Goo
RDF/SPARQL support in BioPortal

Manuel Salvadores

https://github.com/ncbo/goo

Goo: Graph Oriented Objects for Ruby

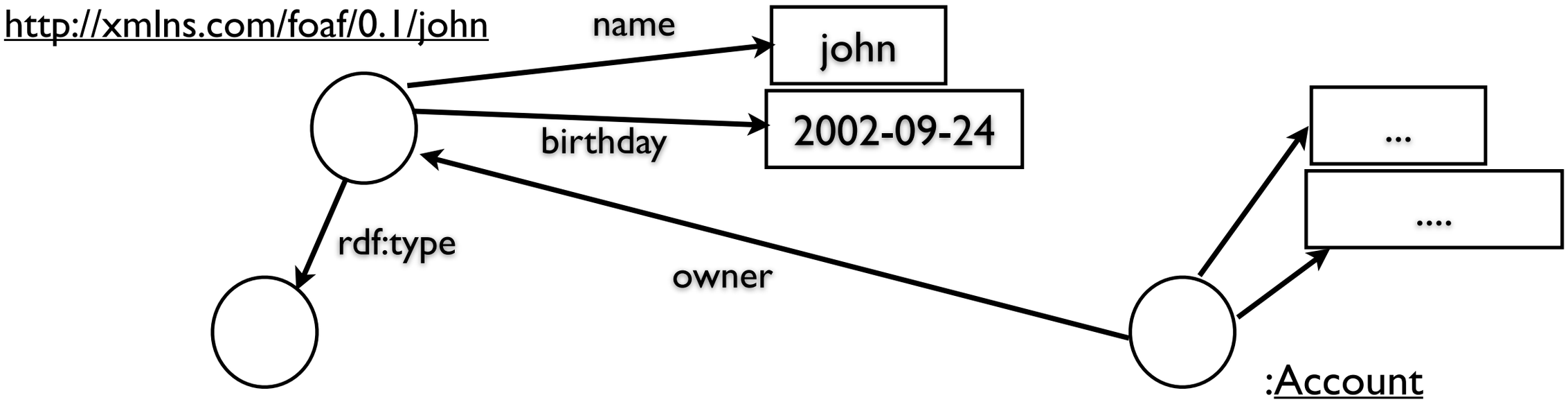
- Goo is a Ruby library that provides ORM-like capabilities to interact with RDF/SPARQL backends.

- Goo provides a DSL for defining schemas for objects and controls how they get validated, serialized, saved and retrieved from the SPARQL backend.

Goo: Driving Requirements

- **Abstraction**: Hide SPARQL/RDF complexity.
- **Scalability**: Queries must be design to use triple store strong points.
- **Flexibility**: Retrieve only what is needed.

Domain-specific Language (DSL)



http://xmlns.com/foaf/0.1/Person

```
class Person < Goo::Base::Resource
  model :person, namespace: :foaf, name_with: :name
  attribute :name, enforce: [:unique]
  attribute :birth_date, enforce: [:date_time], property: :birthday
  attribute :accounts, inverse: [ on: :user_account, property: :owner]
end
```

Check out README.md @

https://github.com/ncbo/goo

Core Goo objects:

resource.rb

settings.rb

where.rb

queries.rb

BioPortal Models

ontologies_linked_data/models

examples ...
groups, projects and ontology

Goo configuration in BioPortal is in ontologies_linked_data

Querying

1
```
john = User.find("john")
notes = Note.where(owner: john)
            .include(:content)
```

First get John and use that user object to match the Note graph.

2
```
notes  = Note.where(owner: [ :username "john"])
         .include(:content)
```

Query directly the Note graph to retrieve every note that has an owner attribute that points to a user with username John

---

3
```
john = User.find("john")
       .include(notes: [:content])
notes = john.notes
```

Access the Note graph through User. Find John and include his notes with their content.

4
```
notes = Note.where(owner: [ :username "john"])
        .include(notes: [:content])
        .page(1,100)
```

Same as (2) but using pagination

```
notes.each do |note|
   puts note.content
end
```

Just a plain Ruby loop over John's notes printing the content.

Goo's Query Strategy

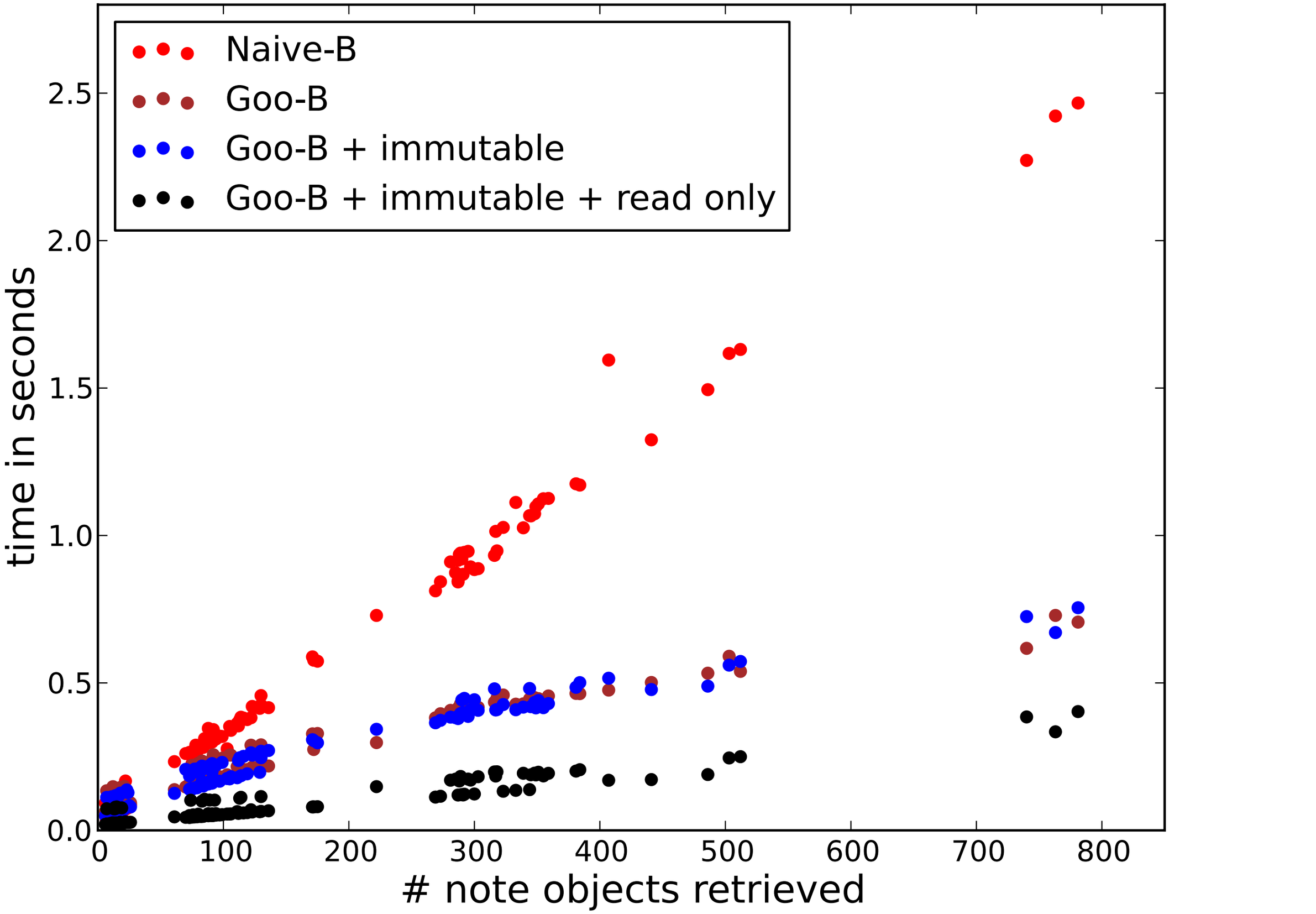
```
notes = Note.where(ontology: [:acronym "$ACR"])
        .include(:content)
        .include(owner: [:username, :email, roles: [:code])
        .include(ontology [:name])
```

1
```
SELECT ?note ?user ?content
FROM :Note
WHERE { ?note a :Note .
    ?note :owner ?user .
    ?note :content ?content .
    ?note :ontology ?ontology .
    ?ontology :acronym "$ACR" . }
```

2
```
SELECT ?user ?email ?username
FROM :User
WHERE { ?user a :User .
    ?user :name ?name .
    ?user :email ?email .
    FILTER (?user = <> || ?user = <> ...)
 }
```

3
```
SELECT ?role ?content
FROM :Role
WHERE { ?role a :Role .
    ?role :code ?code .
    FILTER (?role = <> || ?role = <> ...)
 }
```

4
```
SELECT ?note ?user ?content
FROM : Ontology
WHERE { ?ont a :Ontology .
    ?ont :name ?name .
    FILTER (?ont = <> || ?ont = <> ...)
 }
```


Naive-B
Goo-B
Goo-B + immutable
Goo-B + immutable + read only
time in seconds
note objects retrieved
0.0
0.5
1.0
1.5
2.0
2.5
0
100
200
300
400
500
600
700
800

Default Named Graphs

IRI type of the Model

<http://data.bioontology.org/metadata/OntologyFormat>
<http://data.bioontology.org/metadata/ProvisionalClass>
<http://data.bioontology.org/metadata/Review>
<http://data.bioontology.org/metadata/Group>
<http://data.bioontology.org/metadata/MappingCount>
<http://data.bioontology.org/metadata/Metrics>
<http://data.bioontology.org/metadata/Base>
<http://data.bioontology.org/metadata/SubmissionStatus>
<http://data.bioontology.org/metadata/OntologySubmission>
<http://data.bioontology.org/metadata/Role>
<http://data.bioontology.org/metadata/MappingProcess>
<http://data.bioontology.org/metadata/Reply>
<http://data.bioontology.org/metadata/Category>
<http://data.bioontology.org/metadata/Ontology>
<http://data.bioontology.org/metadata/Details>
<http://data.bioontology.org/metadata/Subscription>
<http://data.bioontology.org/metadata/User>
<http://data.bioontology.org/metadata/NotificationType>
<http://data.bioontology.org/metadata/Slice>
<http://data.bioontology.org/metadata/Contact>
<http://data.bioontology.org/metadata/Note>
<http://data.bioontology.org/metadata/Project>
<http://data.bioontology.org/metadata/RestBackupMapping>

Ontology Class Model

Collections

Standard Attributes

Unmapped property values

```
module LinkedData
  module Models
    class ClassAttributeNotLoaded < StandardError
    end

    class Class < LinkedData::Models::Base
      include ResourceIndex::Class

      model :class, name_with: :id, collection: :submission,
            namespace: :owl, :schemaless => :true,
            rdf_type: lambda { |*x| self.class_rdf_type(x) }

      def self.class_rdf_type(*args)
        submission = args.flatten.first
        return RDF::OWL[:Class] if submission.nil?
        if submission.bring?(:classType)
          submission.bring(:classType)
        end
        if not submission.classType.nil?
          return submission.classType
        end
        unless submission.loaded_attributes.include?(:hasOntologyLanguage)
          submission.bring(:hasOntologyLanguage)
        end
        if submission.hasOntologyLanguage
          return submission.hasOntologyLanguage.class_type
        end
        return RDF::OWL[:Class]
      end

      def self.urn_id(acronym,classId)
        return "urn:#{acronym}:#{classId.to_s}"
      end

      attribute :submission, :collection => lambda { |s| s.resource_id }, :namespace => :metadata

      attribute :label, namespace: :rdfs, enforce: [:list]
      attribute :prefLabel, namespace: :skos, enforce: [:existence], alias: true
      attribute :synonym, namespace: :skos, enforce: [:list], property: :altLabel, alias: true
      attribute :definition, namespace: :skos, enforce: [:list], alias: true
      attribute :obsolete, namespace: :owl, property: :deprecated, alias: true
```

Class model

Class Standard Attributes

```
attribute :prefLabel, namespace: :skos, enforce: [:existence], alias: true
attribute :synonym, namespace: :skos, enforce: [:list], property: :altLabel, alias: true
attribute :definition, namespace: :skos, enforce: [:list], alias: true
attribute :obsolete, namespace: :owl, property: :deprecated, alias: true
```

NCIT uses the predicate below as prefLabel property

http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#P108

Class Standard Attributes

```
ont = LinkedData::Models::Ontology.find("NCIT").first
submission = ont.latest_submission

cls = LinkedData::Models::Class
          .find( RDF::URI.new("http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#C99532") )
          .in( submission )
          .include( :prefLabel).first
```

```
SELECT DISTINCT ?id ?prefLabel ?synonyms
FROM <http://data.bioontology.org/ontologies/NCIT/submissions/38> WHERE {
    ?id a <http://www.w3.org/2002/07/owl#Class> .
    OPTIONAL { ?id ?rewrite0 ?prefLabel . }
    FILTER(?id = <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#C99532>)
    FILTER(?rewrite0 = <http://data.bioontology.org/metadata/def/prefLabel> ||
            ?rewrite0 = <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#P108> ||
            ?rewrite0 = <http://www.w3.org/2004/02/skos/core#prefLabel>) }
```

Unmapped data attributes

```
cls = LinkedData::Models::Class
        .find( RDF::URI.new("http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.c
        .in( submission )
        .include( :unmapped).first
```


```
@unmapped=
{#<RDF::URI:0x3f8de23e7d30(http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#P106)>=>
 [#<RDF::Literal:0x3f8de2014cbc("Therapeutic or Preventive Procedure"^^<http://www.w3.org/2001/XMLSchema#string>)>],
#<RDF::URI:0x3f8de23e696c(http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#P108)>=>
 [#<RDF::Literal:0x3f8de2014618("Diabetes Therapy"^^<http://www.w3.org/2001/XMLSchema#string>)>],
#<RDF::URI:0x3f8de23e3a78(http://www.w3.org/2000/01/rdf-schema#label)>=>[#<RDF::Literal:0x3f8de2013d30("Diabetes Therapy"^^<http://www.w3.org/2001/XMLSchema#string>)>],
#<RDF::URI:0x3f8de23e3b54(http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#P97)>=>
 [#<RDF::Literal:0x3f8de2013808("A therapeutic modality used to aide in the management of an individual's diabetes."^^<http://www.w3.org/2001/XMLSchema#string>)>],
#<RDF::URI:0x3f8de23e6070(http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#P325)>=>
 [#<RDF::Literal:0x3f8de20133d0("A therapeutic modality used to aide in the management of an individual's diabetes."^^<http://www.w3.org/2001/XMLSchema#string>)>],
#<RDF::URI:0x3f8de23e72b8(http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#P322)>=>[#<RDF::Literal:0x3f8de2012ef8("CDISC"^^<http://www.w3.org/2001/XMLSchema#string>)>],
#<RDF::URI:0x3f8de23e7024(http://data.bioontology.org/metadata/def/mappingLoom)>=>[#<RDF::Literal:0x3f8de2012a98("diabetestherapy"^^<http://www.w3.org/2001/XMLSchema#string>
#<RDF::URI:0x3f8de23e3bf4(http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#P207)>=>[#<RDF::Literal:0x3f8de20126b0("C3274787"^^<http://www.w3.org/2001/XMLSchema#string>)>],
#<RDF::URI:0x3f8de23e7da8(http://data.bioontology.org/metadata/prefixIRI)>=>[#<RDF::Literal:0x3f8de2012214("C99532"^^<http://www.w3.org/2001/XMLSchema#string>)>],
#<RDF::URI:0x3f8de23e6d18(http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#code)>=>[#<RDF::Literal:0x3f8de200fbb8("C99532"^^<http://www.w3.org/2001/XMLSchema#string>)>],
#<RDF::URI:0x3f8de23e7998(http://data.bioontology.org/metadata/def/mappingSameURI)>=>[#<RDF::URI:0x3f8de200f7d0(http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#C99532)>],
#<RDF::URI:0x3f8de23e6bec(http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#A8)>=>
 [#<RDF::URI:0x3f8de200f3fc(http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#C61410)>, #<RDF::URI:0x3f8de200e628(http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#C101836)>],
#<RDF::URI:0x3f8de23e6778(http://www.w3.org/1999/02/22-rdf-syntax-ns#type)>=>
 [#<RDF::URI:0x3f8de200f078(http://www.w3.org/2002/07/owl#NamedIndividual)>, #<RDF::URI:0x3f8de200ed58(http://www.w3.org/2002/07/owl#Class)>],
#<RDF::URI:0x3f8de23e6ebc(http://www.w3.org/2000/01/rdf-schema#subClassOf)>=>[#<RDF::URI:0x3f8de200e9d4(http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#C49236)>]}>
```

Unmapped Pagination

```
paging = LinkedData::Models::Class.in(submission).include(:unmapped)
                .page(page,size)
```

```
begin #per page

    page_classes = paging.page(page,size).all
    page_classes.each do |c|
        # do something with c
    end
    page = page_classes.next? ? page + 1 : nil
end while !page.nil?
```

Caching

Goo implements a graph based cache system. Goo knows what graphs get affected in an update so it can always invalidate the cache.

```
SELECT ?user ?username ?affiliation ?projects ?roles
FROM :User FROM :Ontology :Note
WHERE { ?user a :User .
        ?user :created ?created .
        ?note :owner ?user .
        ?note :ontology [ :acronym "SNOMEDCT" ] .
        ?note :visibility [ :code "public" ] .
        ?user :projects ?projects .
        ?user :roles ?roles . }
```

MD5.hexdigest(query) => a993fa... 7ababa98

graphs => [:Note, :Ontology, :User]

CACHE KEY: sparql:query:Note:Ontology:User: a993fa... 7ababa98

Cache

The cache system uses two data structures:

A

graph ids in the query (sorted) SPARQL string MD5.hexdigest

KEY: http://.../ontologies/SNOMEDCT/submissions/6:a993fa... 7aba987aba98

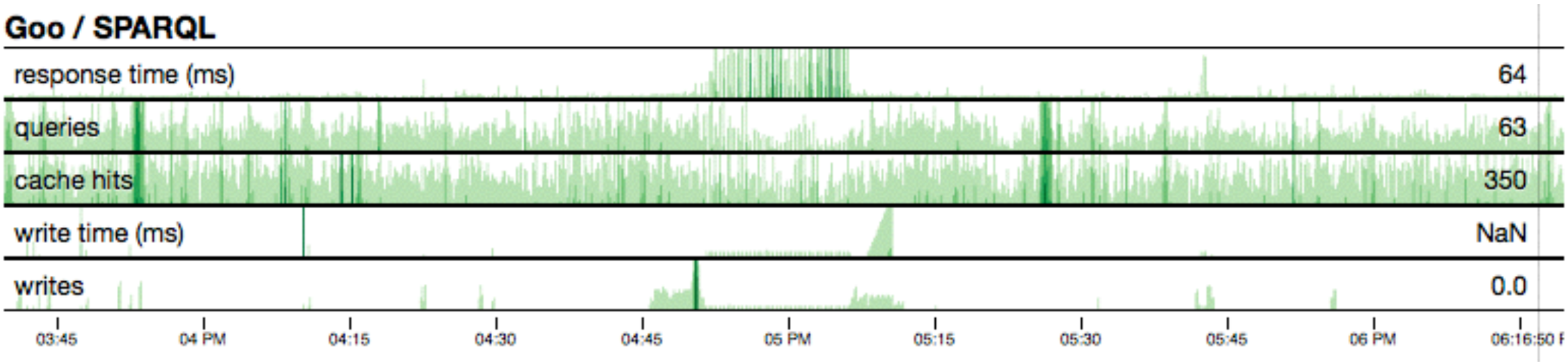
VALUE: Marshal.dump(query_result)

B

KEY: sparql:graph:http://.../ontologies/SNOMEDCT/submissions/6

VALUE: [http://.../ontologies/SNOMEDCT/submissions/6:a993fa... 7aba987aba98 ,
http://.../ontologies/SNOMEDCT/submissions/6:a993fa... a9990a000011 ,
http://.../ontologies/SNOMEDCT/submissions/6:a993fa... eda8768a0a01]

Cache


Goo / SPARQL
response time (ms)
64
queries
63
cache hits
350
write time (ms)
NaN
writes
0.0
03:45
04 PM
04:15
04:30
04:45
05 PM
05:15
05:30
05:45
06 PM

Questions