

守屋 勇樹

情報・システム研究機構 ライフサイエンス統合データベースセンター(DBCLS)

<https://sparql-support.dbcls.jp/>



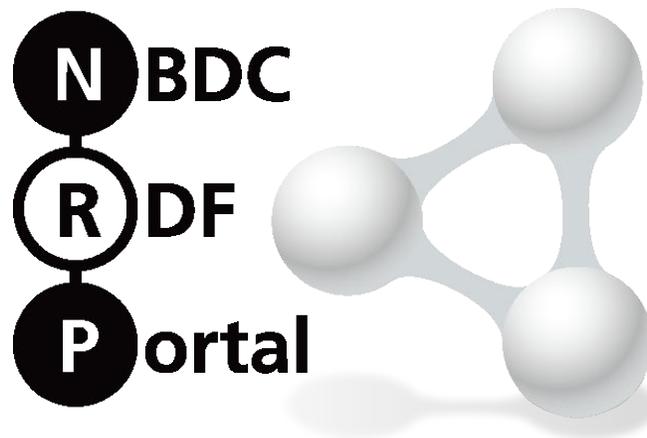
SPARQL support とは

Web ブラウザ上で動作する SPARQL クライアント・SPARQL クエリエディタ
クエリ記述を補助する機能を持っている

開発の背景



標準化 & 統合化



検索



データ抽出

これまで多くのライフサイエンス分野のデータが知識グラフ化(RDF
化)され、SPARQLエンドポイントが公開されている

問い合わせ言語 SPARQL を
使って検索

SPARQL support の UI

SPARQL クエリ

- RDF は '主語・述語・目的語' のトリプルでリソースの関係性を記述した知識グラフ
- 検索したいトリプルパターンを SPARQL クエリで記述する

```
1 # @endpoint https://db-dev.jpostdb.org/proxy/sparql
2 PREFIX : <http://rdf.jpostdb.org/entry/>↔
8 SELECT DISTINCT ?peptide ?uniprot ?seq ?begin
9 WHERE {
10   ?protein jpost:hasPeptideEvidence [
11     faldo:location/faldo:begin/faldo:position ?begin ;
12     jpost:hasPeptide ?peptide
13   ] ;
14     jpost:hasDatabaseSequence ?uniprot .
15   ?uniprot core:reviewed true .
16   ?peptide jpost:hasSequence/rdf:value ?seq .
17 }
```

基本はトリプルの集合なので
難しくは無いが
愚直にタイピングして
いくのは面倒



Run Query

結果

- Table, json 形式で検索結果が得られる

[8 bindings. -- 0.1 sec.] Download ▾

peptide	uniprot	seq	begin
:PEP791_1_1	uniprot:A2A5R2	"HLDVDLDRQSLSSIDR"	1509
:PEP791_1_7	uniprot:A2AJT4	"QRSPIALPVK"	209
:PEP791_1_10	uniprot:A2AJI0	"SASASPLTPCSAPR"	366
:PEP791_1_115	uniprot:A6H8H2	"STLSLALVR"	1319
:PEP791_1_132	uniprot:D3YXK2	"RSVVSFDK"	622

プロテオームデータベースから、実験で同定されたペプチド配列と、タンパク質上でのポジションを取得するクエリの例

SPARQL support の機能

詳細は https://sparql-support.dbcls.jp/sparql-support_j.html#sparql_support

自動補間：変数や関数、句、PREFIXなどをショートカットキー操作で補間

```
?h|          --> ?hoge|
r|           --> rdf:|
F|           --> FILTER (|)
PREFIX obo:| --> PREFIX obo: <http://purl.obolibrary.org/obo/>|
<id>|        --> <http://identifiers.org/>|
PREFIX #2|    --> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
               PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
               PREFIX dct: <http://purl.org/dc/terms/>|
COPY #1|     --> クエリ全体をコピー
```

Z Shellのような補間でクエリ作成のコストを削減

クエリ管理：複数のクエリをブラウザに保存して管理



エンドポイント指定：クエリに埋め込み

```
# @endpoint https://example.org/sparql
```



SPARQL のノーコード・ローコード化 (DBCLS の取り組み)

SPARQList

SPARQL と JavaScript で REST API 化 (API ユーザーはノーコード)

GO term count in a taxonomy (for Stanza 'group_comp' :enrich)

https://db-dev.jpostdb.org/rest/api/enrich_go_count

Run

tax: 9606

go: biological_process

https://db-dev.jpostdb.org/rest/api/enrich_go_count?tax=9606&go=biological_process

API code document

enrich_go_count.md (version: 2021-02-02T04:17:53.308Z)

Parameters

- tax: default: 9606
- go: default: biological_process

Endpoint

<https://db-dev.jpostdb.org/proxy/sparql>

get_go_count

```
PREFIX uniprot: <http://purl.uniprot.org/core/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?go (COUNT (DISTINCT ?up) AS ?count)
FROM <http://jpost.org/graph/uniprot>
FROM <https://jpost.org/graph/ontology>
WHERE {
  ?up uniprot:organism <http://purl.uniprot.org/taxonomy/{{tax}}>;
  uniprot:reviewed 1;
  uniprot:classifiedWith/rdfs:subClassOf+ ?go .
  FILTER (REGEX (STR (?go), "obo:GO_*"))
}
go <http://www.geneontology.org/formats/oboInWLN:080Namespaces "({go})"*xsd:string .
```

Endpoint browser

視覚的にトリプルパターンを選択して、SPARQL クエリを自動生成

Subgraph

```
# @endpoint https://db-dev.jpostdb.org/proxy/sparql
PREFIX jpo: <http://rdf.jpostdb.org/ontology/jpost.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX faldo: <http://biohackathon.org/resource/faldo#>
PREFIX p8: <http://rdf.jpostdb.org/entry/>
SELECT DISTINCT ?peptide ?aa_sequence ?begin
FROM <http://jpost.org/graph/database>
FROM <http://jpost.org/graph/ontology>
WHERE {
  p8:PRT810_1_Q9NYF8 jpo:hasPeptideEvidence [
    faldo:location/faldo:begin/faldo:position ?begin ;
    jpo:hasPeptide ?peptide
  ] .
  ?peptide jpo:hasSequence/rdf:value ?aa_sequence .
}
```

LIMIT 100

run

RDF-config

簡潔にしたデータモデル Yaml から SPARQL クエリや Schema 図を自動生成

```
- RefExEntry <http://purl.jp/bio/01/refex/RFX0016539731>:
  - a: refexo:RefExEntry
  - refexo:expValue:
    - ex_value: Ex value
  - dct:identifier:
    - refex_id: RFX0016539731
  - rdfs:seeAlso:
    - see_also: <http://identifiers.org/affy.probeset/224348_s_at>
  - refexo:refexSample:
    - sample: RefExSample
```

Grasp

GraphQL で RDF からデータ取得

```
query {
  RefExEntry {
    id
    ex_value
    see_also
    sample {
      taxonomy
      age
      stage
      description
      cell_type
      classid
      disease
      dev_site
    }
  }
}
```

SCHEMA

```
type Query {
  RefExEntry: RefExEntry!
}
type RefExEntry {
  id: String!
  ex_value: String!
  see_also: String!
  sample: RefExSample!
  taxonomy: String!
  age: String!
  stage: String!
  description: String!
  cell_type: String!
  classid: String!
  disease: String!
  dev_site: String!
}
```

いろいろな取り組みで SPARQL クエリ記述の削減を目指しているが、REST API の開発やメンテナンス、自動生成されたシンプルなクエリの編集などで、しばらくの間は、開発者はこれまで通り SPARQL クエリを記述していく必要がある



課題

RDF portal への集約で統合して検索できるようになった

- SPARQL クエリの複雑化
- 全ての更新箇所の把握が困難

データ更新したらクエリが正常に動作しなくなることも 'DBあるある' だが、デバッグが煩雑



クエリのどこに問題があるかを体系的に調査するには、本来は SPARQL 構文解析が必要

- 高コスト

複数行コメントアウト

SPARQLは1行コメントアウトのみ

初期から =b から =e までコメントアウトする機能を開発していた

```
1 # @endpoint https://db-dev.jpostdb.org/proxy/sparql
2 PREFIX jpost: <http://rdf.jpostdb.org/ontology/jpost.owl#>↔
5 SELECT DISTINCT ?peptide ?aa_sequence ?exact_position ?protein
6 FROM <http://jpost.org/graph/database>
7 FROM <http://jpost.org/graph/ontology>
8 WHERE {
9   # 1行コメント
10  =b 複数行コメント
11  ?protein jpost:hasPeptideEvidence [
12    faldo:location/faldo:begin/faldo:position ?exact_position ;
13    jpost:hasPeptide ?peptide
14  ] . =e
15  ?peptide jpost:hasSequence/rdf:value ?aa_sequence .
16 }
17 LIMIT 100
```

残った部分で検索することでデバッグ作業

```
1 ## endpoint https://data.allie.dbcls.jp/sparql
2 select (str(?year) as ?PubYear) (SUM(?PubMedCount) AS ?NumOfPMID)
3 where {
4   =b {
5     SELECT DISTINCT ?year ?PubMedCount
6     WHERE {
7       [] allie:publishedIn ?year .
8       BIND (0 as ?PubMedCount)
9     }
10  } UNION {
11    SELECT ?year (count (?year) as ?PubMedCount)
12    WHERE {
13      [] allie:hasShortFormRepresentationOf
14        / rdfs:label "CDS"@en ; =e
15      allie:hasLongFormRepresentationOf
16        / rdfs:label "clinical decision support"@en ;
17    =b allie:appearsIn
18        / allie:hasMemberOf
19        / allie:publishedIn ?year .
20    } group by ?year
21  }
22  {
23    SELECT (MIN(?year) as ?minyear) (MAX(?year) as ?maxyear)
24    WHERE {
25      [] allie:hasShortFormRepresentationOf
26        / rdfs:label "CDS"@en ;
27      allie:hasLongFormRepresentationOf
28        / rdfs:label "clinical decision support"@en ;
29      allie:appearsIn
30        / allie:hasMemberOf
31        / allie:publishedIn ?year .
32    }
33  }
34  FILTER(?year >= ?minyear && ?maxyear >= ?year) =e
35 }
```

もう少し楽に
デバッグ作業
できたら…

複雑なクエリでは複数箇所コメントアウトしたり、省略した主語や行末のセミコロンを書き換える必要がある

デバッグ検索機能

コメントアウトとは逆に、選んだ URI やトリプルパターンを対象に検索

- ‘# @debug true’ でモード指定
- 短縮 URI を右クリックで検索
- プロパティパスとか、ブロック単位を範囲選択して右クリックで検索

短縮 URI

```
1 # @endpoint https://sparql-support.dbcls.jp/proxy/sparql
2 # @debug true 3
3 PREFIX jpost: <http://rdf.jpostdb.org/ontology/jpost.owl#>
9 SELECT DISTINCT *
10 WHERE {
11 VALUES ?psm { :PSM1742_1_4862 }
12 ?pep jpost:hasPsm ?psm ;
13 jpost:hasSequence ?seq .
14 ?psm a jpost:PeptideSpectrumMatch ;
15 jpost:hasModification [
16 a jpost:AmbiguousModification ;
17 faldo:location [
18 a faldo:OneOfPossiblePosition ;
19 faldo:possiblePosition / ( faldo:position | rdfs:hoge ) ?position
]
```

Search as

- > Subject
- > Predicate
- > Object
- > Triple pattern

[3 bindings. -- 0.2 sec.]

p	o
rdf:type	jpost:PeptideSpectrumMatch
dcterms:identifier	"PSM1742_1_4862"
jpost:hasModification	nodeID://b637476

例) 選択した URI を主語としたトリプルがエンドポイントで取得できるか検証

プロパティパス

```
11 VALUES ?psm { :PSM1742_1_4862 }
12 ?pep jpost:hasPsm ?psm ;
13 jpost:hasSequence ?seq .
14 ?psm a jpost:PeptideSpectrumMatch ;
15 jpost:hasModification [
16 a jpost:AmbiguousModification ;
17 faldo:location [
18 a faldo:OneOfPossiblePosition ;
19 faldo:possiblePosition / ( faldo:position | rdfs:hoge ) ?position
20 ]
21 ] .
```

Search as

- > Subject
- > Predicate
- > Object
- > Triple pattern

トリプルパターン

```
14 ?psm a jpost:PeptideSpectrumMatch ;
15 jpost:hasModification [
16 a jpost:AmbiguousModification ;
17 faldo:location [
18 a faldo:OneOfPossiblePosition ;
19 faldo:possiblePosition / ( faldo:position | rdfs:hoge ) ?position
20 ]
21 ] .
22 }
23
24
25
```

Search as

- > Subject
- > Predicate
- > Object
- > Triple pattern

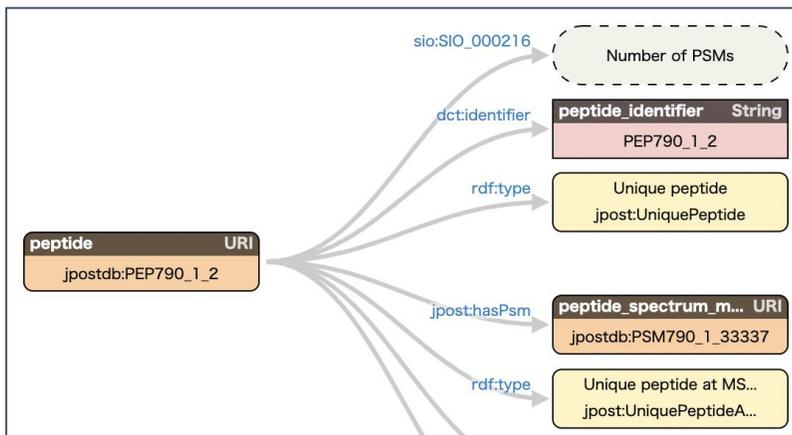
主語や blank node の閉じ括弧、最後のピリオドなどを補完して検索

その他の更新 (トーゴーの日2018以降に追加)

peptide	seq	begin
:PEP790_1_2	"TEGATVFGYQVMDPER"	127
:PEP790_1_3	"EAPQTLAQEVDR"	386
:PEP790_1_4	"PVNYHFAPR"	283
:PEP790_1_5	"LSQAFHAHLEETHGQIER"	40
:PEP790_1_6	"NWEEDDGEEYCTAAADLADAQAVCDK"	39



結果から Endpoint browser



結果から DESCRIBE

```
DESCRIBE <http://rdf.jpostdb.org/entry/PEP790_1_2>
s
:DS790_1
http://rdf.jpostdb.org/entry/PEP790_1_2
http://rdf.jpostdb.org/entry/PEP790_1_2
http://rdf.jpostdb.org/entry/PEP790_1_2
http://rdf.jpostdb.org/entry/PEP790_1_2
http://rdf.jpostdb.org/entry/PEP790_1_2
http://rdf.jpostdb.org/entry/PEP790_1_2
http://rdf.jpostdb.org/entry/PEP790_1_2
p
jpost:hasPeptide
rdf:type
rdf:type
rdf:type
dcterm:identifier
sio:SIO_000216
jpost:hasPsm
o
http://rdf.jpostdb.org/entry/PEP790_1_2
jpost:UniquePeptideAtMsLevel
jpost:UniquePeptide
jpost:Peptide
"PEP790_1_2"
_vb312452
:PSM790_1_33337
```

クリップボードへコピー

raw query query URL short URL auto run

クエリや、クエリにアクセスするための URL をクリップボードへコピーすることで、クエリの共有を促進

- 長いクエリを Slack や Docs に書かなくて良くなる

詳細は https://sparql-support.dbcls.jp/sparql-support_j.html#sparql_support

使い方

<https://sparql-support.dbcls.jp/>

アクセスするだけで利用可能

<https://github.com/moriya-dbcls/sparql-support>

github からダウンロードして、ローカルでも使える

<https://github.com/dbcls/sparql-proxy>

SPARQL-proxy に内包しているので、Docker で立てることも可能

募集

欲しい新機能などの相談は随時募集中