This Talk @

`http://tiny.cc/ldbc_sadi`

SADI

Find. Integrate. Analyze.

Semantic Automated Discovery and Integration

A design-pattern for “native” Linked-Data Semantic Web Services

Mark D. Wilkinson
Fundacion BBVA Chair in Biological Informatics
CBGP-UPM Madrid

Fundación BBVA

CBGP
UPM-INIA

What’s the Problem?

XML Schema

XML Schema allows us to
describe, to a machine,
the structure of an XML
document

Therefore we can
share, integrate, and
aggregate data!

sh nd

What did XML Schema do for us?

"...XML Schema (among other things) allowed us to ~automate the creation of memory-structures which could hold the given XML-formatted data..."

-- Paul Gordon, SUN COE, Calgary

Does not solve the integration or aggregation problem

EMBL Nucleotide Record Schema

XML Schema

*There will be an element called "***qualifier***"*

*It will have an attribute called "***name***"*

The content of that attribute will be ***text***

*There will be a child element called "***value***"*

The content of that child element will be ***free-text***

XML Schema

*There will be an element called "***GBQualifier***"*

*There will be a child element called "***GBQualifier_name***"*

The content of that child element will be ***free-text***

*There will be a child element called "***GBQualifier_value***"*

The content of that child element will be ***free-text***

GenBank Nucleotide Record Schema

EMBL Nucleotide Record Schema

XML Schema

*There will be an element called “***qualifier***”*

*It will have an attribute called “***name***”*

The content of that attribute will be **text**

*There will be a child element called “***value***”*

The content of that child element will be **free-text**

These two fragments represent XML documents that contain EXACTLY the same data;

However we cannot immediately integrate them...

XML Schema

*There will be an element called “***GBQualifier***”*

*There will be a child element called “***GBQualifier_name***”*

The content of that child element will be **free-text**

*There will be a child element called “***GBQualifier_value***”*

The content of that child element will be **free-text**

GenBank Nucleotide Record Schema

EMBL Nucleotide Record Schema

XML Schema

*There will be an element called "***qualifier***"*

*It will have an attribute called "***name***"*

The content of that attribute will be **text**

*There will be a child element called "***value***"*

The content of that child element will be **free-text**

...because the "meaning" of each Schema element is implicit.

Therefore, we resort to "Schema Mapping" to integrate the data

XML Schema

*There will be an element called "***GBQualifier***"*

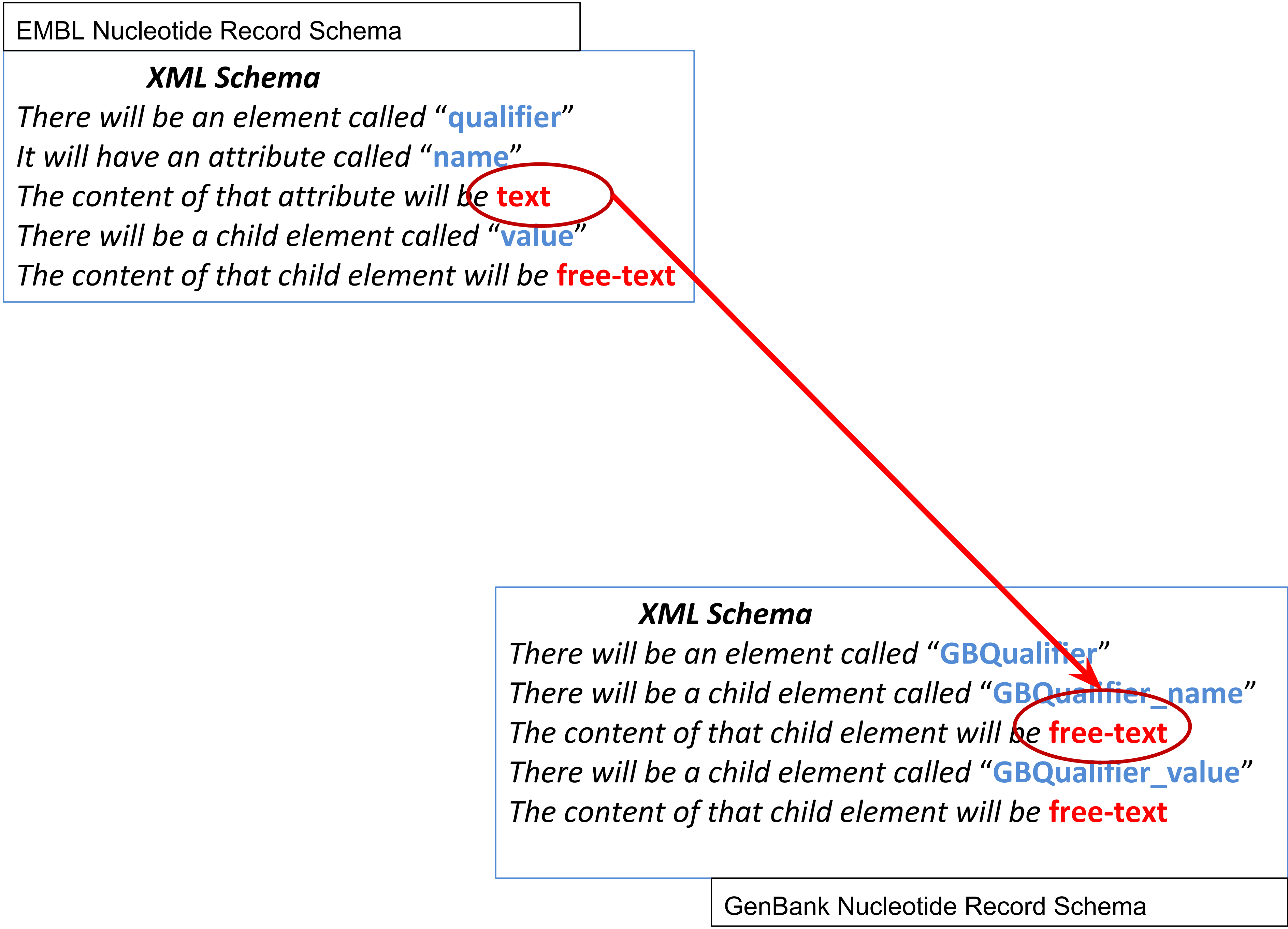
*There will be a child element called "***GBQualifier_name***"*

The content of that child element will be **free-text**

*There will be a child element called "***GBQualifier_value***"*

The content of that child element will be **free-text**

GenBank Nucleotide Record Schema



EMBL Nucleotide Record Schema

XML Schema

*There will be an element called “***qualifier***”*

*It will have an attribute called “***name***”*

The content of that attribute will be **text**

*There will be a child element called “***value***”*

The content of that child element will be **free-text**

XML Schema

*There will be an element called “***GBQualifier***”*

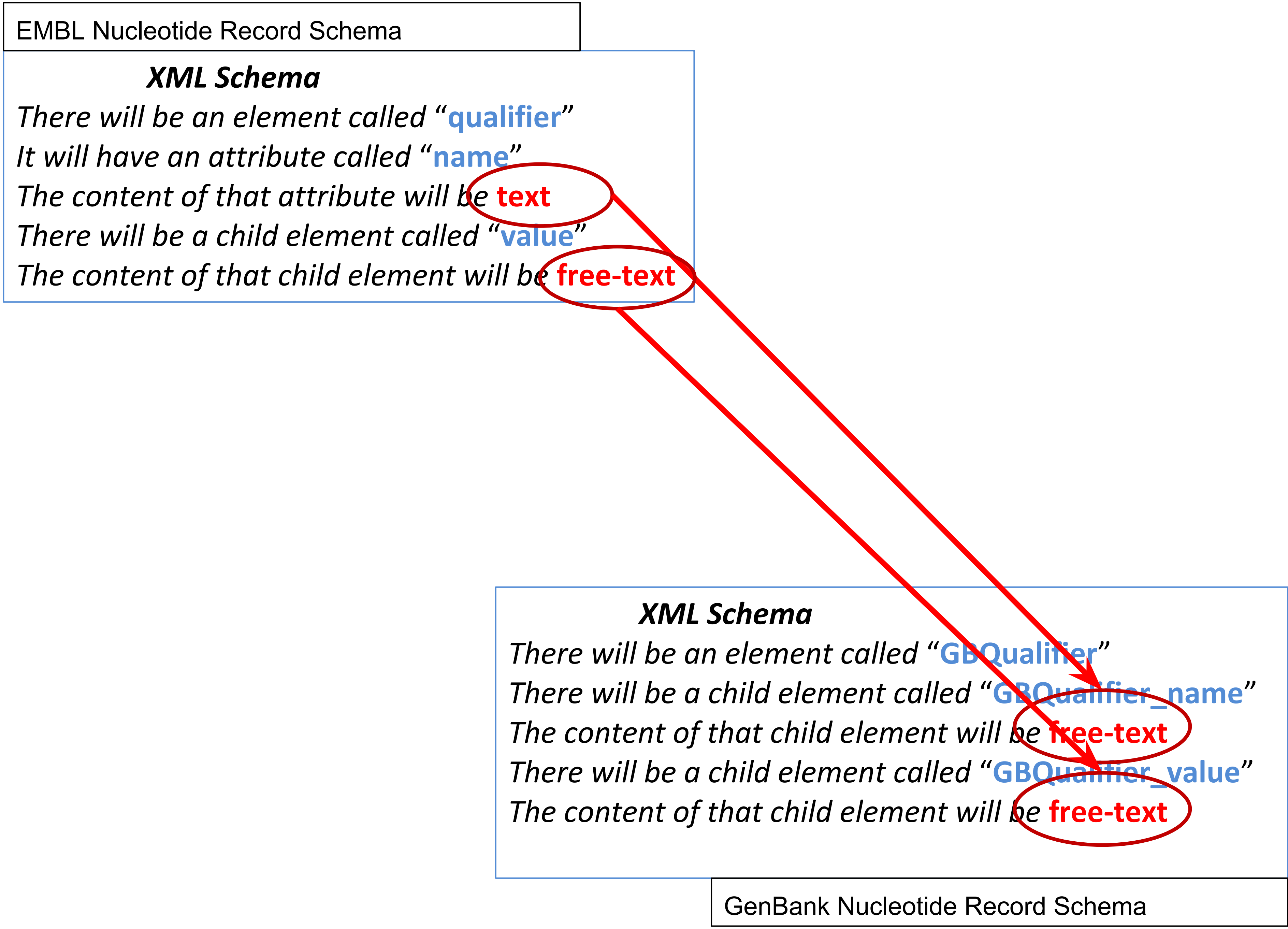
*There will be a child element called “***GBQualifier_name***”*

The content of that child element will be **free-text**

*There will be a child element called “***GBQualifier_value***”*

The content of that child element will be **free-text**

GenBank Nucleotide Record Schema



EMBL Nucleotide Record Schema

XML Schema

*There will be an element called "***qualifier***"*

*It will have an attribute called "***name***"*

The content of that attribute will be **text**

*There will be a child element called "***value***"*

The content of that child element will be **free-text**

XML Schema

*There will be an element called "***GBQualifier***"*

*There will be a child element called "***GBQualifier_name***"*

The content of that child element will be **free-text**

*There will be a child element called "***GBQualifier_value***"*

The content of that child element will be **free-text**

GenBank Nucleotide Record Schema

So, obviously, all we need to do is automate the process of schema-mapping, and then we will achieve interoperability!

So, obviously, a automate the
process of sch d then we will
ach ty!

Though there have been numerous attempts to automate schema mapping none have proven reliable in an open-Web situation

Ozan Kılıç Y, Aydin MN: **Automatic XML Schema Matching.** European and Mediterranean Conference on Information Systems 2009 (EMCIS2009), July 13-14, 2009

Nevertheless...

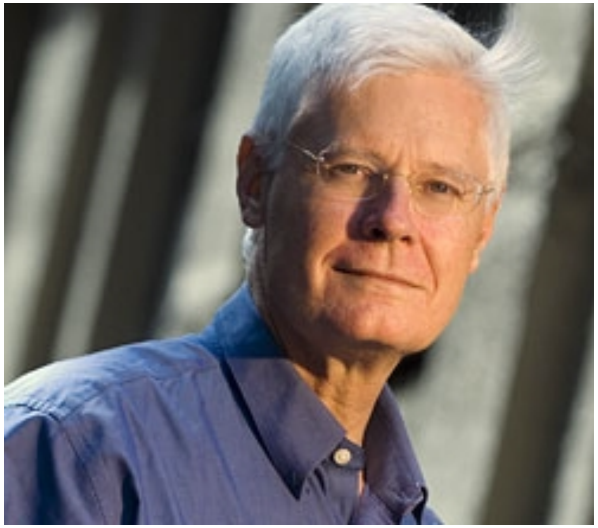
Web Services

“Service Oriented Architectures”

WSDL
(and many other 4-letter words)

But…

XML Schema



“The phrase ‘practical Web Services’ is not intrinsically an oxymoron, but [I] argue that there are few in existence.”

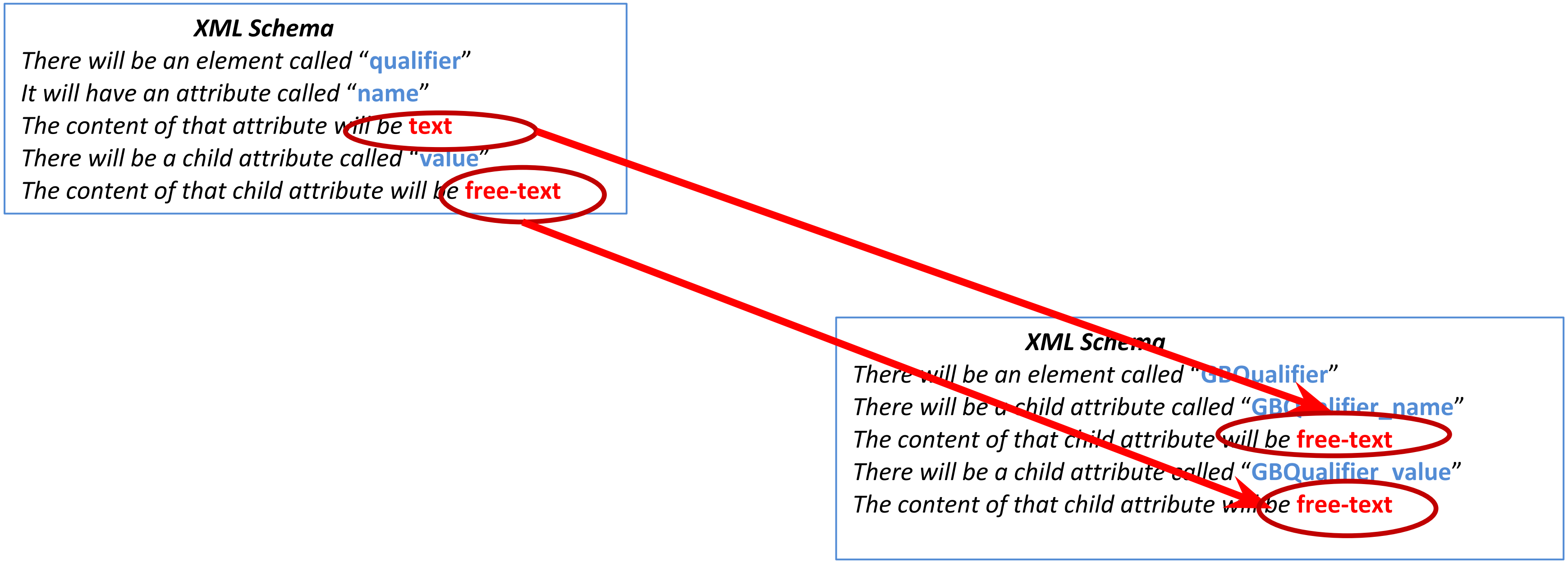
-- Charles Petrie, Stanford University

Why?

Because the automated-schema matching problem
is ***so disruptive***
that ***there is little point*** in building
“modular/reusable” Web Services...

They are simply too difficult to integrate
with other Web Services, so why bother even trying?

-- adapted from Petrie, SWSIP 2009



XML Schema
*There will be an element called “***qualifier***”*
*It will have an attribute called “***name***”*
The content of that attribute will be **text**
*There will be a child attribute called “***value***”*
The content of that child attribute will be **free-text**

XML Schema
*There will be an element called “***GBQualifier***”*
*There will be a child attribute called “***GBQualifier_name***”*
The content of that child attribute will be **free-text**
*There will be a child attribute called “***GBQualifier_value***”*
The content of that child attribute will be **free-text**

Then we moved into very dark times...

We still want SOA’s, so...

…rather than modular Services, we’ll just build Services that do the entire operation as a single function!

These Services, therefore,
had a much higher complexity

(both w.r.t. data types and
the functional description of the service)

So…
perversely…

XML Schema

made the interoperability problem

WORSE!

But there is hope!

“Linked Data” movement

Resource Description Framework
“RDF”

The “Semantic Web” movement

Web Ontology Language
“OWL”

What does RDF do for us?

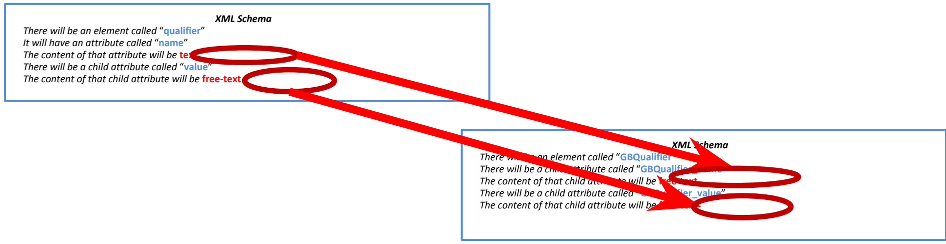
"...RDF replaces XML Schema, because RDF says that ***there is only one data model***..."

-- Paul Gordon, SUN COE, Calgary

What does OWL do for us?

“...the semantics are ***no longer implicit*** in that data model...”

-- Paul Gordon, SUN COE, Calgary



XML Schema

*There will be an element called “**qualifier**”*
*It will have an attribute called “**name**”*
*The content of that attribute will be **text***
*There will be a child attribute called “**value**”*
*The content of that child attribute will be **free-text***

XML Schema

*There will be an element called “**GBQualifier**”*
*There will be a child attribute called “**GBQualifier_name**”*
*The content of that child attribute will be **free-text***
*There will be a child attribute called “**GBQualifier_value**”*
*The content of that child attribute will be **free-text***

SADI

Find. Integrate. Analyze.

Semantic **A**utomated **D**iscovery and **I**ntegration

A semantics-based Web Services design-pattern

http://sadiframework.org

SADI

Find. Integrate. Analyze.

Make Web Services look more like the Semantic Web standards-compliant

Lightweight

(only 2 “rules”)

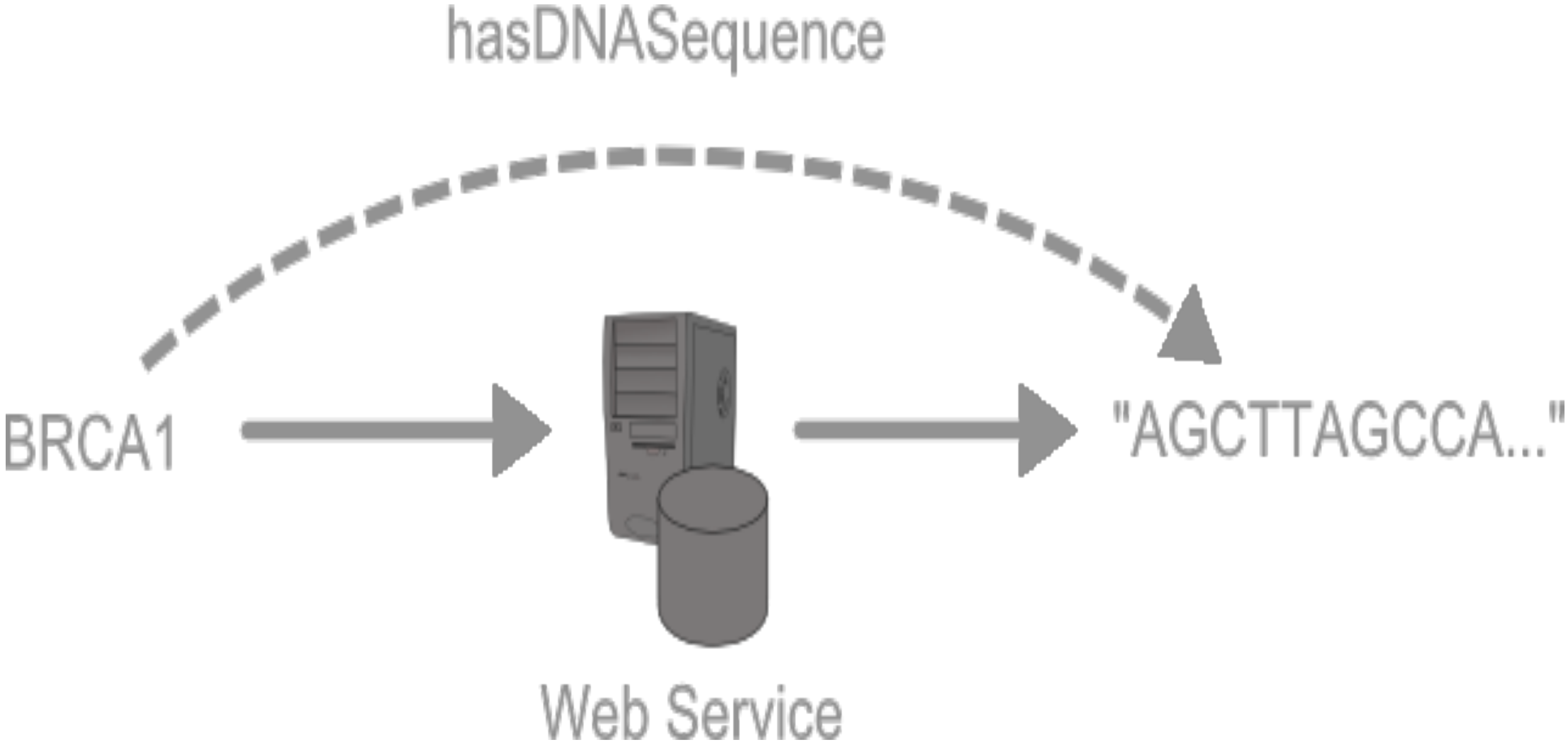
Rules were based on our
observations of Web Service functionality

(specifically in the bioinformatics space)

Observation #1:

Web Services in Bioinformatics create
implicit biological relationships
between their input and output

Observation #1:



SADI Design Pattern #1

Make the implicit ***explicit…***

A Web Service should create “triples” linking the input data to the output data, thus explicitly describing the semantic relationship between them

Observation #2:

HTTP GET and POST

GET guarantees
the response relates to the request URI
in a very precise and predictable way

POST does not…

Observation #2:

HTTP GET and POST

That’s why Web Services have a fundamentally different *behaviour* than the Semantic Web

Observation #2:

HTTP GET and POST

We can fix that!

(without breaking any existing rules or standards!)

SADI Design Pattern #2

SUBJECT URI of the **output** graph (triples)

is the same as

SUBJECT URI of the **input** graph (triples)

(the output is “about” the input... Now explicitly!)

Consequence

Web Services now exhibit a very similar behavior to the Web itself

POST “behaves like” GET

SADI Interface Definitions

Service Interfaces defined by
two OWL classes:

SADI Interface Definitions

OWL Class #1: My Input Class

SADI Interface Definitions

OWL Class #2: My Output Class

SADI Service Invocation

Consumes OWL Individuals (RDF) of Class #1

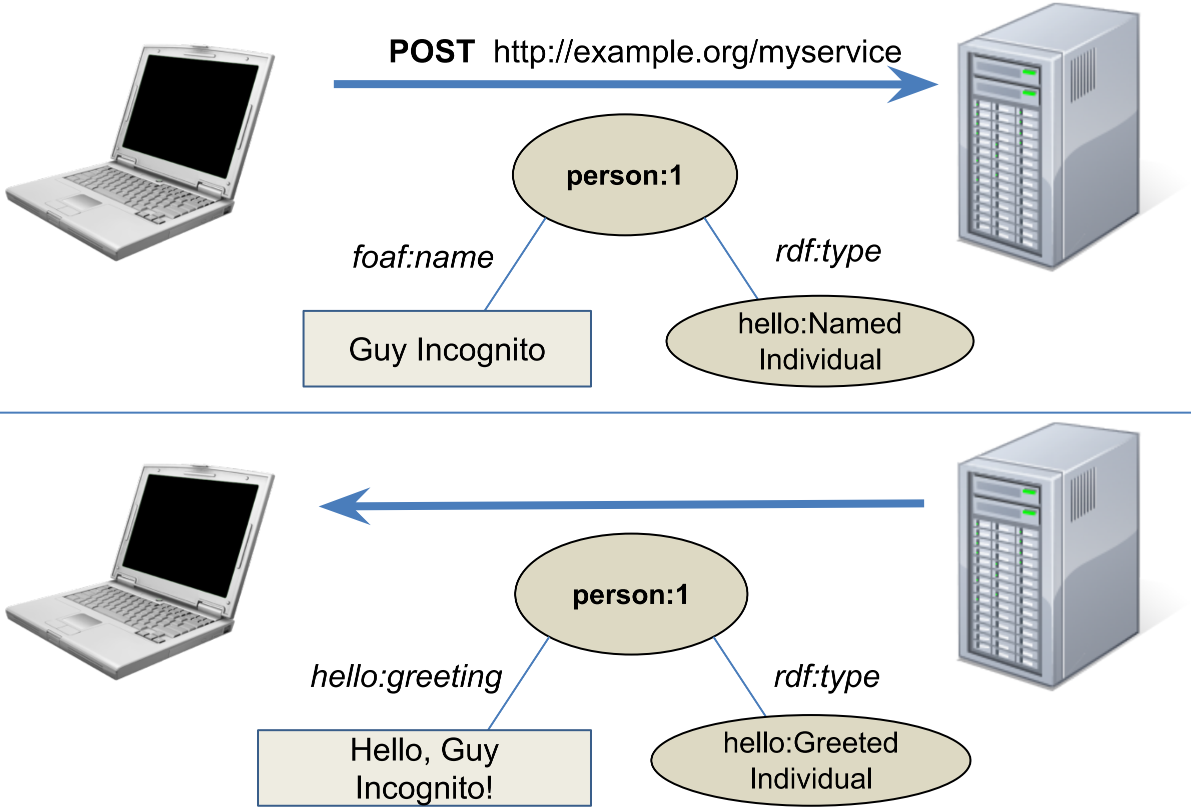
Returns OWL Individuals (RDF) of Class #2

…but the URI of those two individuals is the same!
(see design pattern #2)

Service Description

INPUT OWL Class
NamedIndividual: things with
a “name” property
from “foaf” ontology

OUTPUT OWL Class
GreetedIndividual: things with
a “greeting” property
from “hello” ontology



POST http://example.org/myservice

person:1

foaf:name

rdf:type

Guy Incognito

hello:Named Individual

person:1

hello:greeting

rdf:type

Hello, Guy Incognito!

hello:Greeted Individual

Service Discovery

Input and output are about the same “thing”

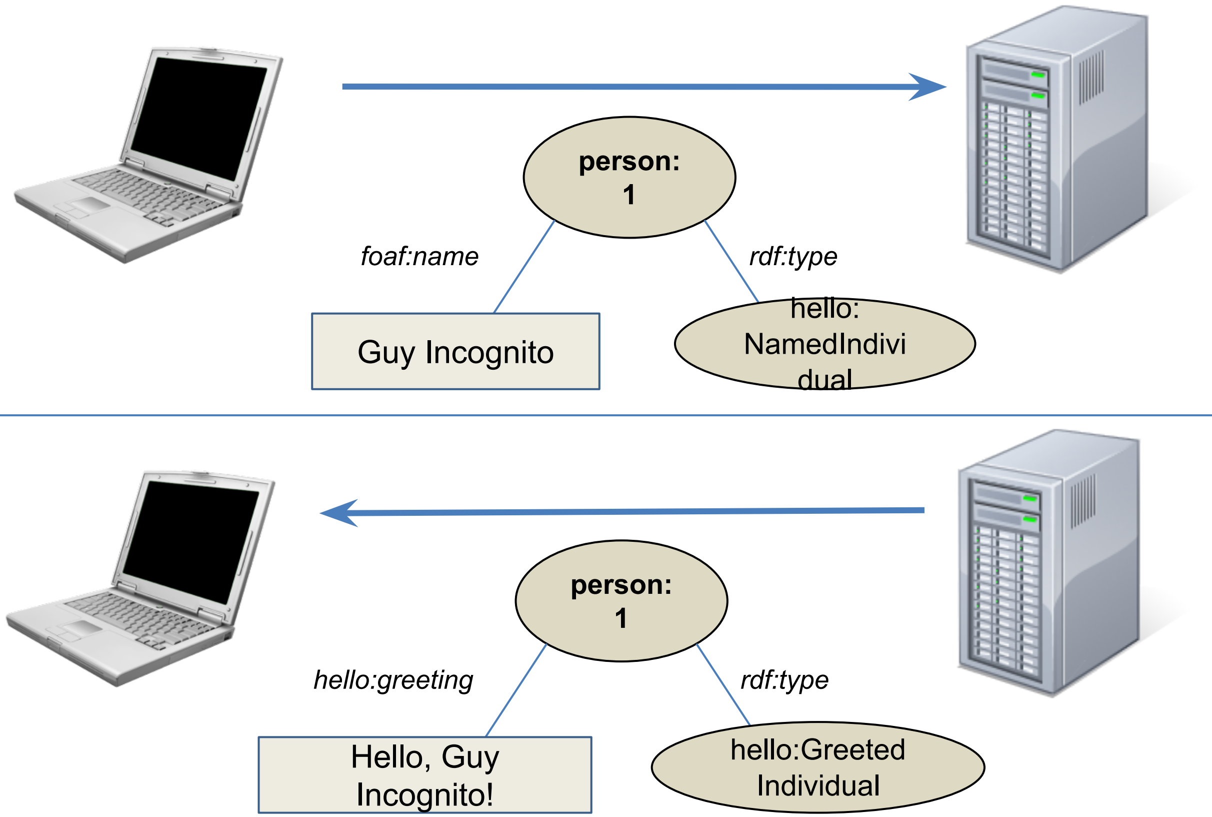
Therefore, to describe what a service ***does***
simply compare (“diff”) the
Input and Output OWL classes

Service Description

INPUT OWL Class
NamedIndividual: things with
a “name” property
from “foaf” ontology

OUTPUT OWL Class
GreetedIndividual: things with
a “greeting” property
from “hello” ontology

The service provides a “greeting” to any entity that has a “name” property



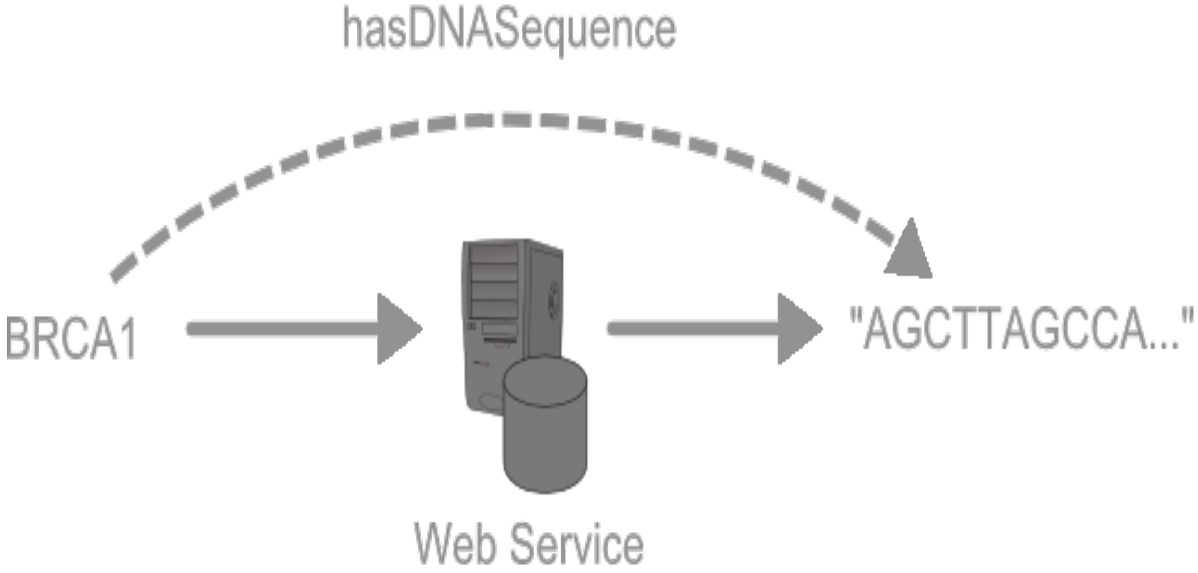
Service Registry

Index of all properties

consumed/produced

by all services

Real-world Example



Input Data: BRCA1 rdf:type Gene ID

Output Data: BRCA1 hasDNASequence AGCTTAGCCA…

Registry Index: Service provides “hasDNASequence” property to Gene IDs e.g. The question:

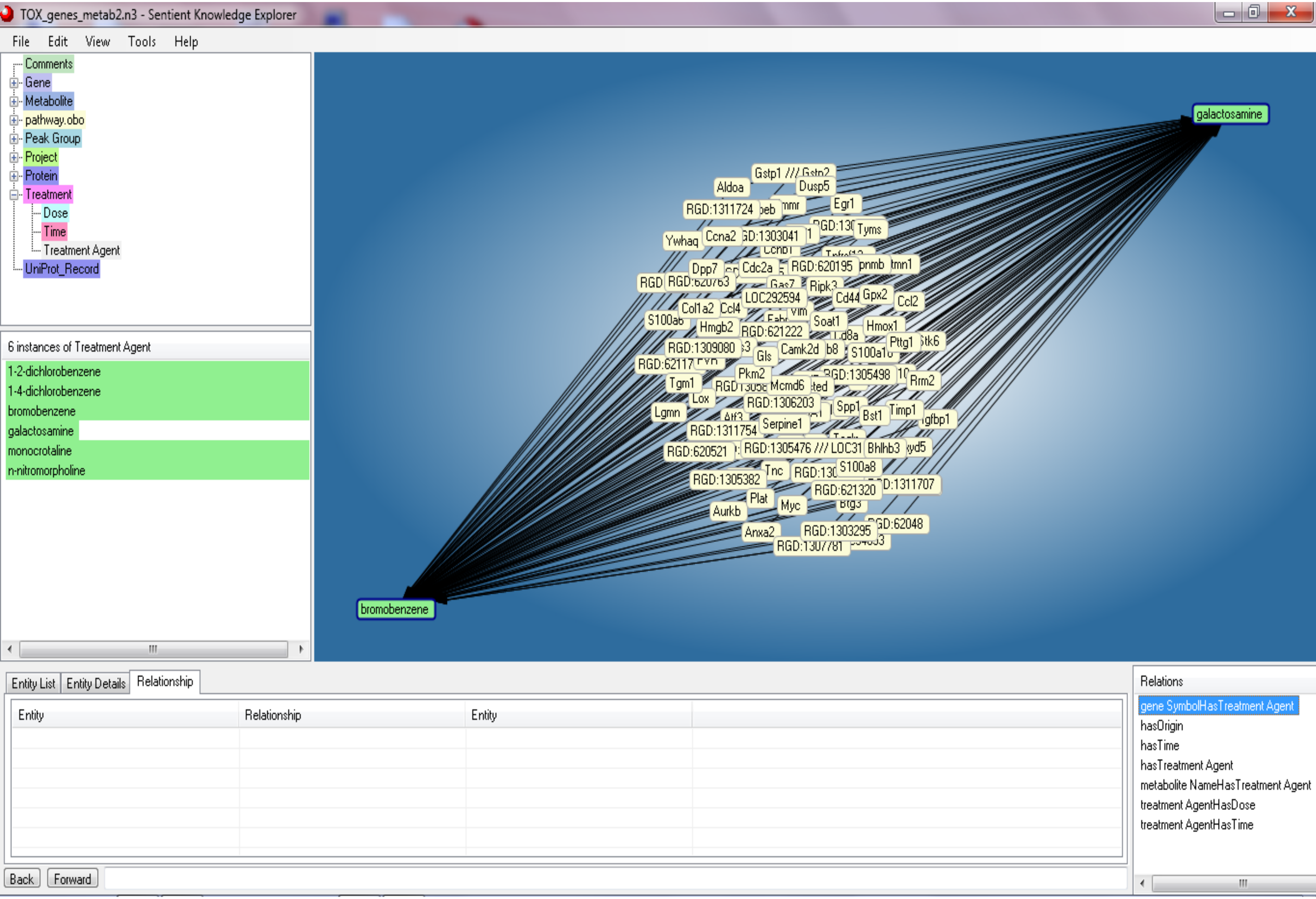
“*what is the DNA sequence of BRCA1?*”

Discover a SADI Web Service that generates the DNA Sequence property for gene identifiers

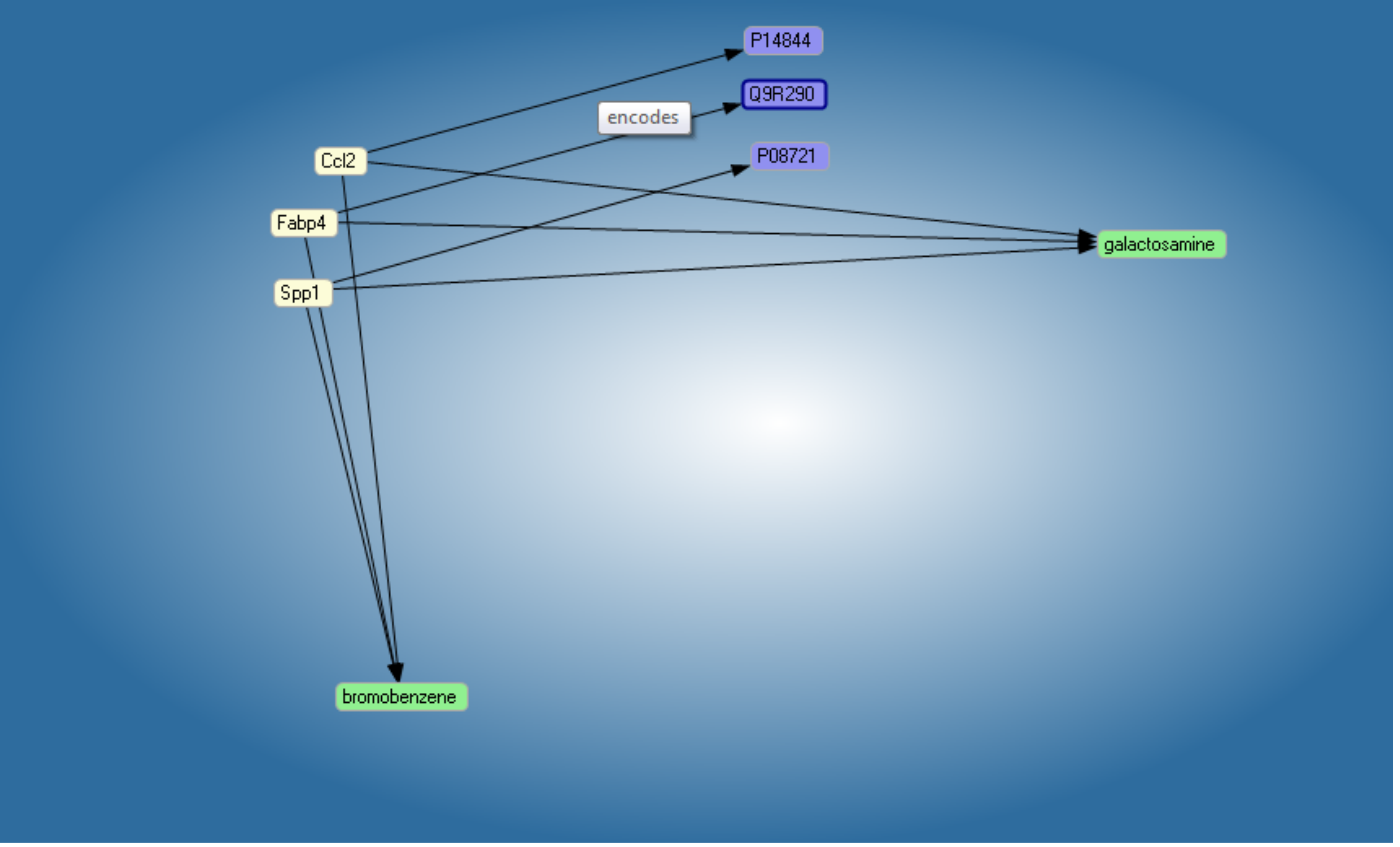
Describing service functionality in this way turns out to be extremely powerful!

Knowledge Explorer
Plug-in

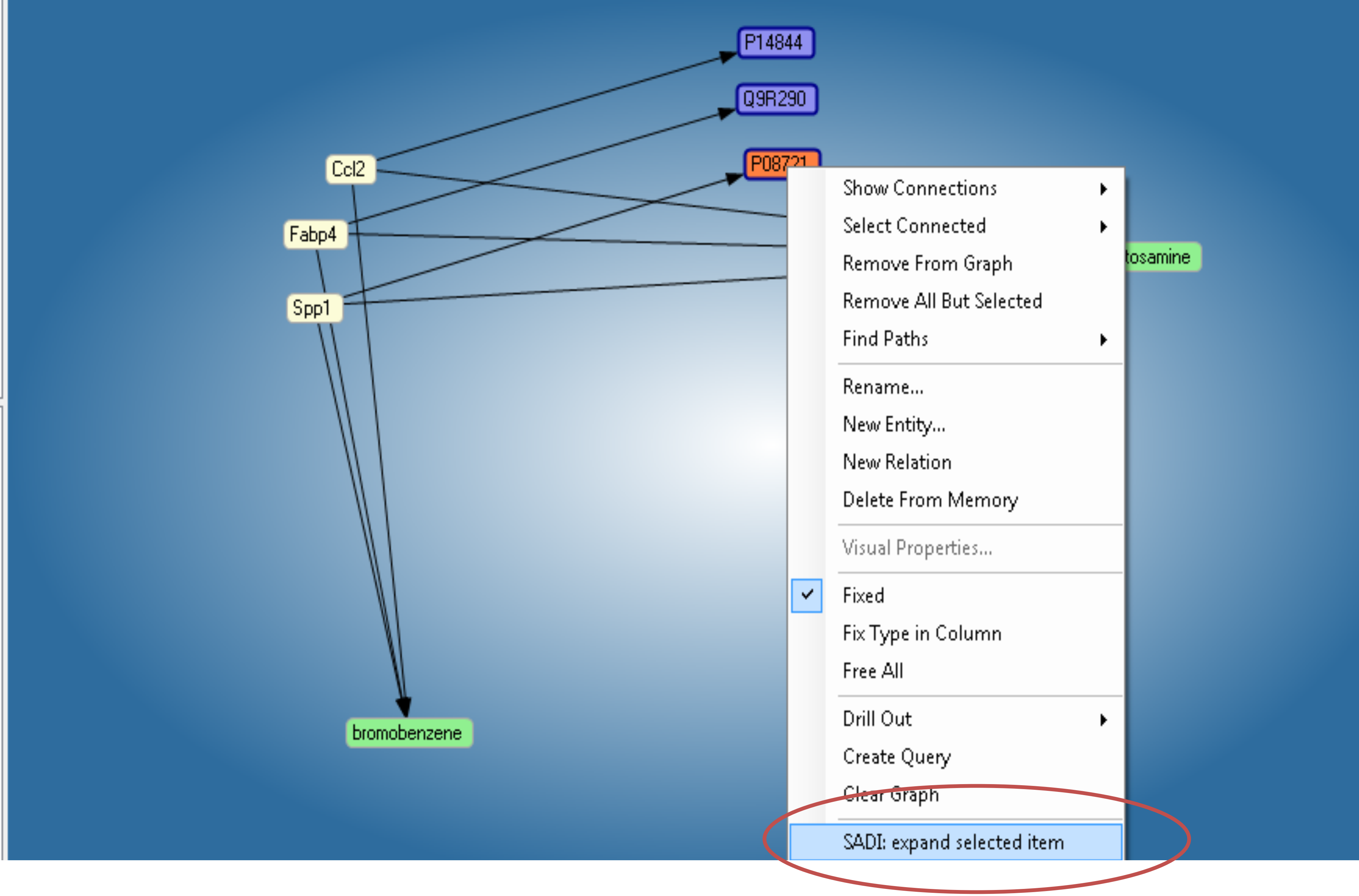
For more information about the Knowledge Explorer surf to:
http://io-informatics.com

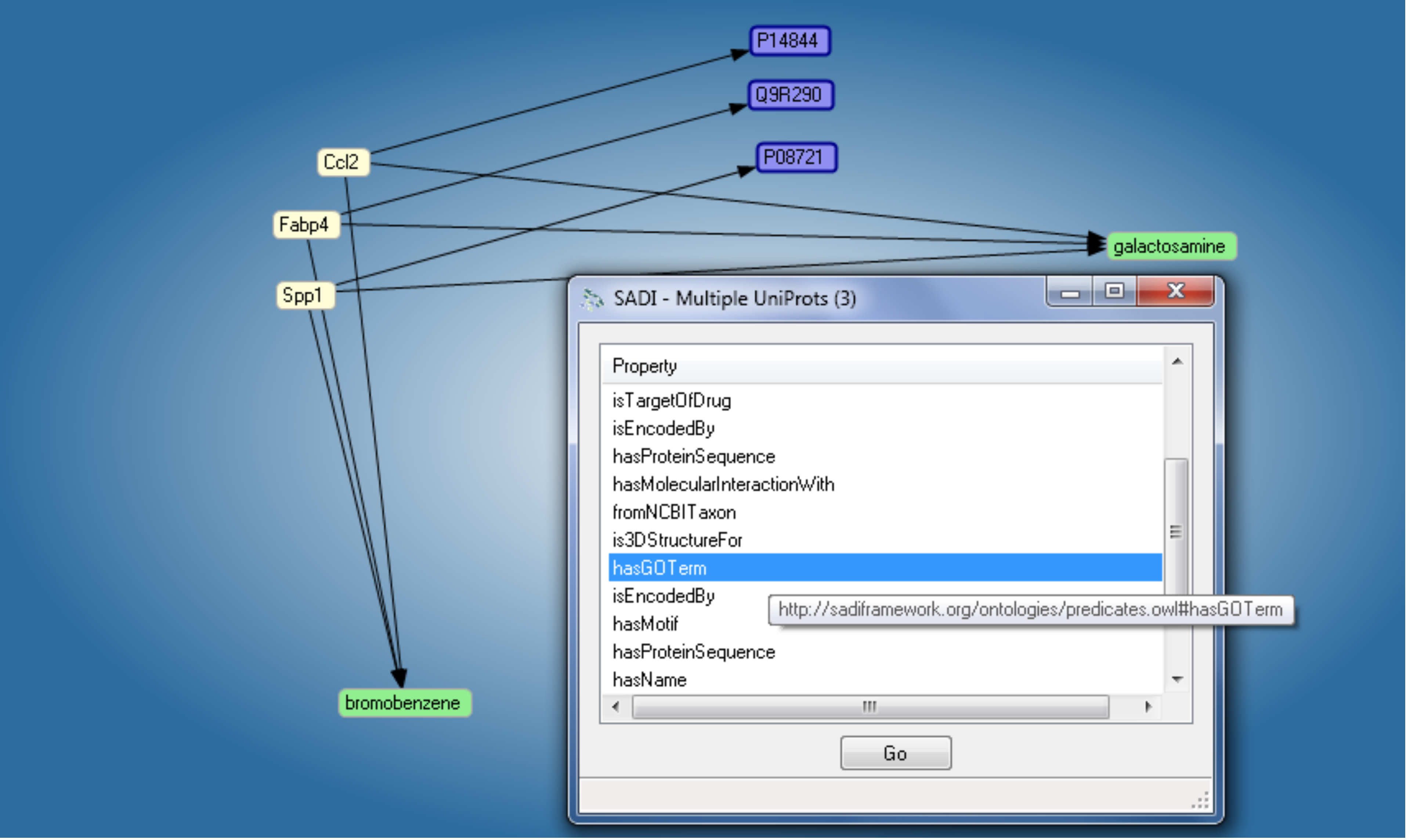


Entity	Relationship	Entity

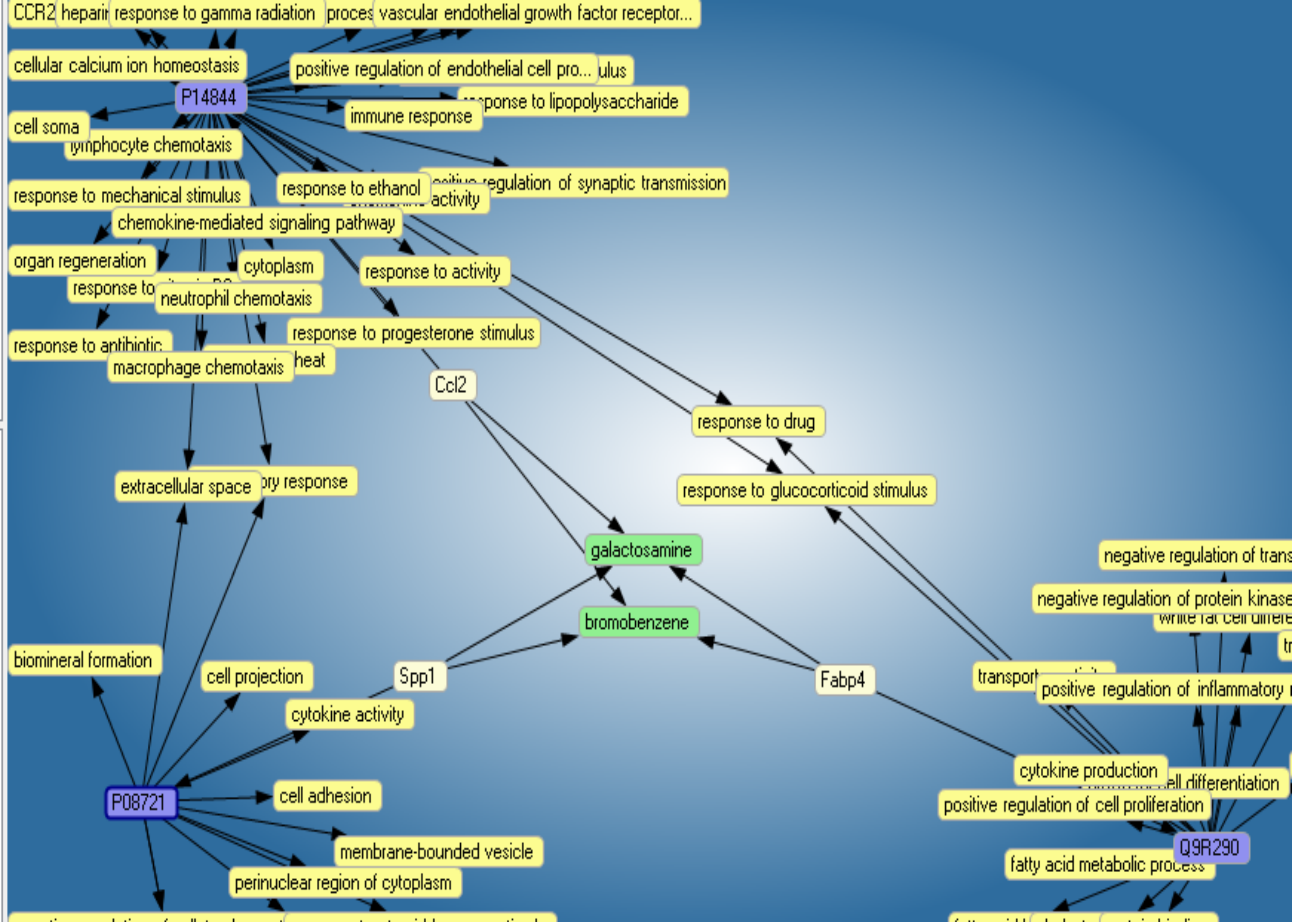


SADI has just invoked a service that provided the “Encodes” property for the three genes of interest. Three new nodes appear that are “Protein Sequence” type nodes


P14844
Q9R290
P08721
Ccl2
Fabp4
Spp1
tosamine
bromobenzene
Show Connections
Select Connected
Remove From Graph
Remove All But Selected
Find Paths
Rename...
New Entity...
New Relation
Delete From Memory
Visual Properties...
✓ Fixed
Fix Type in Column
Free All
Drill Out
Create Query
Clear Graph
SADI: expand selected item



Ask the SADI Registry what properties can be provided to things of type “Protein Sequence”;
Discover a service that provides the hasGOTerm property


CCR2
heparir
response to gamma radiation
proces
vascular endothelial growth factor receptor...
cellular calcium ion homeostasis
positive regulation of endothelial cell pro...
ulus
P14844
response to lipopolysaccharide
immune response
cell soma
lymphocyte chemotaxis
positive regulation of synaptic transmission
response to ethanol
activity
response to mechanical stimulus
chemokine-mediated signaling pathway
organ regeneration
cytoplasm
response to activity
response to
neutrophil chemotaxis
response to progesterone stimulus
response to antibiotic
macrophage chemotaxis
heat
Ccl2
response to drug
extracellular space
ory response
response to glucocorticoid stimulus
galactosamine
negative regulation of trans
negative regulation of protein kinase
bromobenzene
biomineral formation
cell projection
Spp1
Fabp4
transport
positive regulation of inflammatory
cytokine activity
cytokine production
ell differentiation
P08721
cell adhesion
positive regulation of cell proliferation
Q9R290
membrane-bounded vesicle
fatty acid metabolic process
perinuclear region of cytoplasm

SHARE

Semantic **H**ealth **A**nd **R**esearch **E**nvironment

SPARQL + Registry Lookup + Service Invocation + Workflow Orchestration + DL Reasoning

SHARE

Semantic **H**ealth **A**nd **R**esearch **E**nvironment

SHARE answers arbitrary SPARQL queries by finding and executing SADI Services

Example #1

What is the phenotype of every allele of the *Antirrhinum majus* DEFICIENS gene

```
SELECT ?allele  ?image  ?desc

WHERE {
      locus:DEF       genetics:hasVariant        ?allele .
        ?allele       info:visualizedByImage     ?image .
        ?image       info:hasDescription        ?desc
}
```

Example #1

What is the phenotype of every allele of the *Antirrhinum majus* DEFICIENS gene

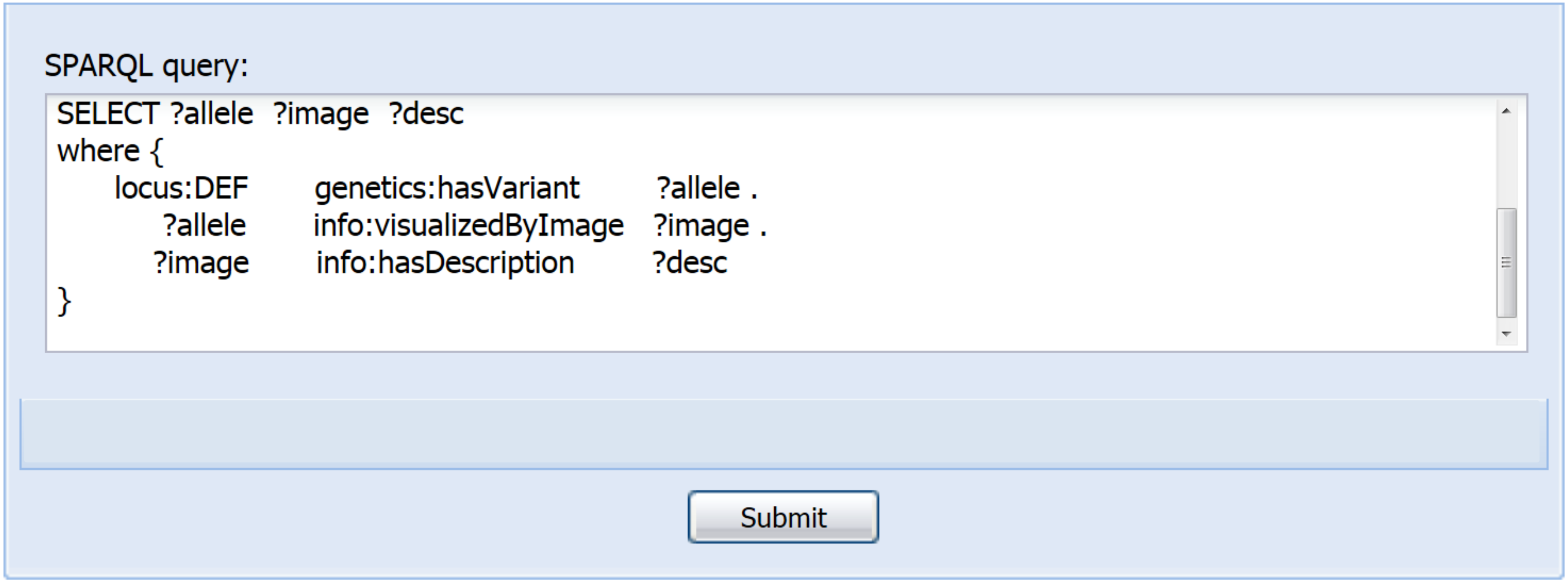
```
SELECT ?allele  ?image  ?desc

WHERE {
      locus:DEF       genetics:hasVariant         ?allele .
        ?allele       info:visualizedByImage      ?image .
        ?image       info:hasDescription         ?desc
}
```

Note that there is no “FROM” clause!
We don’t tell it ***where*** it should get the information,
The machine has to figure that out by itself...

SHARE

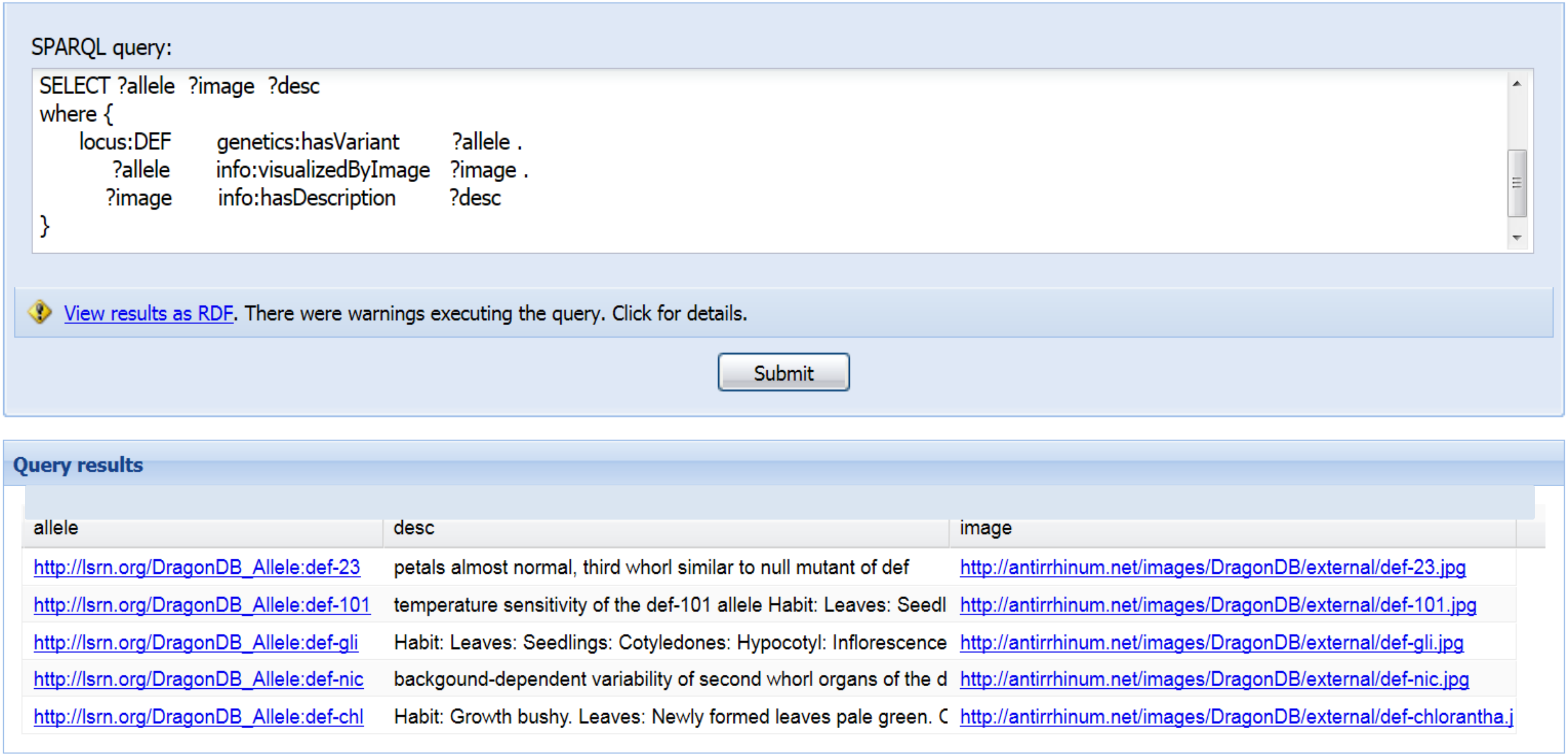
Enter that query into
SHARE



SPARQL query:

```
SELECT ?allele  ?image  ?desc
where {
     locus:DEF       genetics:hasVariant        ?allele .
         ?allele       info:visualizedByImage   ?image .
        ?image       info:hasDescription        ?desc
}
```

Click “Submit”...



SPARQL query:

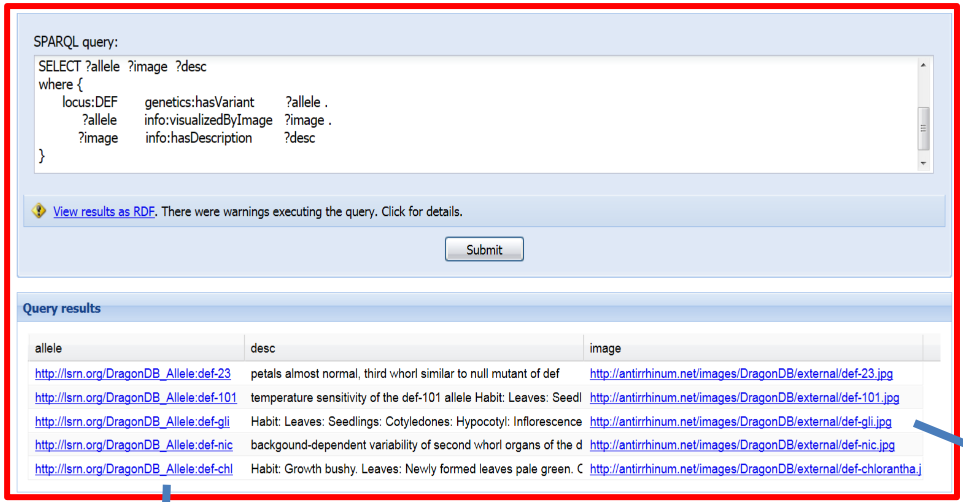
```
SELECT ?allele  ?image  ?desc
where {
     locus:DEF      genetics:hasVariant       ?allele .
         ?allele      info:visualizedByImage   ?image .
        ?image      info:hasDescription        ?desc
}
```

View results as RDF. There were warnings executing the query. Click for details.

Query results

allele	desc	image
http://lsrn.org/DragonDB_Allele:def-23	petals almost normal, third whorl similar to null mutant of def	http://antirrhinum.net/images/DragonDB/external/def-23.jpg
http://lsrn.org/DragonDB_Allele:def-101	temperature sensitivity of the def-101 allele Habit: Leaves: Seedl	http://antirrhinum.net/images/DragonDB/external/def-101.jpg
http://lsrn.org/DragonDB_Allele:def-gli	Habit: Leaves: Seedlings: Cotyledones: Hypocotyl: Inflorescence	http://antirrhinum.net/images/DragonDB/external/def-gli.jpg
http://lsrn.org/DragonDB_Allele:def-nic	backgound-dependent variability of second whorl organs of the d	http://antirrhinum.net/images/DragonDB/external/def-nic.jpg
http://lsrn.org/DragonDB_Allele:def-chl	Habit: Growth bushy. Leaves: Newly formed leaves pale green. C	http://antirrhinum.net/images/DragonDB/external/def-chlorantha.j

Because it is the Semantic *Web*
The query results are live hyperlinks
to the respective Database or images



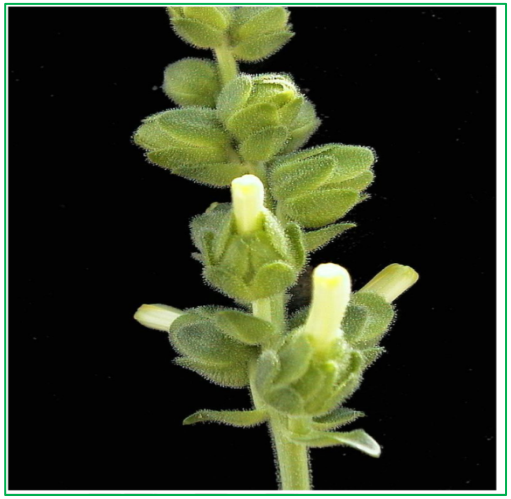
SPARQL query:

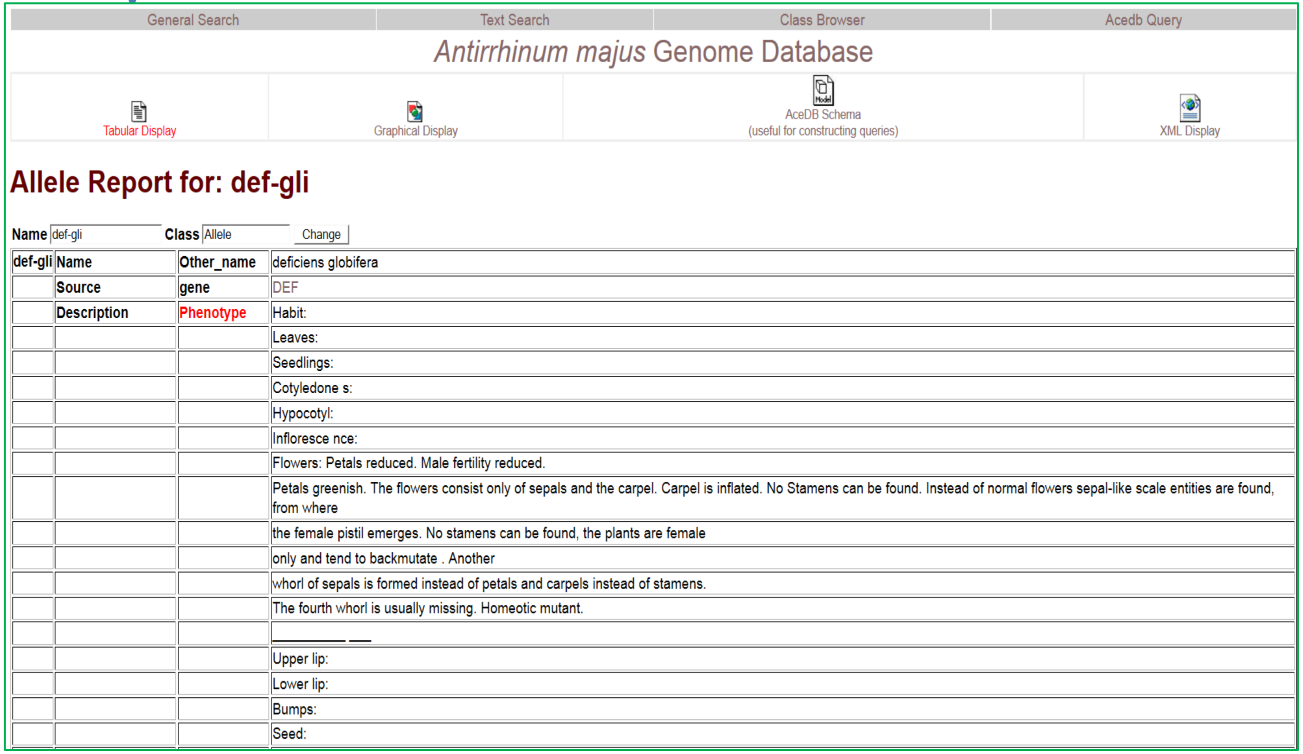
```
SELECT ?allele  ?image  ?desc
where {
     locus:DEF       genetics:hasVariant       ?allele .
          ?allele       info:visualizedByImage   ?image .
         ?image        info:hasDescription       ?desc
}
```

View results as RDF. There were warnings executing the query. Click for details.

Query results

allele	desc	image
http://lsrn.org/DragonDB_Allele:def-23	petals almost normal, third whorl similar to null mutant of def	http://antirrhinum.net/images/DragonDB/external/def-23.jpg
http://lsrn.org/DragonDB_Allele:def-101	temperature sensitivity of the def-101 allele Habit: Leaves: Seedl	http://antirrhinum.net/images/DragonDB/external/def-101.jpg
http://lsrn.org/DragonDB_Allele:def-gli	Habit: Leaves: Seedlings: Cotyledones: Hypocotyl: Inflorescence	http://antirrhinum.net/images/DragonDB/external/def-gli.jpg
http://lsrn.org/DragonDB_Allele:def-nic	backgound-dependent variability of second whorl organs of the d	http://antirrhinum.net/images/DragonDB/external/def-nic.jpg
http://lsrn.org/DragonDB_Allele:def-chl	Habit: Growth bushy. Leaves: Newly formed leaves pale green. C	http://antirrhinum.net/images/DragonDB/external/def-chlorantha.j





Antirrhinum majus Genome Database

Allele Report for: def-gli

Name def-gli Class Allele Change

def-gli	**Name**	**Other_name**	deficiens globifera
	Source	**gene**	DEF
	Description	**Phenotype**	Habit:
			Leaves:
			Seedlings:
			Cotyledone s:
			Hypocotyl:
			Infloresce nce:
			Flowers: Petals reduced. Male fertility reduced.
			Petals greenish. The flowers consist only of sepals and the carpel. Carpel is inflated. No Stamens can be found. Instead of normal flowers sepal-like scale entities are found, from where
			the female pistil emerges. No stamens can be found, the plants are female
			only and tend to backmutate . Another
			whorl of sepals is formed instead of petals and carpels instead of stamens.
			The fourth whorl is usually missing. Homeotic mutant.
			__________ ___
			Upper lip:
			Lower lip:
			Bumps:
			Seed:

Importantly

We posed, and answered a complex SPARQL query

without a SPARQL endpoint

(in fact, the data didn’t even have to exist...)

Example #2

Show me the latest Blood Urea Nitrogen and Creatinine levels of patients **who appear to be rejecting their transplants**

```
SELECT ?patient ?bun ?creat
FROM <http://sadiframework.org/ontologies/patients.rdf>
WHERE {
    ?patient rdf:type patient:LikelyRejecter .
    ?patient l:latestBUN ?bun .
    ?patient l:latestCreatinine ?creat .
}
```

Likely Rejecter:

A patient who has creatinine levels that are increasing over time

- - Wilkinson “MD”

Likely Rejecter:

Our triplestore contains various blood chemistry measurements at various time-points

Likely Rejecter:

…but there is no “likely rejecter” property in our triplestore

SHARE determines

by DL Reasoning

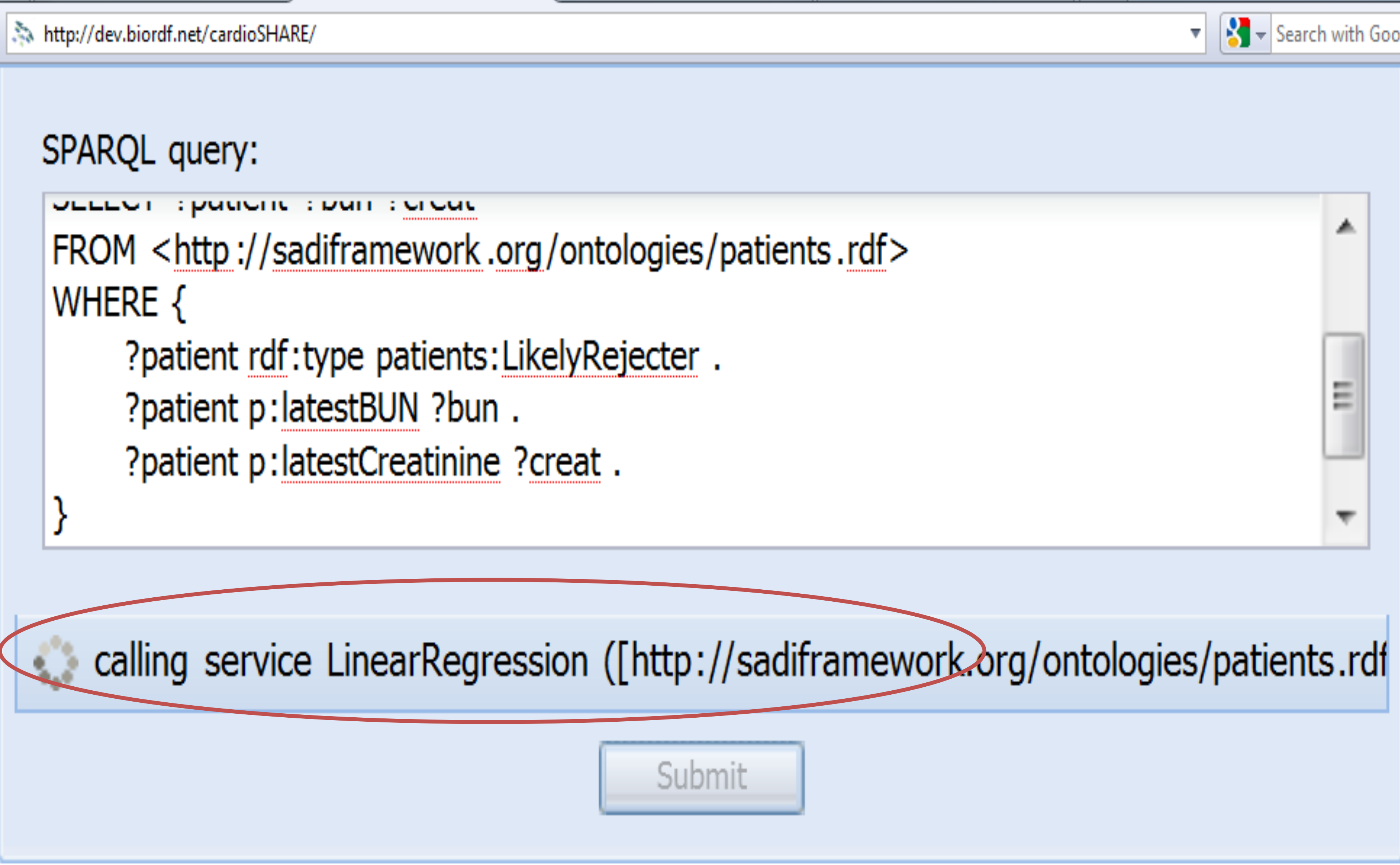
the **need** to do a
Linear Regression analysis over
Creatinine blood chemistry measurements

SHARE determines

by DL Reasoning

how and where that analysis
can be done

and orchestrates a workflow
that **does it**



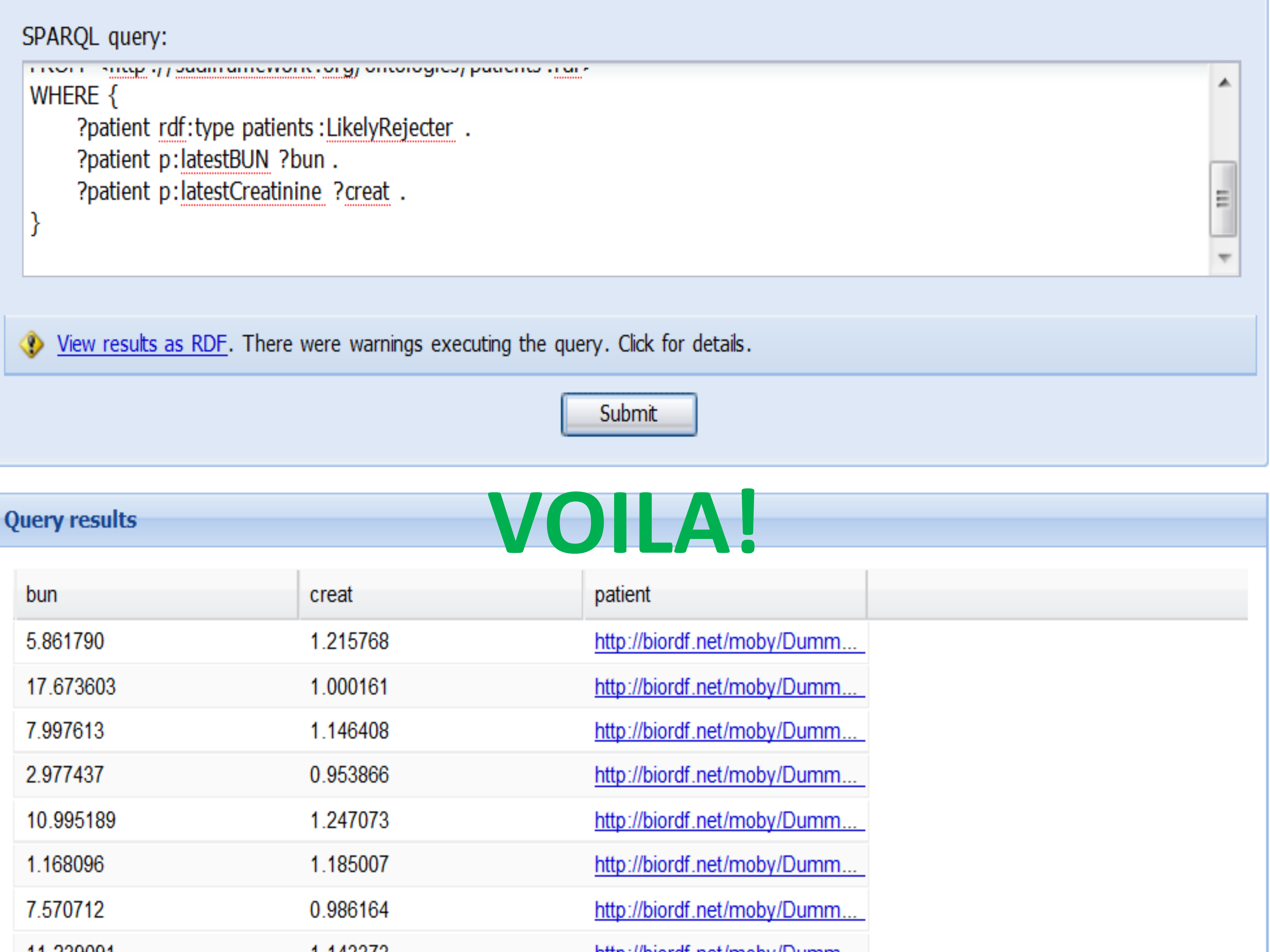
http://dev.biordf.net/cardioSHARE/

SPARQL query:

```
SELECT ?patient ?bun ?creat
FROM <http://sadiframework.org/ontologies/patients.rdf>
WHERE {
      ?patient rdf:type patients:LikelyRejecter .
      ?patient p:latestBUN ?bun .
      ?patient p:latestCreatinine ?creat .
}
```

calling service LinearRegression ([http://sadiframework.org/ontologies/patients.rdf

The SHARE system utilizes Semantics (via SADI) to discover and access analytical services on the Web that do linear regression analysis



SPARQL query:

```
FROM <http://sadiframework.org/ontologies/patients.rdf>
WHERE {
     ?patient rdf:type patients:LikelyRejecter .
     ?patient p:latestBUN ?bun .
     ?patient p:latestCreatinine ?creat .
}
```

View results as RDF. There were warnings executing the query. Click for details.

VOILA!

Query results

bun	creat	patient
5.861790	1.215768	http://biordf.net/moby/Dumm...
17.673603	1.000161	http://biordf.net/moby/Dumm...
7.997613	1.146408	http://biordf.net/moby/Dumm...
2.977437	0.953866	http://biordf.net/moby/Dumm...
10.995189	1.247073	http://biordf.net/moby/Dumm...
1.168096	1.185007	http://biordf.net/moby/Dumm...
7.570712	0.986164	http://biordf.net/moby/Dumm...
[illegible]	[illegible]	http://biordf.net/moby/Dumm...

SHARE formulated a path
(workflow)
to generate data *de novo*

because the data required by
the query didn’t exist

That’s enough for now

:-)

This Talk @

`http://tiny.cc/ldbc_sadi`

SADI

Find. Integrate.
Analyze.

SADI is an open-source initiative

(please forgive the chaos as we move from
Google Code to GitHub!)

http://sadiframework.org

*Mark Wilkinson markw@illuminae.
com*