The UniProt SPARQL endpoint: 20 billion quads in production
Why provide a public SPARQL endpoint

• A 10 man wet laboratory can not afford:
  – to host their own database houses holding all or even a bit of all life science data.
  – not to have access, and use, existing life science information.
• Classical SQL can be provided on the web
  – Is not practical
  – No federation
  – No standards adherence
• Document centric REST is not enough
  – Swiss-Prot available as REST (over e-mail !!) since 1986
Your SPARQL query

Add common prefixes

1

Submit Query

Examples

1. Select all taxa from the UniProt taxonomy: (show)
2. Select all bacterial taxa, and their scientific name, from the UniProt taxonomy: (show)
3. Select all E-Coli K12 (including strains) UniProt entries and their amino acid sequence: (show)
4. Select the UniProt entry with the mnemonic 'A4_HUMAN': (show)
5. Select a mapping of UniProt to PDB entries using the UniProt cross-references to the PDB database: (show)
6. Select all cross-references to external databases of the category '3D structure databases' of UniProt entries that are classified with the keyword '3Fe-45': (show)
7. Select all UniProt entries, and their recommended protein name, that have a preferred gene name that contains the text 'DNA': (show)
8. Select the preferred gene name and disease annotation of all human UniProt entries that are known to be involved in a disease: (show)
9. Select all human UniProt entries with a sequence variant that leads to a 'loss of function': (show)
10. Select all human UniProt entries with a sequence variant that leads to a tyrosine to phenylalanine substitution: (show)
11. Select all UniProt entries with annotated (show)
12. Select all UniProt entries that were integrated on the 30th of November: (show)
13. Was any UniProt entry Integrated on the 9th of January: (show)
14. Construct new triples of the type 'HumanProtein' from all human UniProt entries: (show)
15. Select all triples that relate to the EMBL CDS entry AA089367: (show)
16. Select all triples that map to the taxon that

About

This SPARQL endpoint contains all UniProt data. It is free to access and supports the SPARQL 1.1 Standard.

There are 19,361,572,066 triples in this release (2015_03).

Documentation

The documentation about UniProt RDF is spread into 2 parts
1. Classes and predicates defined by the UniProt consortium
2. Statistics and diagrams

News

Regulation of translation initiation through folding | New proteomics mapping files | New FTP repository for reference proteomes
UniProt release 2015_03
vocabulary of human diseases | Changes to keywords
UniProt release 2015_02
Thalidomide, the pharmacological version of yin and yang | Cross-references to UniProt Proteomes | Cross-references to DEPPOD
UniProt release 2015_01

News archive
Node 1
64 cpu cores
256 GB ram
2.5 TB consumer SSD

Node 2
64 cpu cores
256 GB ram
2.5 TB consumer SSD

Load Balancer = Apache mod_balancer
19,361,572,066

Load Balancer = Apache mod_balancer

Node 1
- Tomcat + Sesame + UI
- Virtuoso 7.2 (+)

Node 2
- Tomcat + Sesame + UI
- Virtuoso 7.2 (+)
Dedicated machine for loading and testing

- Loading RDF data “solved” problem
  - 500,000 triples per second easy
    - that’s what our machine plus virtuoso 7.2
      - and some tricks does
    - 1,000,000 possible (gunzip limit on our machine)
      - nquads or rdf/xml
    - higher values needs parallel readers
      - or even lighter weight parsers
  - highest observed rate
    - 2.5 million per second on 1/4 exadata
      - could be pushed higher
Challenges as a public endpoint provider

• Query load unpredictable

• Simple data discovery queries are hard
  – 1 TB+ of DB files
  – e.g. from monitoring services

• Query timeouts not sufficient
  – aim for 100% utilisation
  – what can http reasonably support
  – we want to be able to answer hard questions
Queries per UniProt release
peak: 35 million per month
50 queries per second
Real users

Mix between hard analytics and super specific
Estimate somewhere between:
  300 - 2000 real humans per month
Really hard queries

```
SELECT (COUNT(DISTINCT(?iri) AS ?iriCount))
WHERE
{
    {?iri ?p ?o}
    UNION
    {?s ?iri ?o}
    UNION
    {?s ?p ?iri}
    FILTER(isIRI(?iri))
}
```
Counting all 3,897,109,089 IRI takes a while

- Via iSQL
  - 9 to 10 hours
  - if no other users
- SQL alternative

```sql
> SELECT COUNT(RI_ID) FROM RDF_IRI;
count INTEGER

3897109089
1 Rows. -- 1126860 msec.
```

- 18 minutes
  - Faster tricks?
Compilation wise unlikely to be found

• Are templates a good idea?
  – e.g. JVM has intrinsics
    • Long.bitCount()
      – in java
        ```java
        i = i - ((i >>> 1) & 0x5555555555555555L);
        i = (i & 0x3333333333333333L)
            + ((i >>> 2) & 0x3333333333333333L);
        i = (i + (i >>> 4)) & 0x0f0f0f0f0f0f0f0fL;
        i = i + (i >>> 8);
        i = i + (i >>> 16);
        i = i + (i >>> 32);
        return (int)i & 0x7f;
        ```
      – or 1 X86 instruction (in use since 1.6)
        » POPCNT
Template/Intrinsics based SPARQL compilation

- Recognising query template matches can be difficult
  - query normalisation?
Similar query

SELECT (COUNT(DISTINCT(?p) AS ?pc)
WHERE {?s ?p ?pc }

- Virtuoso
- Index only scan?
- GraphDB
- Information stored in predicate statistics that are key for optimiser
  - Can information be fetched from there?
Challenges

• **Virtuoso**
  – transitive queries
  – standards compliance

• **GraphDB**
  – analytical queries
  – complete store scans

• **Oracle 12c1**
  – configuration
  – global RDF tablespace
    • difficult to manage as a normal Oracle DB
Public monitoring key aid in quality assurance

- Public monitoring also hard
  - often lower uptime than what is being monitored
  - robots.txt
  - not enough community support
  - service description
    - not being parsed
  - HEAD last modified?
Key-Value orientated SPARQL endpoint anyone?

- assume 400 million named graphs
  - average 50 triples
  - max 5000 triples
  - get the whole named graph
- single IO operation

CONSTRUCT {}
FROM uniprot:P05067
WHERE {}