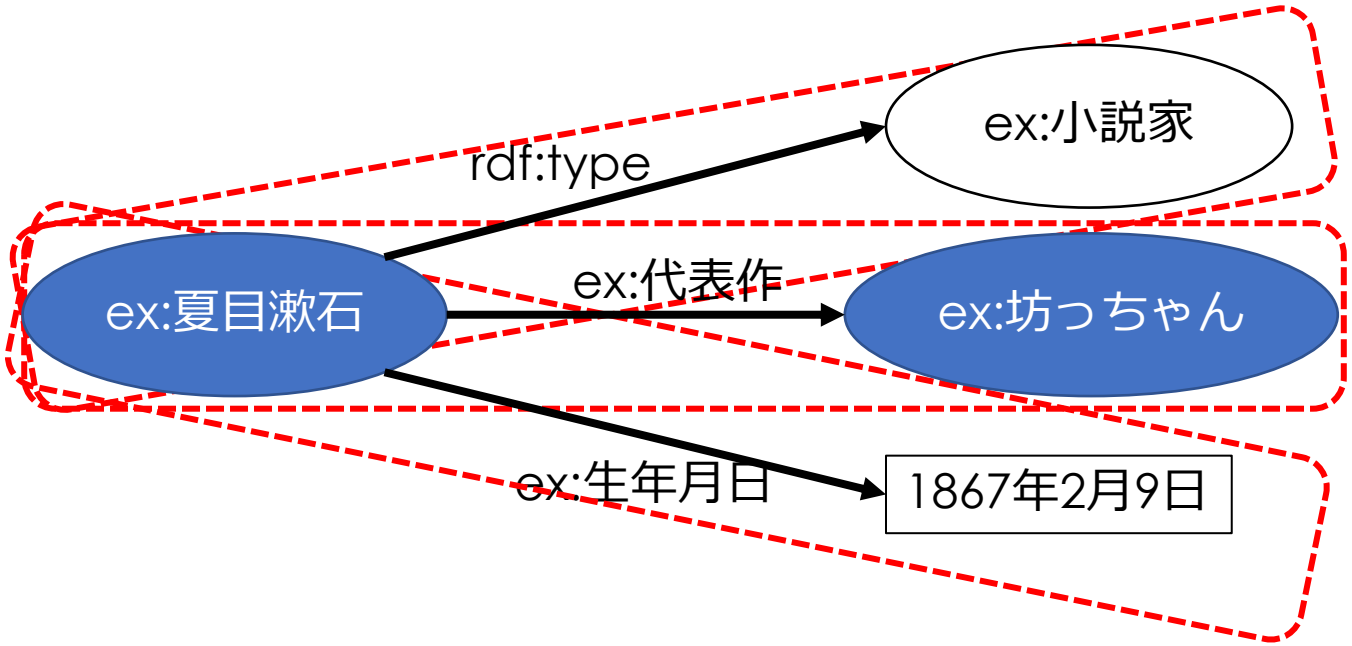
```
SELECT  * WHERE {
        ?s ?p ?o .
}
```



検索イメージ

- WHERE句内のトリプルパターンに赤枠部分が該当

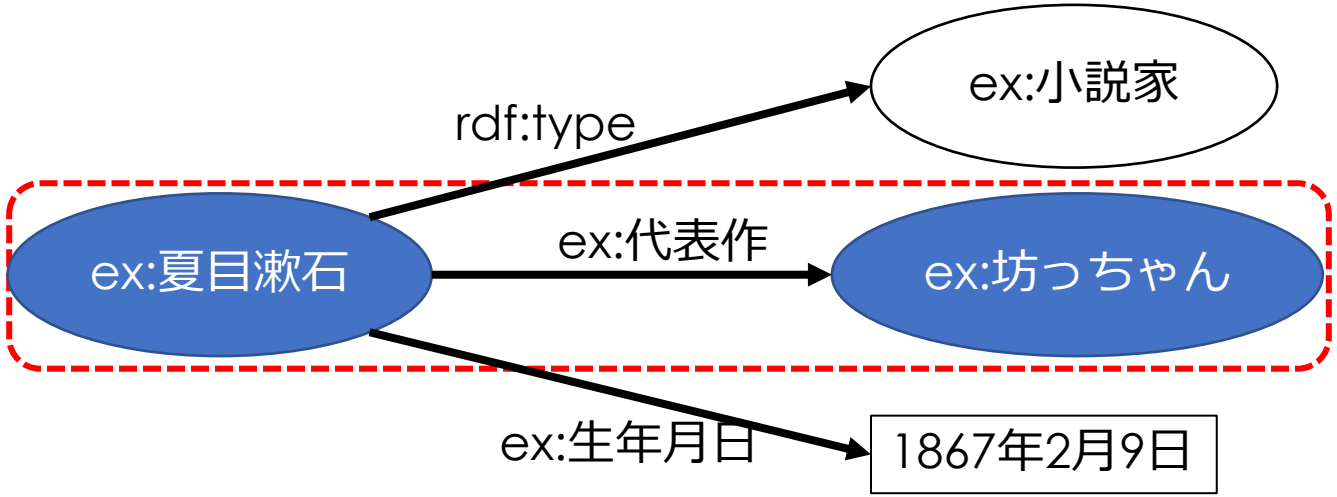
```
SELECT  * WHERE {
        ex:夏目漱石 ?p ?o .
}
```



検索イメージ

- WHERE句内のトリプルパターンに赤枠部分が該当

```
SELECT  * WHERE {
        ?s ex:代表作 ?o .
}
```



FILTER句

トリプルパターンからさらに絞り込みを行う。
括弧内に条件を記述。

例）生年月日の値が”1867年2月9日”に完全一致するパターンを検索

```
SELECT  *
WHERE {
        ?s ex:生年月日 ?o .
        FILTER(?o = “1867年2月9日”)
}
```

対象とするトリプルパターンが多い場合、
文字列検索は遅くなるので注意

FILTER句の正規表現

変数の値から正規表現検索を行う。
FILTER regex(変数, 正規表現, 大/小文字区別)

例）代表作の値が”Harry Potter”で始まる文字列のトリプルを検索

```
SELECT  *

WHERE {

        ?s ex:代表作 ?o .

        FILTER regex(?o, “^Harry Potter”, “i”)

}
```

- ?oを対象
- ^は先頭から始まるの意味
- “i”は大文字小文字を区別しないというオプション値（区別する場合はいらない）

対象とするトリプルパターンが多い場合、
正規表現検索はとても遅くなるので注意

OPTIONAL句

指定したトリプルパターンがあれば追加で取得する.
{}内にトリプルパターンを記述.

例）基本は生年月日プロパティの値を取得するが、
代表作プロパティの値として何かしら持っている場合は追加で取得.

```
SELECT  *

WHERE {

        ?s ex:生年月日 ?o .

        OPTIONAL { ?s ex:代表作 ?daihyo . }

}
```

リソースごとに持つプロパティの種類が異なるため、
あれば取得するというゆるい表現でエラー回避

DESCRIBE

指定したリソースに関するトリプルを全て取得する.
リソースの被リンクについても取得する.
したがって，結果はRDF形式である

例）Thalesに関するトリプルを全て取得

```
DESCRIBE  <http://dbpedia.org/resource/Thales>
```

例）ラベルが”Thales”@enであるリソース(1件)に関するトリプルを全て取得

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

DESCRIBE  ?s

WHERE { ?s  rdfs:label  "タレス"@ja . } limit 1
```

ASK

クエリの解をtrue/falseで返す

例）あるリソースがラベル”Thales”@enを持つというトリプルが存在するか？

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

ASK  {

        ?s  rdfs:label  “Thales”@en .

}
```

CONSTRUCT

検索結果から新たにトリプルを生成する.

したがって，結果はRDFの形式である.

例）ミレトスで生まれ，ラベル”タレス”@jaを持つリソース(?s)について検索する.
検索結果から， Pythagorasが?sを知っているというトリプルを生成する.

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
CONSTRUCT  {
            dbr:Pythagoras  foaf:knows  ?s .
} WHERE {
             ?s  rdfs:label  “タレス"@ja ;
                        dbo:birthPlace  dbr:Miletus .
}
```

基本的な句

- SELECT句
  - 検索結果として取得する変数を記述する
- WHERE句
  - {} の中に検索条件（トリプルパターン）を記述する
- FILTER句
  - トリプルパターンを絞り込むための条件を指定する
    - 数値の範囲や文字列の正規表現なども可
- FROM句
  - RDFストア内から検索対象となるグラフIRIを指定
  - （RDFストア内に複数のRDFグラフが格納されており，それぞれのグラフにIRIが付与されている前提）

よく使う機能（一部）

- COUNT
  - カウントする
  - SELECT COUNT(*) WHERE {?s ?p ?o}
- ORDER BY
  - 検索結果の出力順序を指定（降順:DESC()，昇順:ASC()など）
  - WHERE {?s ?p ?o } ORDER BY DESC(?o)
- LIMIT
  - 検索結果取得件数の上限を指定
  - WHERE {?s ?p ?o} LIMIT 100
- OFFSET
  - 検索結果取得の開始位置を指定
  - WHERE {?s ?p ?o} OFFSET 1 LIMIT 100
- DISTINCT
  - 検索結果から重複を削除
  - SELECT DISTINCT * WHERE {?s ?p ?o}
- GROUP BY
  - 結果を任意の単位でグループ化して集計する
  - WHERE {?s ?p ?o} GROUP BY ?o

SPARQLエンドポイント

- RDFストアに対してSPARQLクエリを投げかける窓口
- たいていの場合URLのqueryパラメータ値にURLエンコードしたSPARQLクエリを与えることで，結果を得ることができるREST APIとして提供されている
  - http://ドメイン名/sparql?query=ここにSPARQL
  - 例：DBpediaからThalesが持つpropertyとその値の組を100件までJSONで取得
    - http://dbpedia.org/sparql?query=select+distinct+*+where+%7B%3Chttp%3A%2F%2Fdbpedia.org%2Fresource%2FThales%3E+%3Fp+%3Fo%7D+LIMIT+100&format=json

DBpedia

- WikipediaのInfobox情報をメインにRDF化して提供するLOD
  - トップページ：　http://dbpedia.org/
  - SPARQLエンドポイント：　http://dbpedia.org/sparql

Harry Potter

From Wikipedia, the free encyclopedia

This article is about the series of novels. For other uses, including related topics and derivative works, see Harry Potter (disambiguation). For the character in the series, see Harry Potter (character). For the film adaptations, see Harry Potter (film series). For the franchise as a whole, see Wizarding World.

Harry Potter is a series of fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc concerns Harry's struggle against Lord Voldemort, a dark wizard who intends to become immortal, overthrow the wizard governing body known as the Ministry of Magic and subjugate all wizards and Muggles (non-magical people).

Since the release of the first novel, *Harry Potter and the Philosopher's Stone*, on 26 June 1997, the books have found immense popularity, critical acclaim and commercial success worldwide. They have attracted a wide adult audience as well as younger readers and are often considered cornerstones of modern young adult literature.[2] As of February 2018, the books have sold more than 500 million copies worldwide, making them the best-selling book series in history, and have been translated into eighty languages.[3] The last four books consecutively set records as the fastest-selling books in history, with the final instalment selling roughly eleven million copies in the United States within twenty-four hours of its release.

The series was originally published in English by two major publishers, Bloomsbury in the United Kingdom and Scholastic Press in the United States. A play, *Harry Potter and the Cursed Child*, based on a story co-written by Rowling, premiered in London on 30 July 2016 at the Palace Theatre, and its script was published by

Harry Potter

Harry Potter

The Philosopher's Stone
(1997)
The Chamber of Secrets
(1998)
The Prisoner of Azkaban
(1999)
The Goblet of Fire
(2000)
The Order of the Phoenix
(2003)
The Half-Blood Prince
(2005)
The Deathly Hallows
(2007)

Author	J. K. Rowling
Country	United Kingdom
Language	English
Genre	Fantasy, drama, young adult fiction, mystery, thriller, Bildungsroman
Publisher	Bloomsbury Publishing (UK) Pottermore (e-books; all languages)

Infobox

YASGUI

- オープンになっている様々なRDFストアのSPARQLエンドポイントにクエリを投げることができる

https://yasgui.triply.cc/

（例１）FILTER, OPTIONAL

- エンジニアの名前（日本語）と,死没地(ある場合，英語)の組を100件まで表示

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?label ?dp_label WHERE {
  ?s rdf:type dbo:Engineer ; rdfs:label ?label .
  OPTIONAL {?s dbo:deathPlace ?dp . ?dp
rdfs:label ?dp_label . }
  FILTER(LANG(?label)="ja")
  FILTER(LANG(?dp_label)="en")
} limit 100
```

LANG(変数)：　変数が文字列の場合に，その言語型を取得する。jaは日本語

（例２）GROUP BY

- Personクラスのサブクラスごとにインスタンス数をカウントして降順表示

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?job (count(?s) AS ?cnt) WHERE {
  ?job rdfs:subClassOf <http://dbpedia.org/ontology/Person> .
  ?s rdf:type ?job .
} GROUP BY ?job
ORDER BY DESC(count(?s))
```

AS: カウントした値を変数に割当

変数?jobに入るであろうリソース（クラス）でグルーピング

?jobに入るであろうクラスのインスタンス数でカウントし、結果を降順表示する

SPARQL検索

- アクティビティの一覧を取得する

```
PREFIX ex: <http://example.org/virtualhome2kg/instance/>
PREFIX : <http://example.org/virtualhome2kg/ontology/>
select DISTINCT * where {
    ?activity :virtualHome ex:scene1 .
}
```

実行結果

- 「インターネットをブラウズする」というアクティビティ中のイベントとアクションを取得する

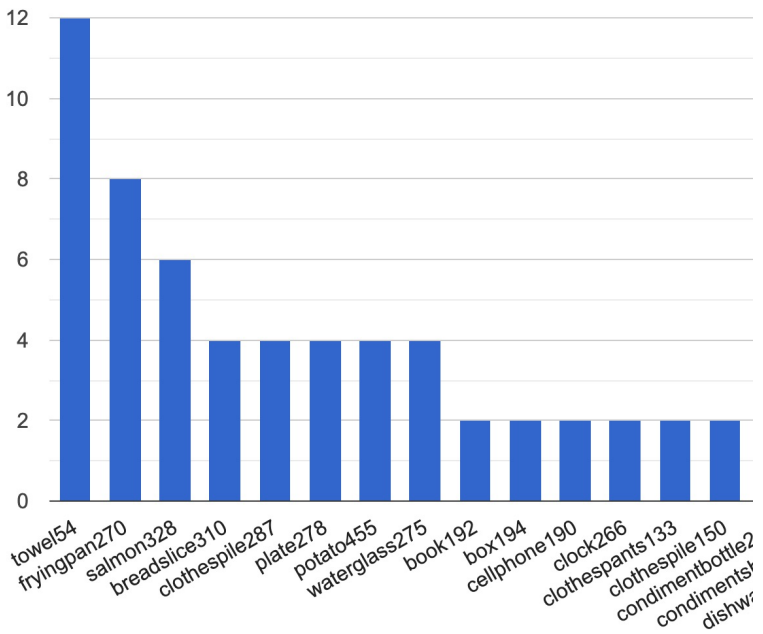
```
PREFIX ex: <http://example.org/virtualhome2kg/instance/>
PREFIX : <http://example.org/virtualhome2kg/ontology/>
select DISTINCT * where {
    ex:browse_internet_scene1 :hasEvent ?event .
    ?event :action ?action .
}
```

実行結果

SPARQL検索

- よく掴まれているオブジェクト

```
PREFIX ho: <http://www.owl-ontologies.com/VirtualHome.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://example.org/virtualhome2kg/ontology/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX ac: <http://example.org/virtualhome2kg/ontology/action/>
select ?name (count(?object) AS ?count) where {
        ?objectClass rdfs:subClassOf :Object .
    ?object a ?objectClass ;
            rdfs:label ?label ;
            dcterms:identifier ?id .
    ?event ho:object ?object .
    ?event :action ac:grab .
    BIND(concat(?label, ?id) AS ?name)
} group by ?object ?name order by desc(count(?object))
```



実行結果

SPARQL検索

- オブジェクトの高さ情報を追加する

```
PREFIX x3do: <https://www.web3d.org/specifications/X3dOntology4.0#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://example.org/virtualhome2kg/ontology/>
PREFIX ex: <http://example.org/virtualhome2kg/instance/>
CONSTRUCT {
    ?object :height ?height_node .
    ?height_node rdf:value ?size_y1 ;
           :unit :meter .
} WHERE {
        ?state1 :isStateOf ?object ; :bbox ?shape1 .
        ?shape1 x3do:bboxSize ?size1 .
        ?size1 rdf:rest ?size_y .
    ?size_y rdf:first ?size_y1 .
    BIND(REPLACE(STR(?object), STR(ex:) ,"") AS ?object_name)
    BIND(URI(CONCAT(STR(ex:),"height_", ?object_name)) AS ?height_node)
}
```

実行結果

Google Colab

- https://colab.research.google.com/drive/1HsNHaD2rLZBmDHYjb7l13uFI8EHUPSrO?usp=sharing
- ご自身のドライブにコピーして使ってください