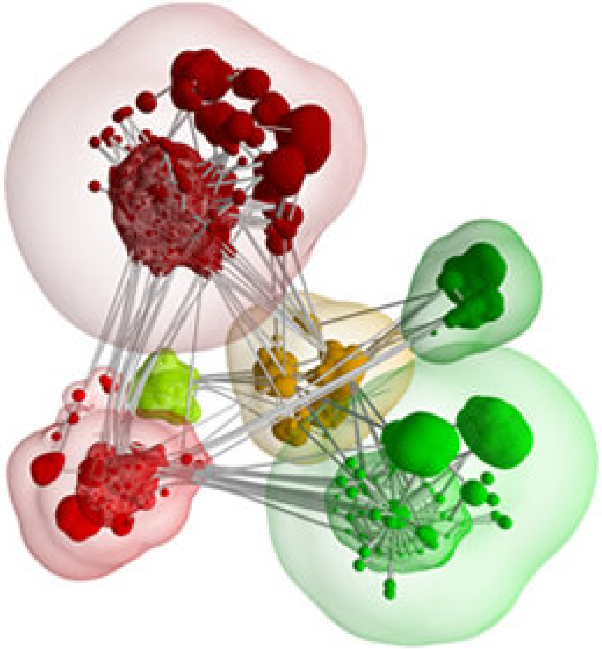
Optimizing Sparql and Prolog for reasoning on large scale diverse ontologies



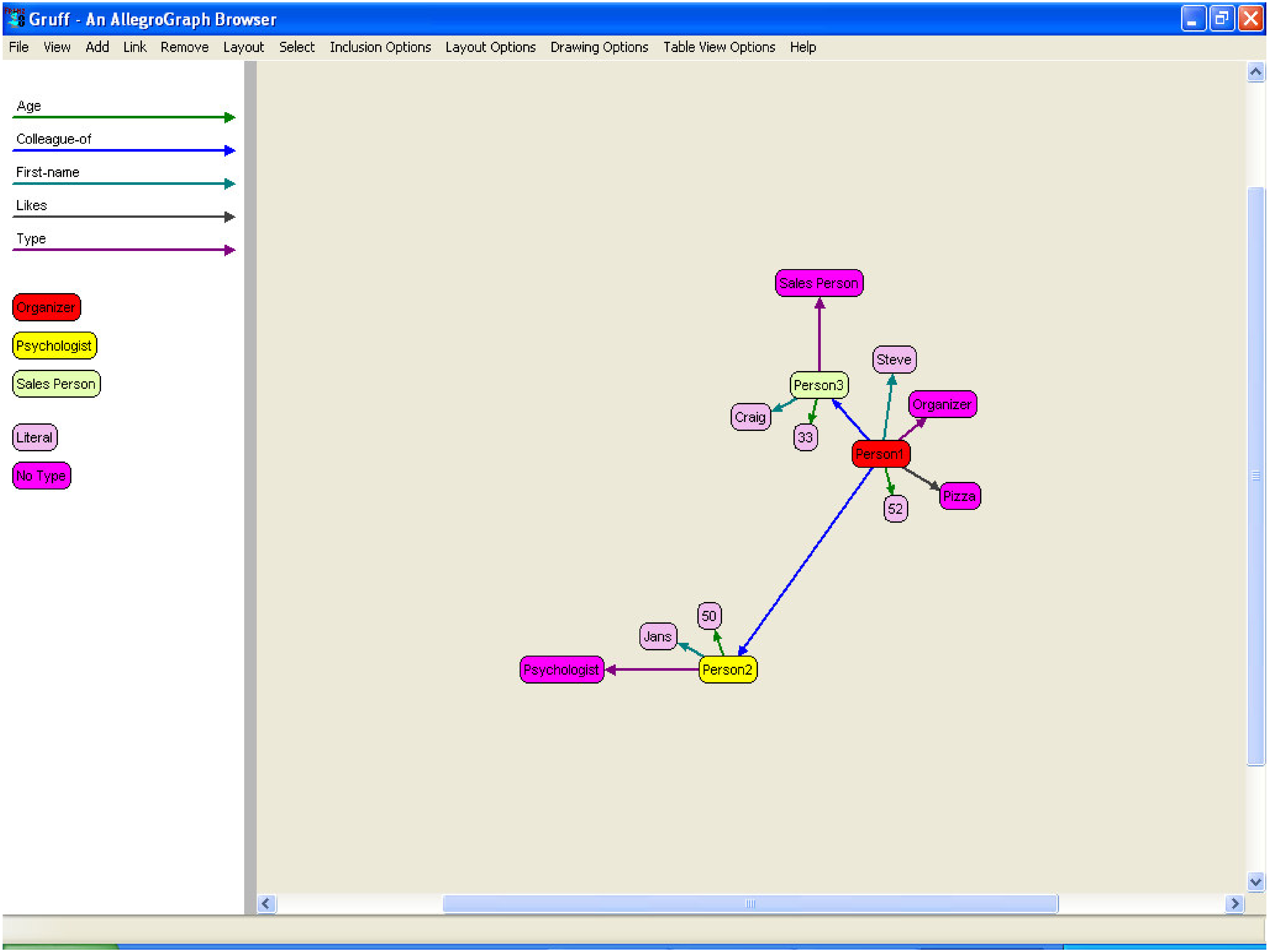
Jans Aasman, Ph.D.
CEO Franz Inc
Ja@Franz.com

This presentation

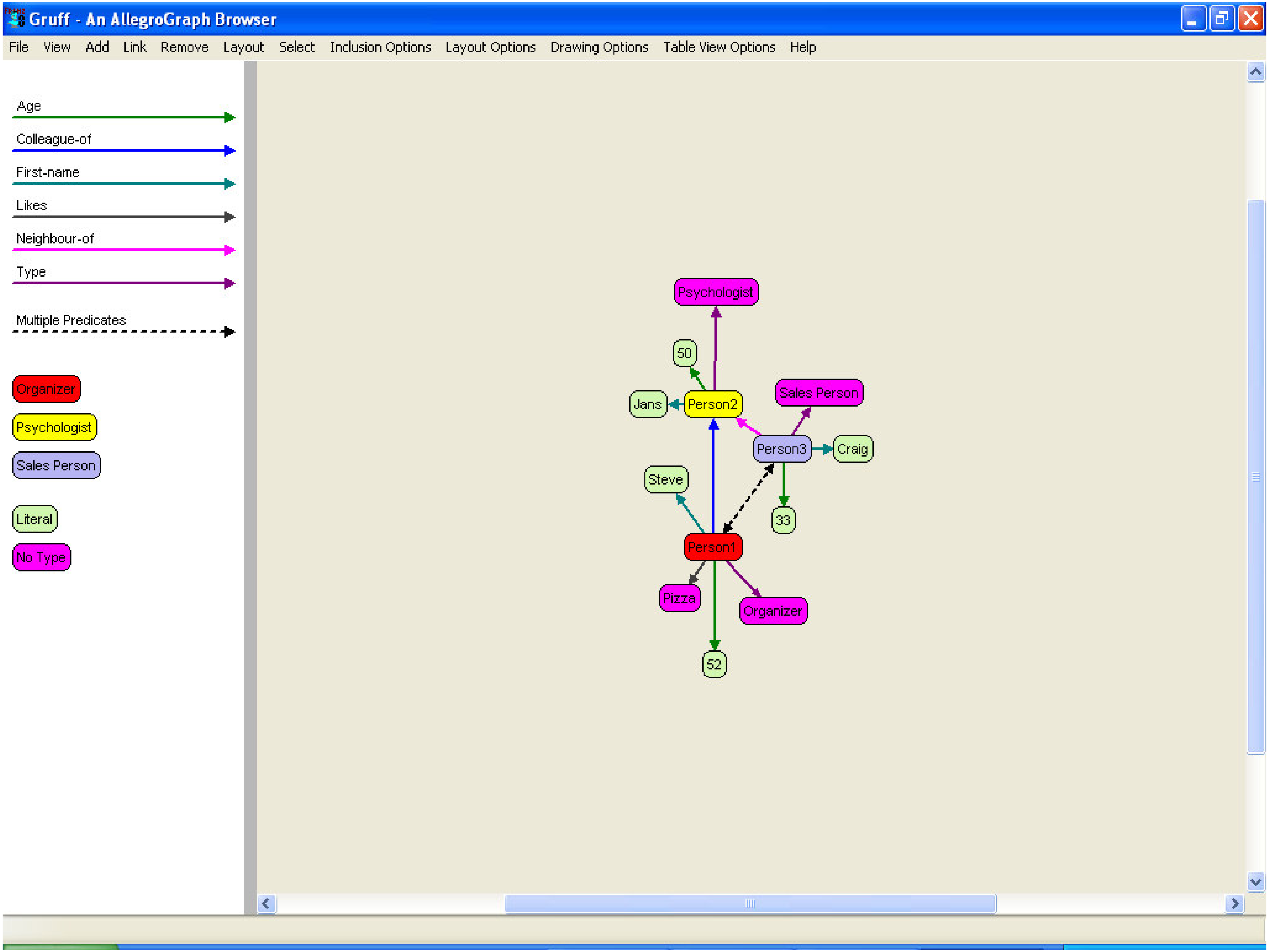
- Triples and a Graph database (2 minutes, I promise)
- AllegroGraph features
- Loading, indexing and querying: how did we do
- New numbers, 3.1 vs 3.2 and AllegroGraph vs (O)ther
- The secret sauce

Graphs, triples, triple-store?

```
createTripleStore(“seminar.db" )

addTriple (Person1 first-name Steve)
addTriple (Person1 isa  Organizer)
addTriple (Person1 age 52)
addTriple (Person2 first-name Jans)
addTriple (Person2 isa Psychologist)
addTriple (Person2 age 50)
addTriple (Person3 first-name Craig)
addTriple (Person3 isa SalesPerson)
addTriple (Person3 age 32)

addTriple (Person1 colleague-of Person2)
addTriple (Person1 colleague-of Person3)

addTriple (Person1 likes Pizza)
```


Gruff - An AllegroGraph Browser
File
View
Add
Link
Remove
Layout
Select
Inclusion Options
Layout Options
Drawing Options
Table View Options
Help
Age
Colleague-of
First-name
Likes
Type
Organizer
Psychologist
Sales Person
Literal
No Type
Sales Person
Steve
Person3
Organizer
Craig
33
Person1
Pizza
52
50
Jans
Psychologist
Person2

```
addTriple ( Person3 neighbor-of Person1)
addTriple ( Person3 neighbor-of Person2)
```


Gruff - An AllegroGraph Browser
File
View
Add
Link
Remove
Layout
Select
Inclusion Options
Layout Options
Drawing Options
Table View Options
Help
Age
Colleague-of
First-name
Likes
Neighbour-of
Type
Multiple Predicates
Organizer
Psychologist
Sales Person
Literal
No Type
Psychologist
50
Jans
Person2
Sales Person
Person3
Craig
Steve
33
Person1
Pizza
Organizer
52

And now you can query in Prolog or Sparql

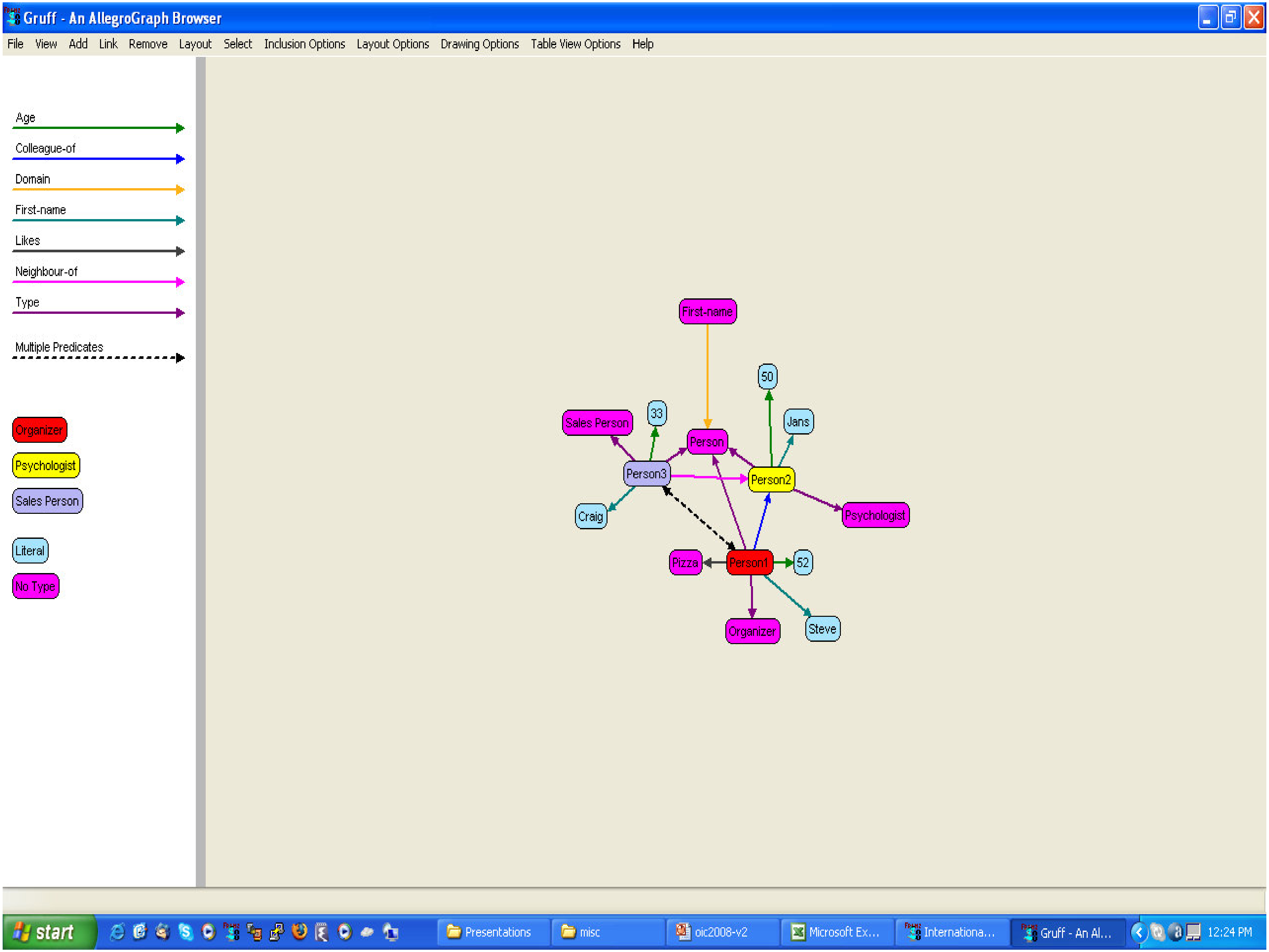
```
(select (?xname ?yname)
   (q ?x colleague-of ?y)
   (q ?y neighbor-of ?x)
   (q ?x first-name ?xname)
   (q ?y first-name ?yname))
```

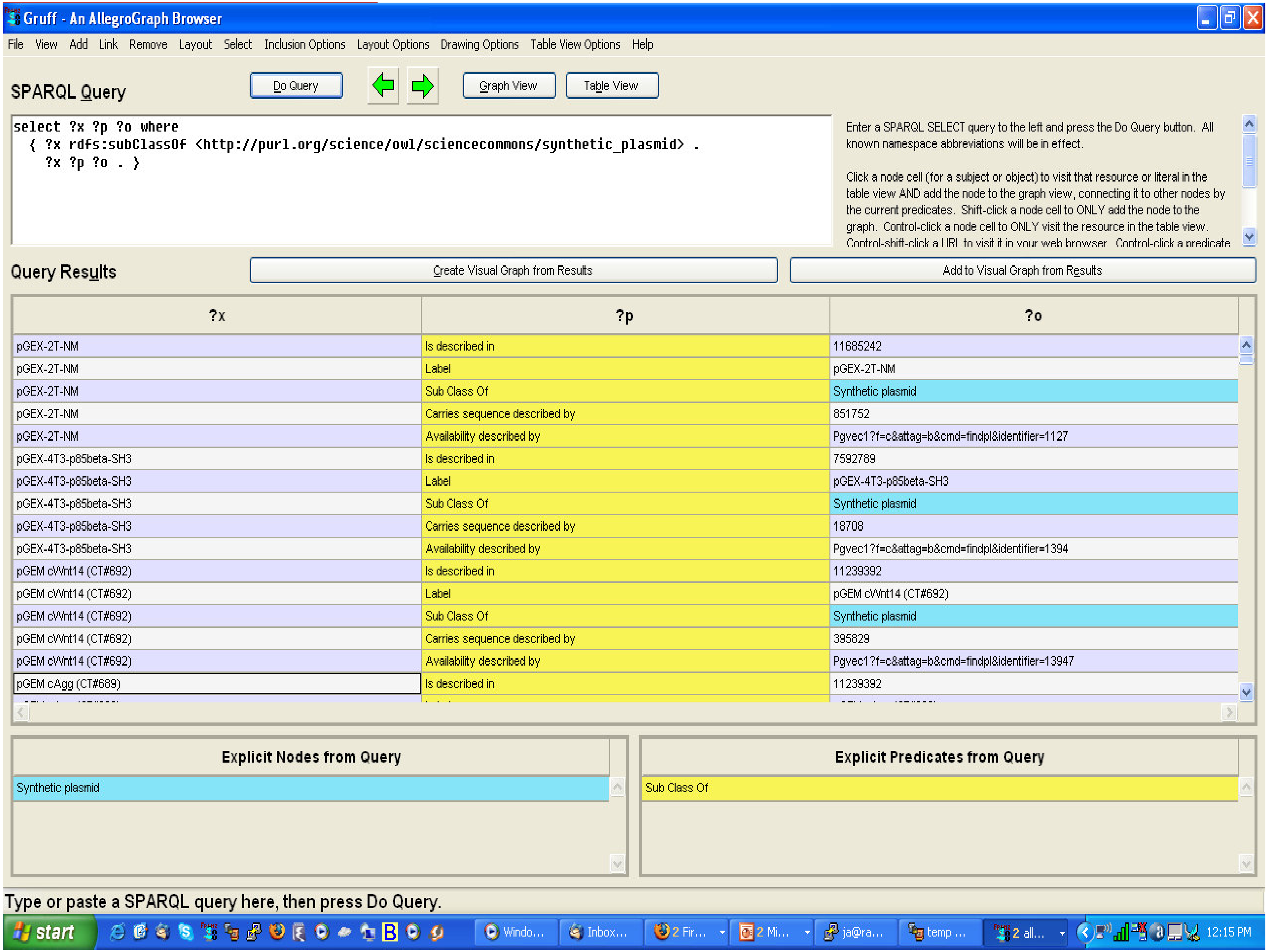
```
SELECT ?xname ?yname WHERE {
     ?x ex:colleague-of ?y .
     ?y ex:neighbor-of ?x .
     ?x ex:first-name ?xname .
     ?y ex:first-name ?yname . }
```

Or reason

```
addTriple ( first-name domain Person)
```

```
Every subject that has a predicate ‘first-name’
must be of type Person.
```


Gruff - An AllegroGraph Browser
File
View
Add
Link
Remove
Layout
Select
Inclusion Options
Layout Options
Drawing Options
Table View Options
Help
Age
Colleague-of
Domain
First-name
Likes
Neighbour-of
Type
Multiple Predicates
Organizer
Psychologist
Sales Person
Literal
No Type
First-name
50
33
Sales Person
Jans
Person
Person3
Person2
Psychologist
Craig
Pizza
Person1
52
Organizer
Steve
start
Presentations
misc
oic2008-v2
Microsoft Ex...
Internationa...
Gruff - An Al...
12:24 PM



SPARQL Query

```
select ?x ?p ?o where
  { ?x rdfs:subClassOf <http://purl.org/science/owl/sciencecommons/synthetic_plasmid> .
    ?x ?p ?o . }
```

Enter a SPARQL SELECT query to the left and press the Do Query button. All known namespace abbreviations will be in effect.

Click a node cell (for a subject or object) to visit that resource or literal in the table view AND add the node to the graph view, connecting it to other nodes by the current predicates. Shift-click a node cell to ONLY add the node to the graph. Control-click a node cell to ONLY visit the resource in the table view. Control-shift-click a URL to visit it in your web browser. Control-click a predicate

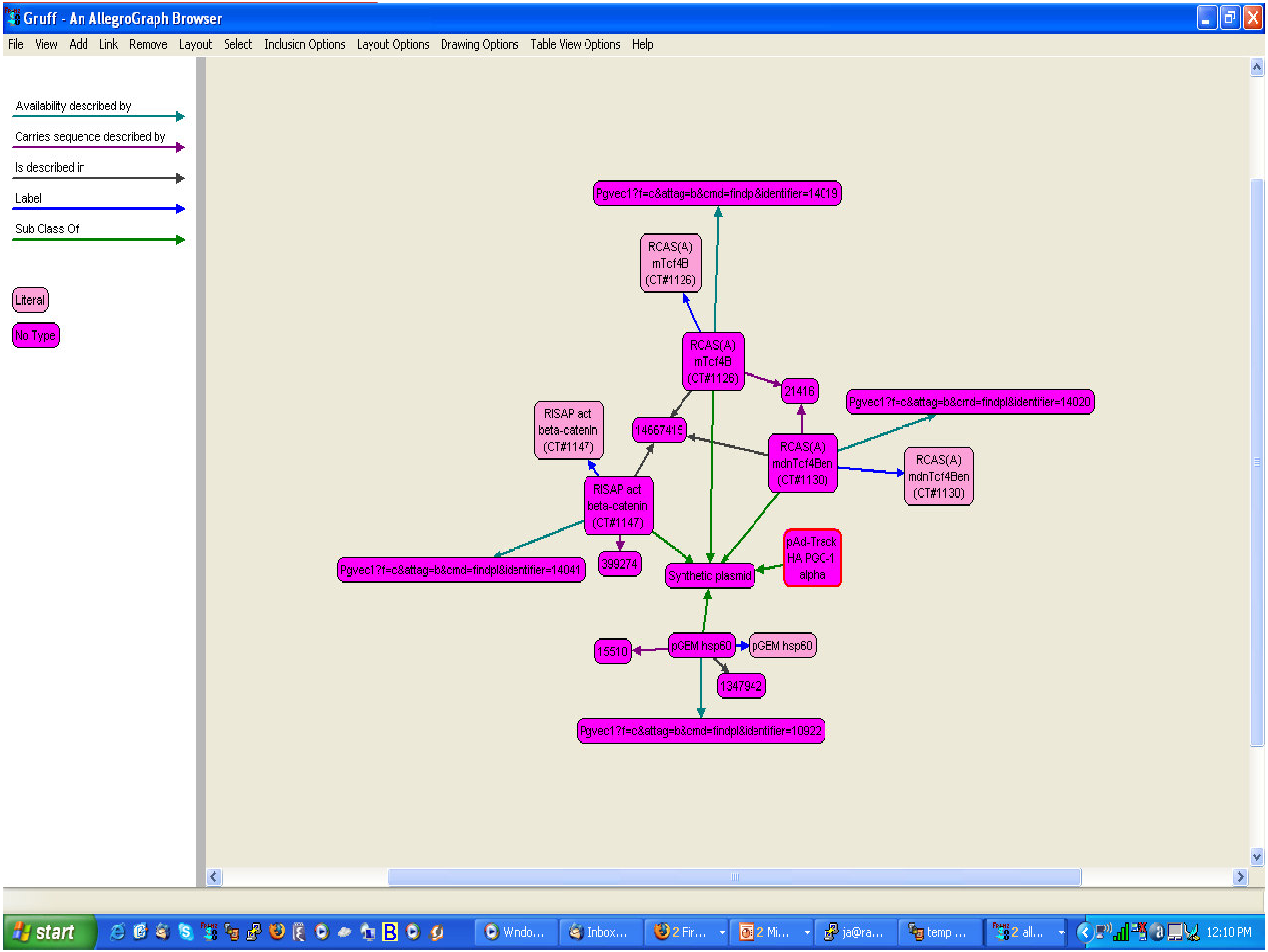
Query Results

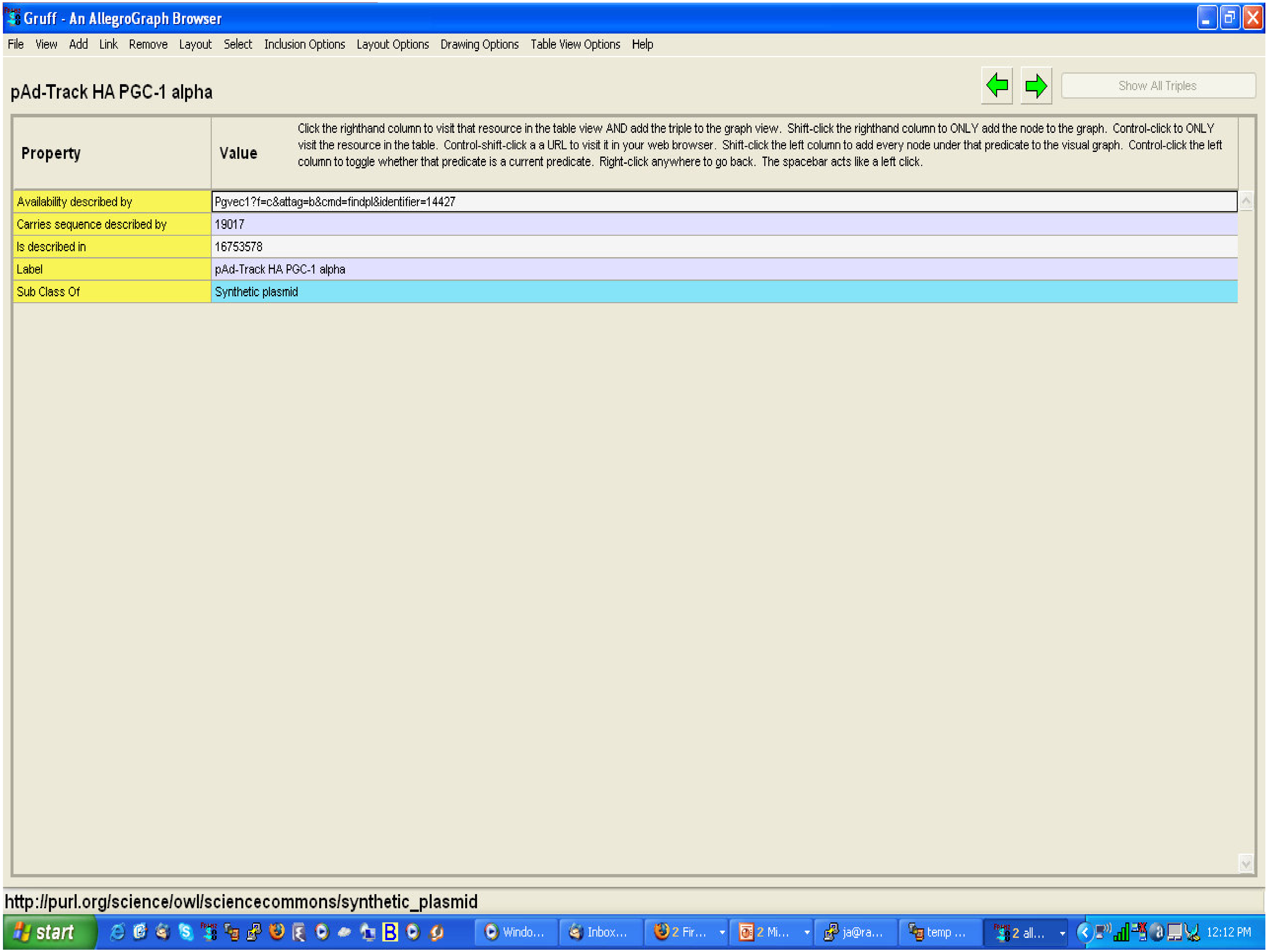
?x	?p	?o
pGEX-2T-NM	Is described in	11685242
pGEX-2T-NM	Label	pGEX-2T-NM
pGEX-2T-NM	Sub Class Of	Synthetic plasmid
pGEX-2T-NM	Carries sequence described by	851752
pGEX-2T-NM	Availability described by	Pgvec1?f=c&attag=b&cmd=findpl&identifier=1127
pGEX-4T3-p85beta-SH3	Is described in	7592789
pGEX-4T3-p85beta-SH3	Label	pGEX-4T3-p85beta-SH3
pGEX-4T3-p85beta-SH3	Sub Class Of	Synthetic plasmid
pGEX-4T3-p85beta-SH3	Carries sequence described by	18708
pGEX-4T3-p85beta-SH3	Availability described by	Pgvec1?f=c&attag=b&cmd=findpl&identifier=1394
pGEM cWnt14 (CT#692)	Is described in	11239392
pGEM cWnt14 (CT#692)	Label	pGEM cWnt14 (CT#692)
pGEM cWnt14 (CT#692)	Sub Class Of	Synthetic plasmid
pGEM cWnt14 (CT#692)	Carries sequence described by	395829
pGEM cWnt14 (CT#692)	Availability described by	Pgvec1?f=c&attag=b&cmd=findpl&identifier=13947
pGEM cAgg (CT#689)	Is described in	11239392

Explicit Nodes from Query
Synthetic plasmid

Explicit Predicates from Query
Sub Class Of

Type or paste a SPARQL query here, then press Do Query.


Gruff - An AllegroGraph Browser
File View Add Link Remove Layout Select Inclusion Options Layout Options Drawing Options Table View Options Help
Availability described by
Carries sequence described by
Is described in
Label
Sub Class Of
Literal
No Type
Pgvec1?f=c&attag=b&cmd=findpl&identifier=14019
RCAS(A) mTcf4B (CT#1126)
RCAS(A) mTcf4B (CT#1126)
21416
Pgvec1?f=c&attag=b&cmd=findpl&identifier=14020
RISAP act beta-catenin (CT#1147)
14667415
RCAS(A) mdnTcf4Ben (CT#1130)
RCAS(A) mdnTcf4Ben (CT#1130)
RISAP act beta-catenin (CT#1147)
pAd-Track HA PGC-1 alpha
Pgvec1?f=c&attag=b&cmd=findpl&identifier=14041
399274
Synthetic plasmid
15510
pGEM hsp60
pGEM hsp60
1347942
Pgvec1?f=c&attag=b&cmd=findpl&identifier=10922
start
Windo...
Inbox...
2 Fir...
2 Mi...
ja@ra...
temp ...
2 all...
12:10 PM

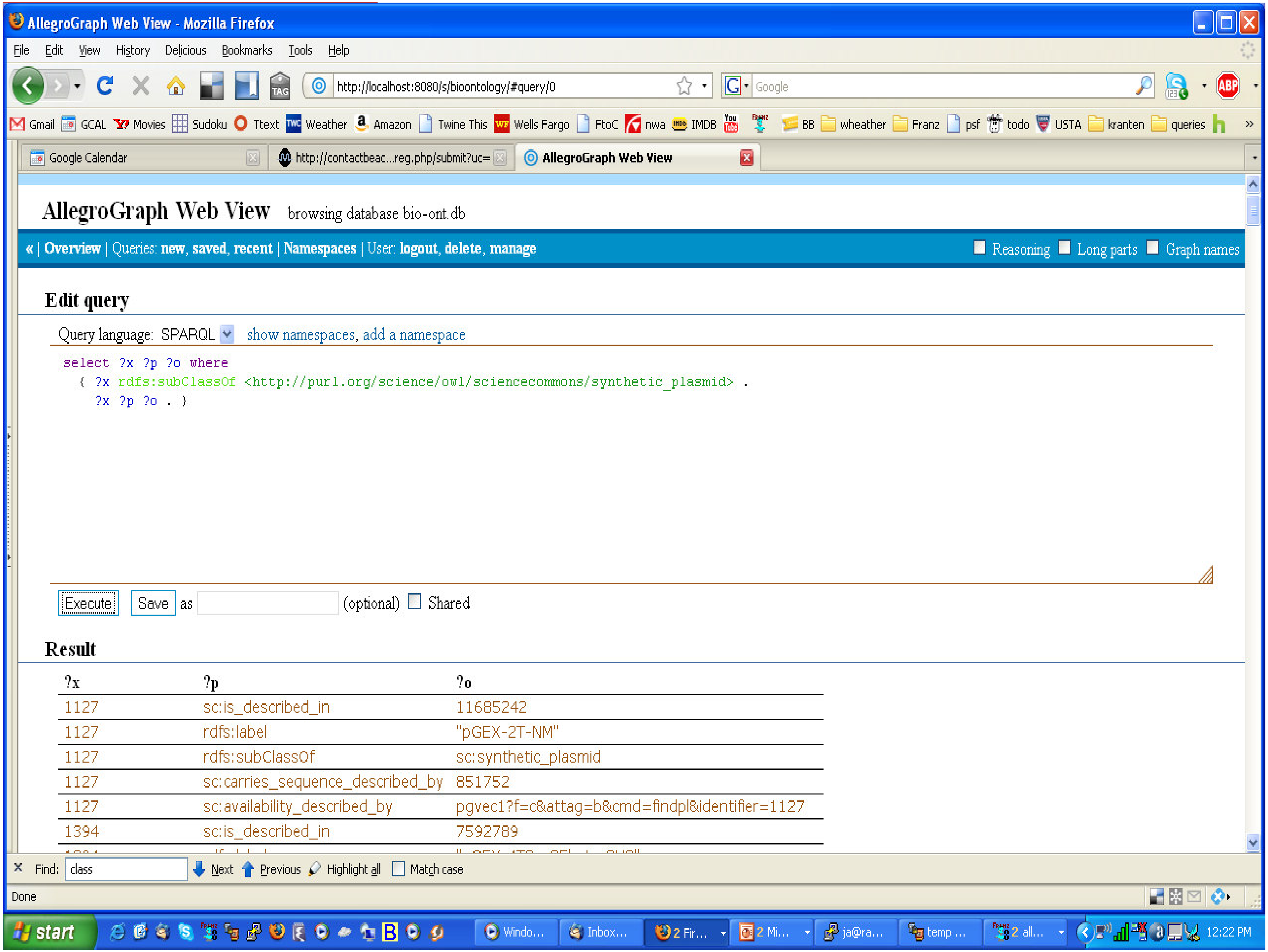


pAd-Track HA PGC-1 alpha

Click the righthand column to visit that resource in the table view AND add the triple to the graph view. Shift-click the righthand column to ONLY add the node to the graph. Control-click to ONLY visit the resource in the table. Control-shift-click a a URL to visit it in your web browser. Shift-click the left column to add every node under that predicate to the visual graph. Control-click the left column to toggle whether that predicate is a current predicate. Right-click anywhere to go back. The spacebar acts like a left click.

Property	Value
Availability described by	Pgvec1?f=c&attag=b&cmd=findpl&identifier=14427
Carries sequence described by	19017
Is described in	16753578
Label	pAd-Track HA PGC-1 alpha
Sub Class Of	Synthetic plasmid

http://purl.org/science/owl/sciencecommons/synthetic_plasmid


AllegroGraph Web View - Mozilla Firefox
File Edit View History Delicious Bookmarks Tools Help
http://localhost:8080/s/bioontology/#query/0
Google
Gmail GCAL Movies Sudoku Ttext Weather Amazon Twine This Wells Fargo FtoC nwa IMDB BB wheather Franz psf todo USTA kranten queries
Google Calendar
http://contactbeac...reg.php/submit?uc=
AllegroGraph Web View
« | Overview | Queries: new, saved, recent | Namespaces | User: logout, delete, manage
□ Reasoning □ Long parts □ Graph names
Done
start
Windo... Inbox... 2 Fir... 2 Mi... ja@ra... temp ... 2 all...
12:22 PM

AllegroGraph Web View

browsing database bio-ont.db

Edit query

Query language: SPARQL show namespaces, add a namespace

```
select ?x ?p ?o where
  { ?x rdfs:subClassOf <http://purl.org/science/owl/sciencecommons/synthetic_plasmid> .
    ?x ?p ?o . }
```

Execute Save as (optional) □ Shared

Result

?x	?p	?o
1127	sc:is_described_in	11685242
1127	rdfs:label	"pGEX-2T-NM"
1127	rdfs:subClassOf	sc:synthetic_plasmid
1127	sc:carries_sequence_described_by	851752
1127	sc:availability_described_by	pgvec1?f=c&attag=b&cmd=findpl&identifier=1127
1394	sc:is_described_in	7592789

Find: class Next Previous Highlight all □ Match case

AllegroGraph [1]

- Scalable and persistent Triple Store
  - Loads a 1.1 Billion triples in 20 hours on a single CPU and 8 hours on a 4 processor AMD machine (in federation)
- Federated
  - Create an abstract store that is a collection of other triple stores. Prolog and SPARQL and Reasoning work transparently against abstract store
- Compliant with standards
  - RDF, RDFS, OWL, SPARQL, Named Graphs, ISO Prolog, OWL-lite reasoning
- RDFS++ reasoner:
  - All of RDFS, inverseOf, sameAs, hasValue, transitiveProperty
- Full text indexing
- Java (Jena/Sesame) and Python interface.

AllegroGraph [2]

- Relational database efficiency for range queries
  - We support most xml schema types (dates, times, longitudes, latitudes, durations, telephone numbers, etc)
- Spatial database efficiency for geospatial primitives
  - Find elements in bounding boxes as fast as in spatial databases
- Temporal reasoning
  - Reasoning about times and intervals (Allen Logic)
- Social Network Analytics library
  - Find actor degrees and centrality, cliques, group centrality and cohesiveness

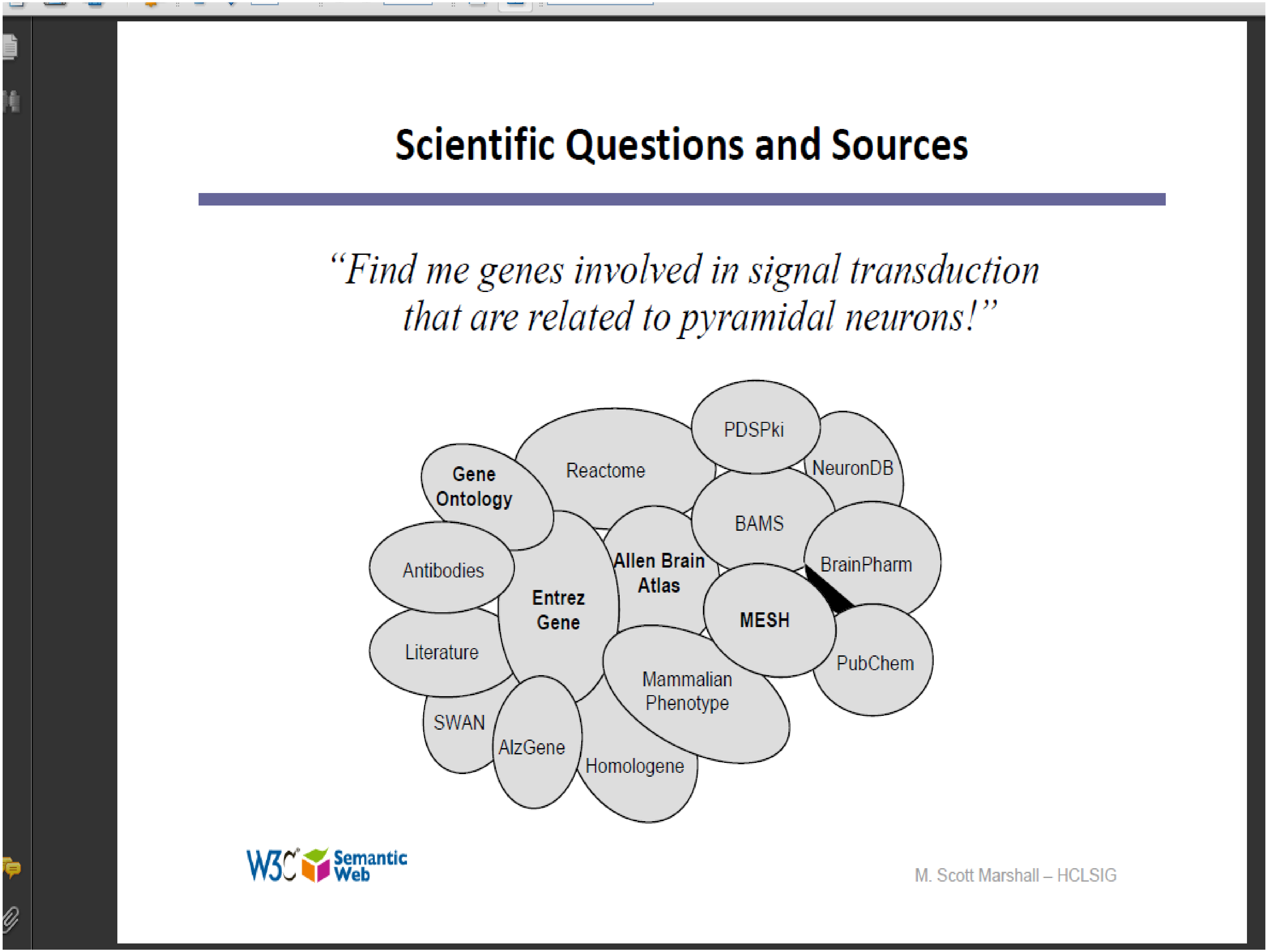
So how were we doing

- We were very fast at loading and indexing

- But queries on a reasoning store were slower then we wanted

Datasets we work with

- Science Commons (350,000,000 triples….)
- Linked Data (1,400,000,000 triples)
- LUBM8000 (1,200,000,000 triples)

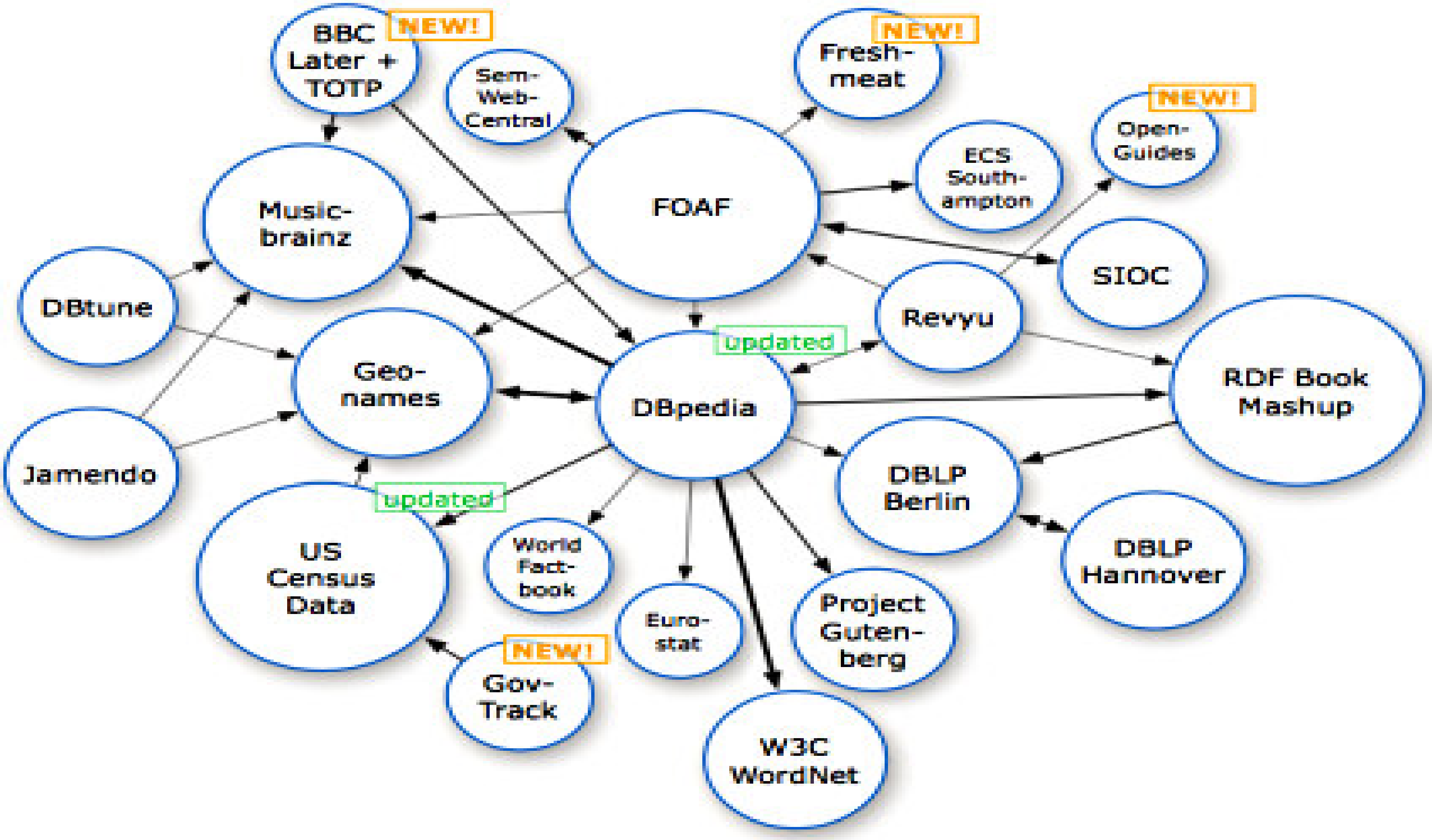


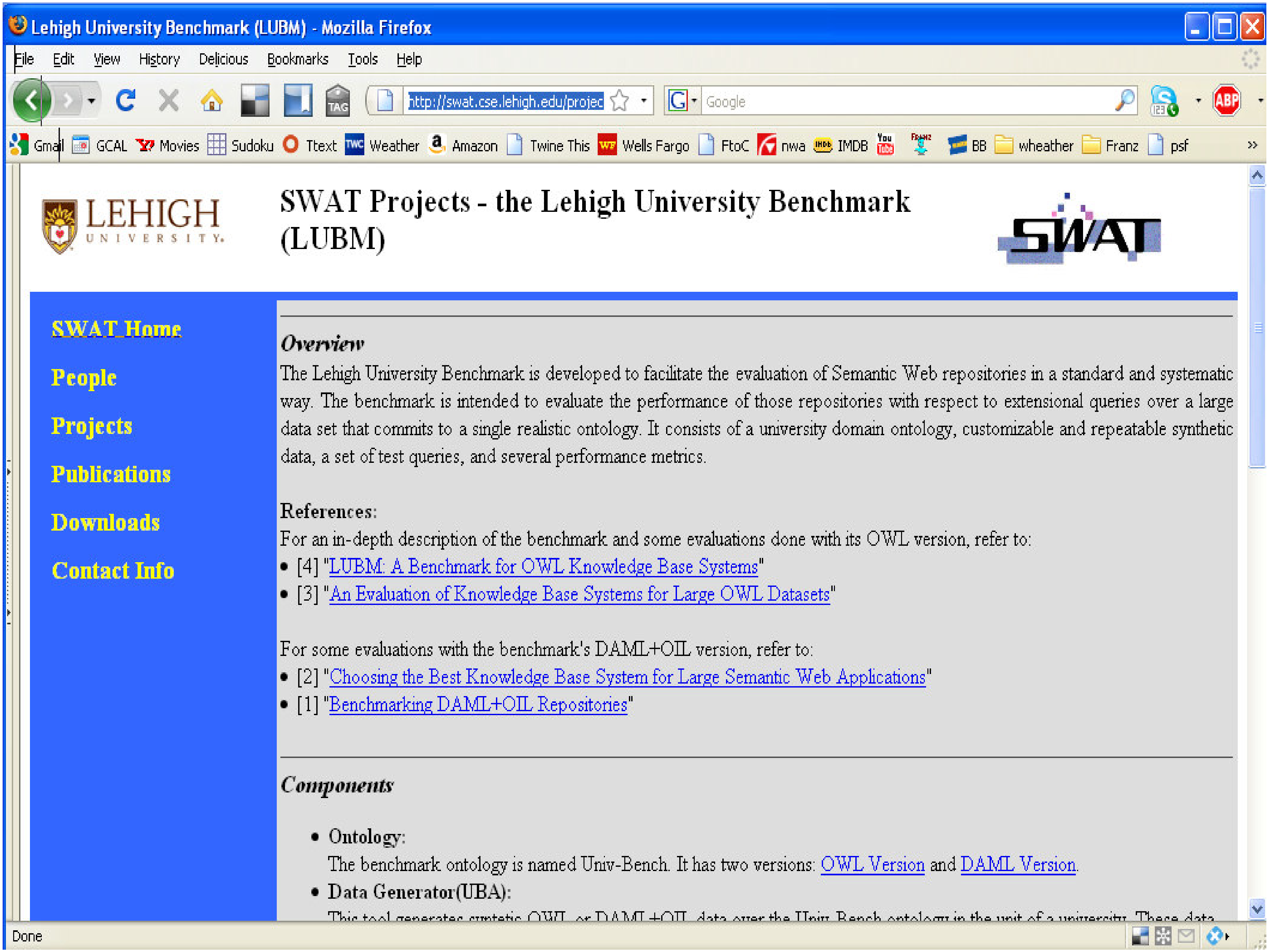
Scientific Questions and Sources

“Find me genes involved in signal transduction that are related to pyramidal neurons!”

M. Scott Marshall – HCLSIG

Find the socio-economic indicators for the place where Obama was born




Lehigh University Benchmark (LUBM) - Mozilla Firefox
File Edit View History Delicious Bookmarks Tools Help
http://swat.cse.lehigh.edu/projec
Google
Gmail GCAL Movies Sudoku Ttext Weather Amazon Twine This Wells Fargo FtoC nwa IMDB BB wheather Franz psf
LEHIGH UNIVERSITY.
SWAT
Done

SWAT Projects - the Lehigh University Benchmark (LUBM)

SWAT Home
People
Projects
Publications
Downloads
Contact Info

Overview

The Lehigh University Benchmark is developed to facilitate the evaluation of Semantic Web repositories in a standard and systematic way. The benchmark is intended to evaluate the performance of those repositories with respect to extensional queries over a large data set that commits to a single realistic ontology. It consists of a university domain ontology, customizable and repeatable synthetic data, a set of test queries, and several performance metrics.

References:

For an in-depth description of the benchmark and some evaluations done with its OWL version, refer to:

- [4] "LUBM: A Benchmark for OWL Knowledge Base Systems"
- [3] "An Evaluation of Knowledge Base Systems for Large OWL Datasets"

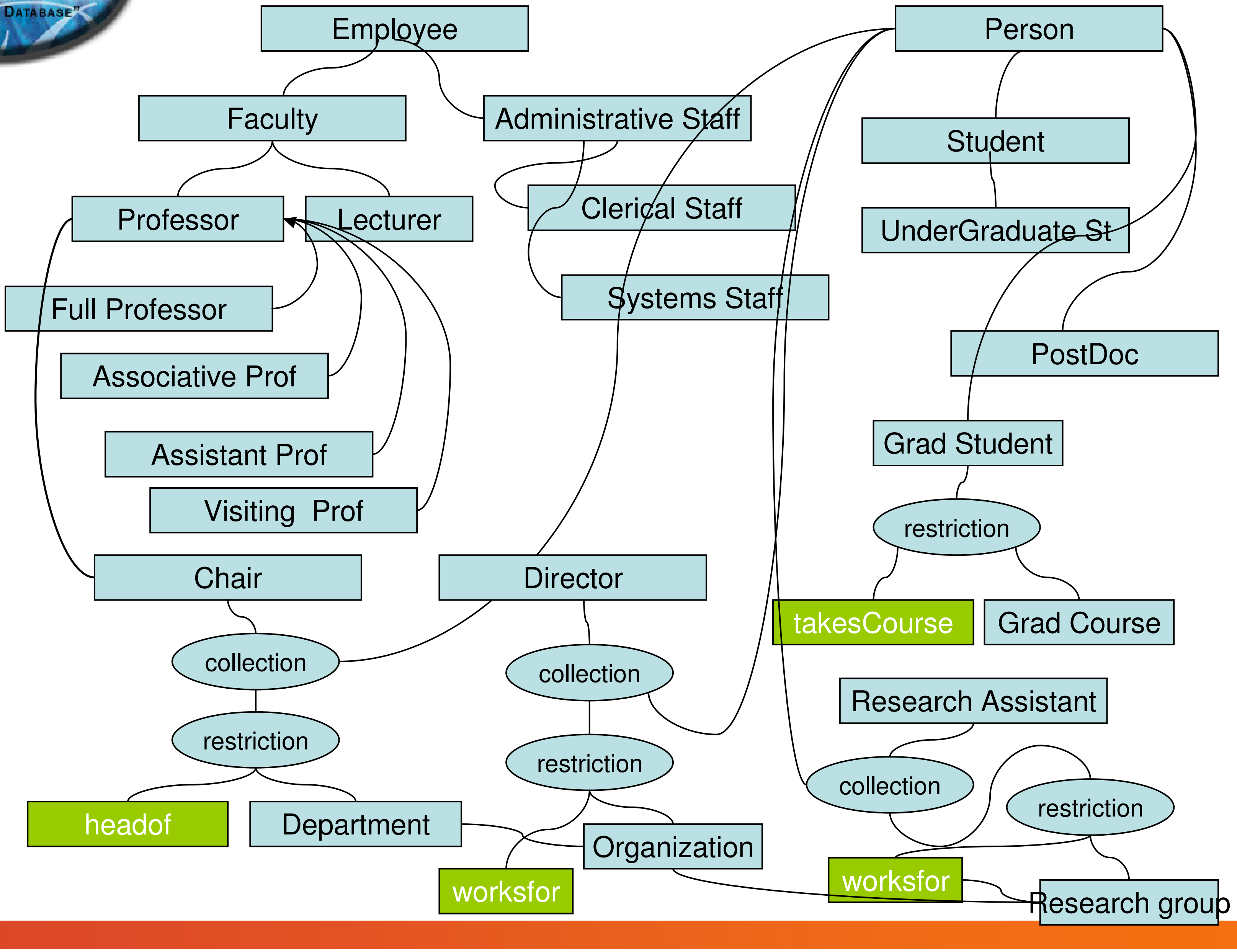
For some evaluations with the benchmark's DAML+OIL version, refer to:

- [2] "Choosing the Best Knowledge Base System for Large Semantic Web Applications"
- [1] "Benchmarking DAML+OIL Repositories"

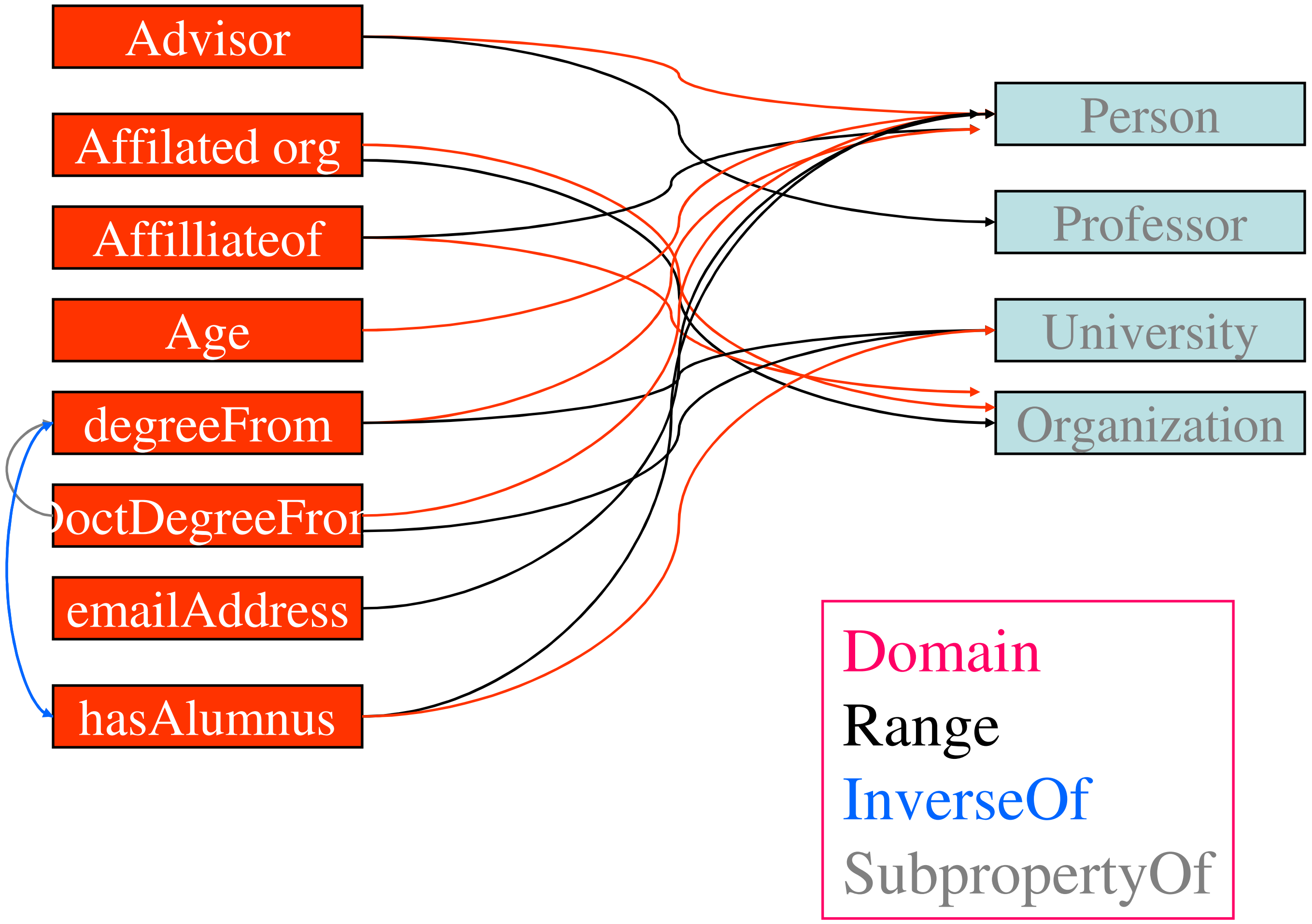
Components

- **Ontology:**
  The benchmark ontology is named Univ-Bench. It has two versions: OWL Version and DAML Version.
- **Data Generator(UBA):**
  This tool generates synthetic OWL or DAML+OIL data over the Univ-Bench ontology in the unit of a university. These data

A small part of the class hierarchy of LUBM



A small part of the property descriptions of LUBM

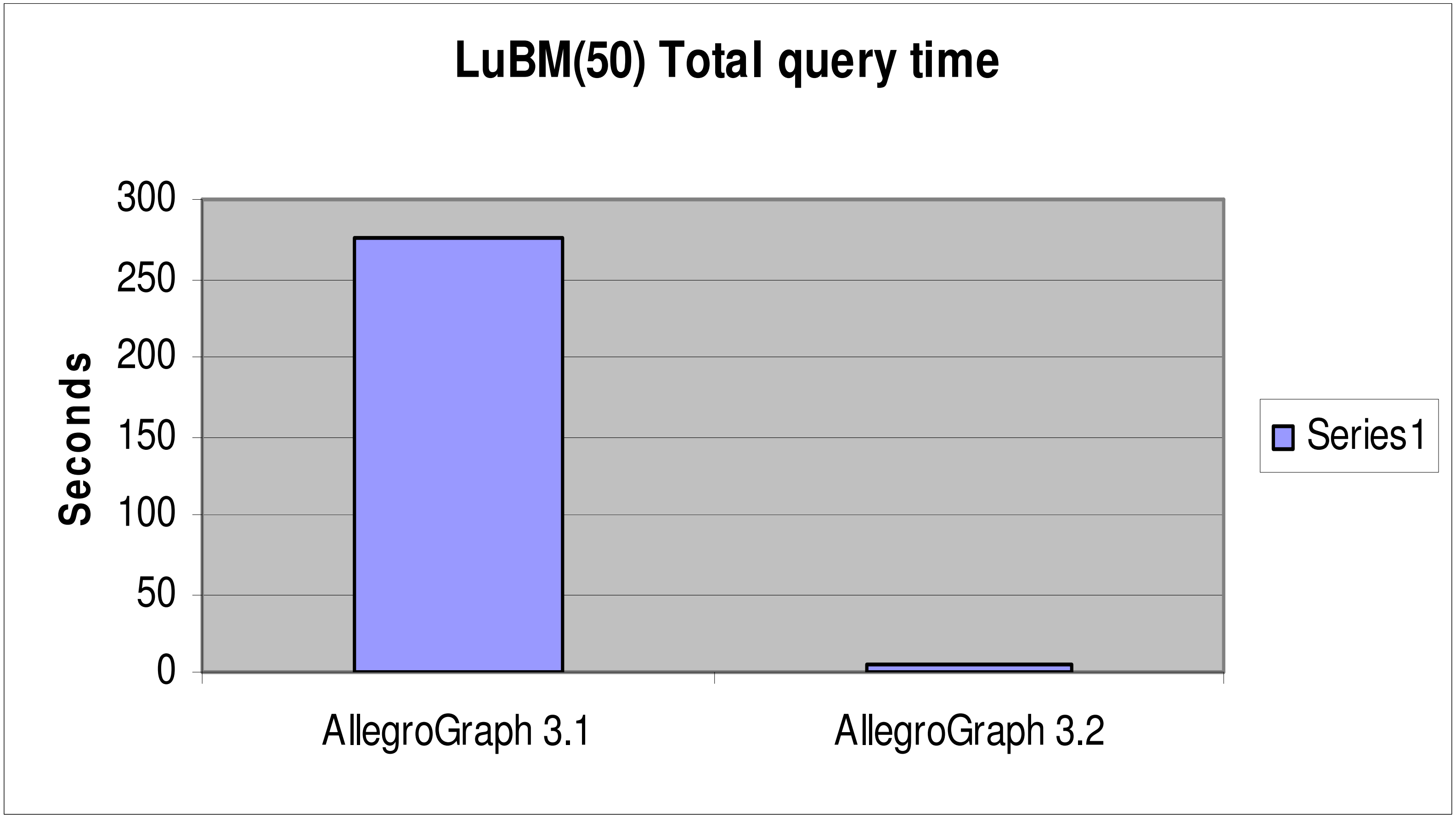


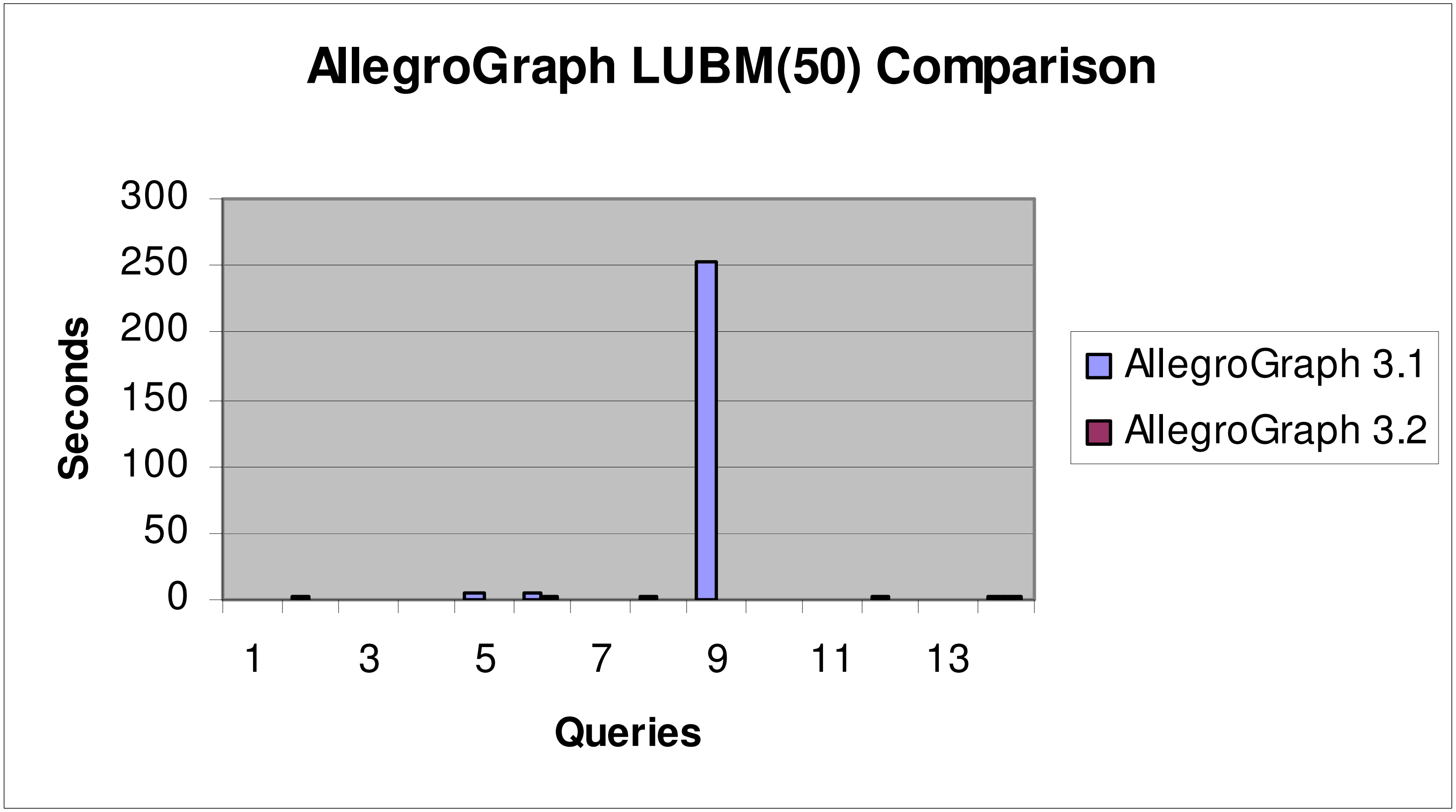
Our LUBM Benchmarks..

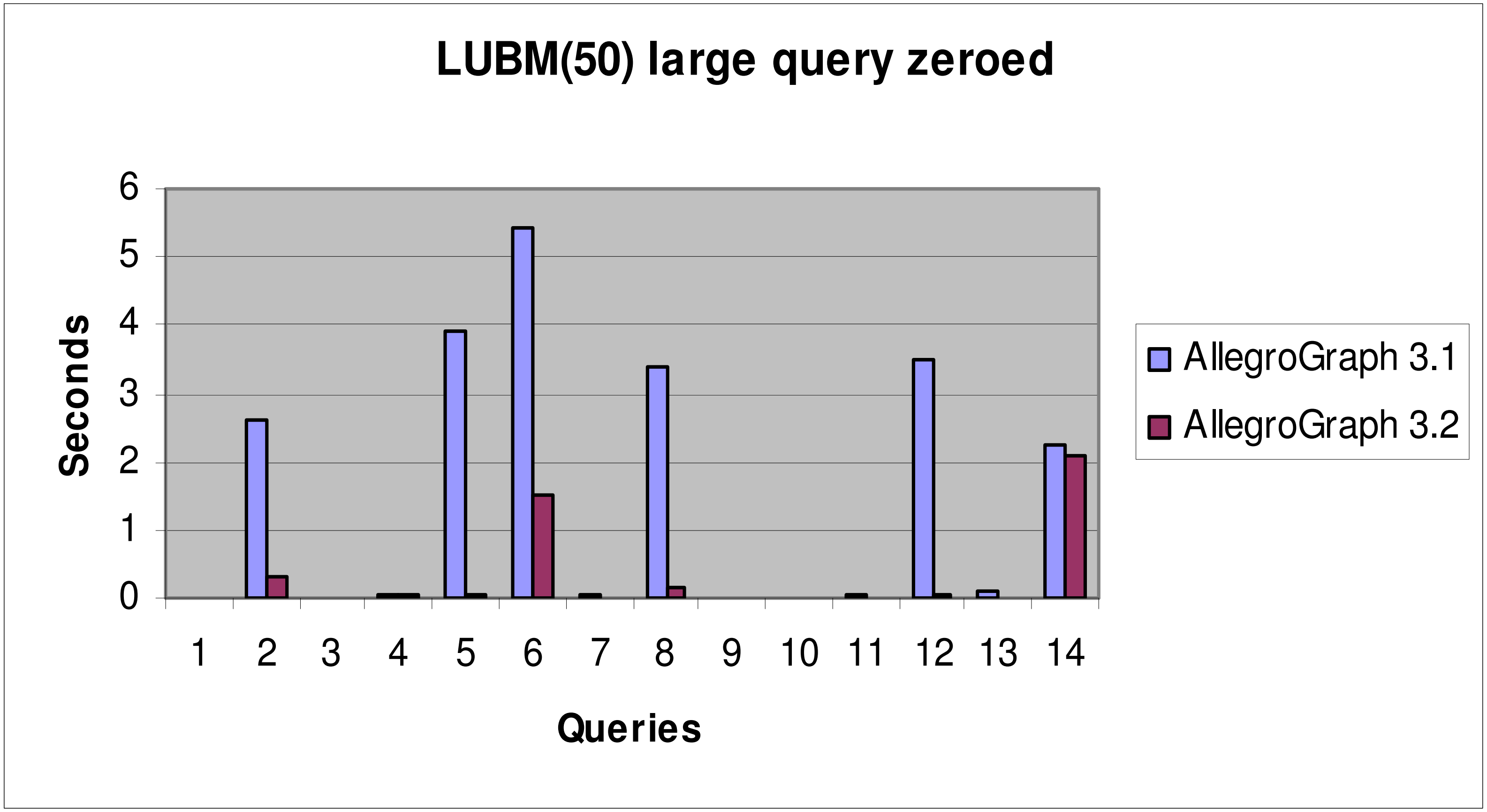
- Lubm 50 => 7,000,000 triples
- Lubm 8000 => 1,100,000,000 triples

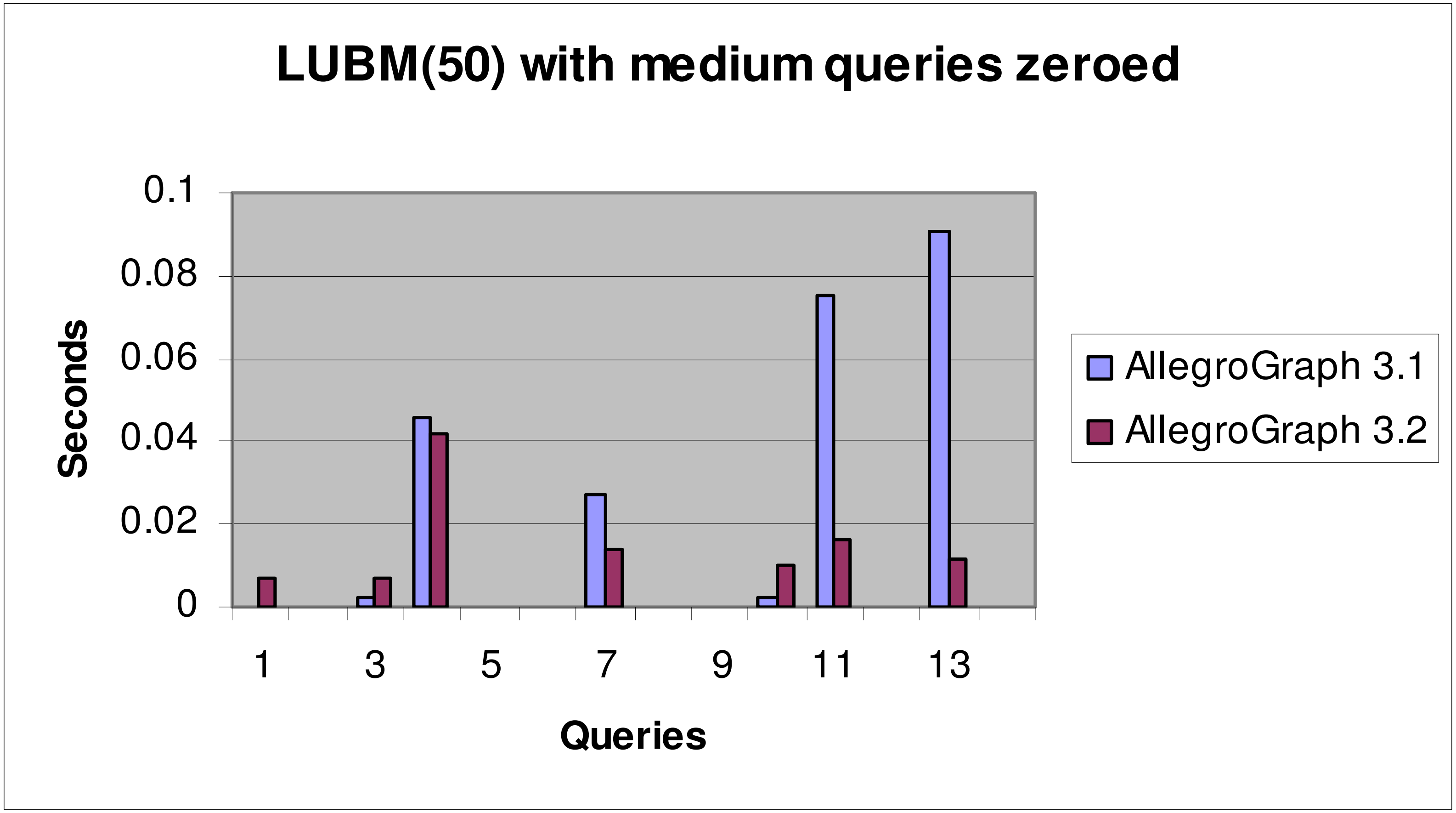
- We use a 4 processor, 1.8 GHz, 16 Gig machine with 64 bit Fedora Core.

- We compare 3.1 against 3.2
- And (O)ther against 3.2

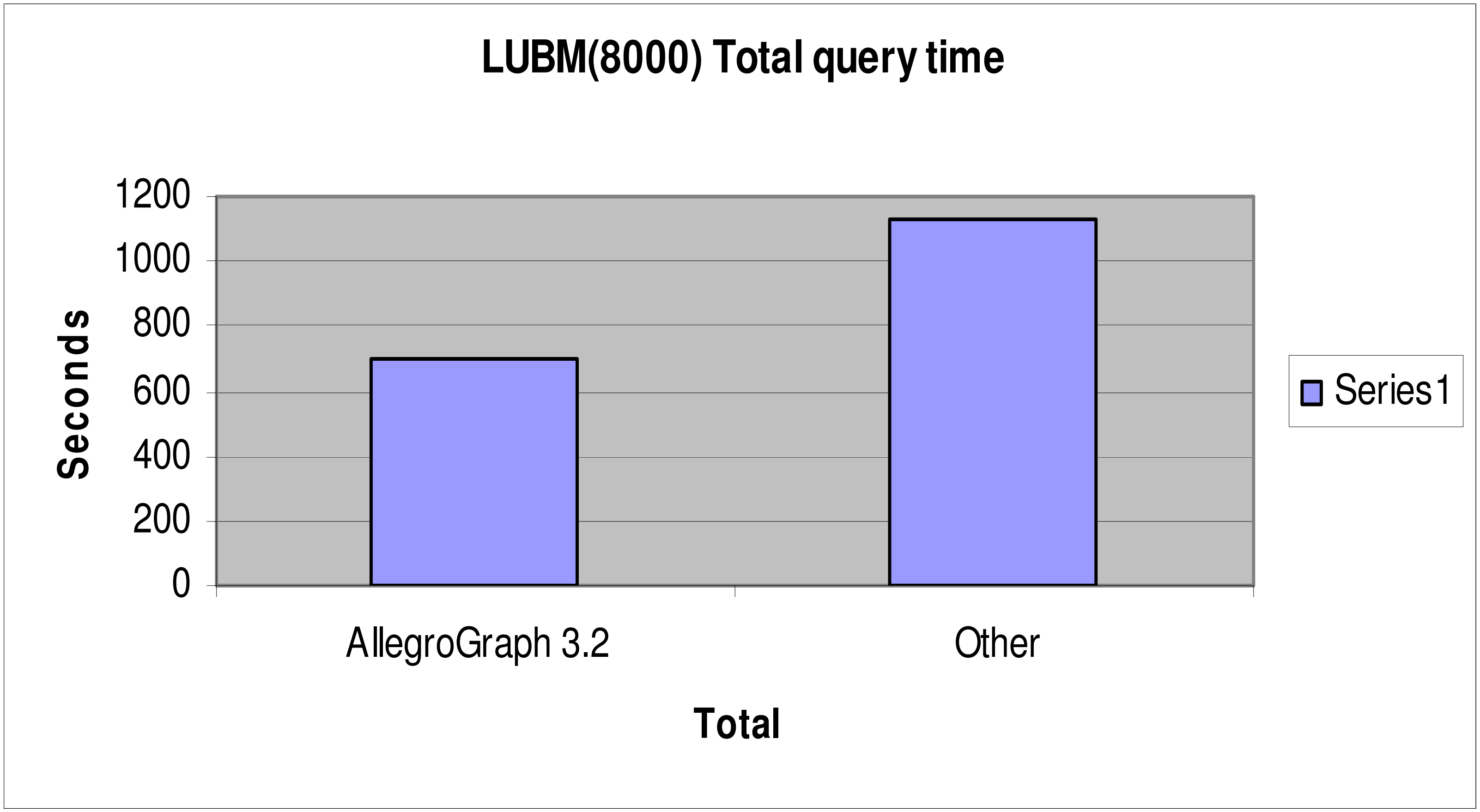

LuBM(50) Total query time
Seconds
300
250
200
150
100
50
0
AllegroGraph 3.1
AllegroGraph 3.2
Series1

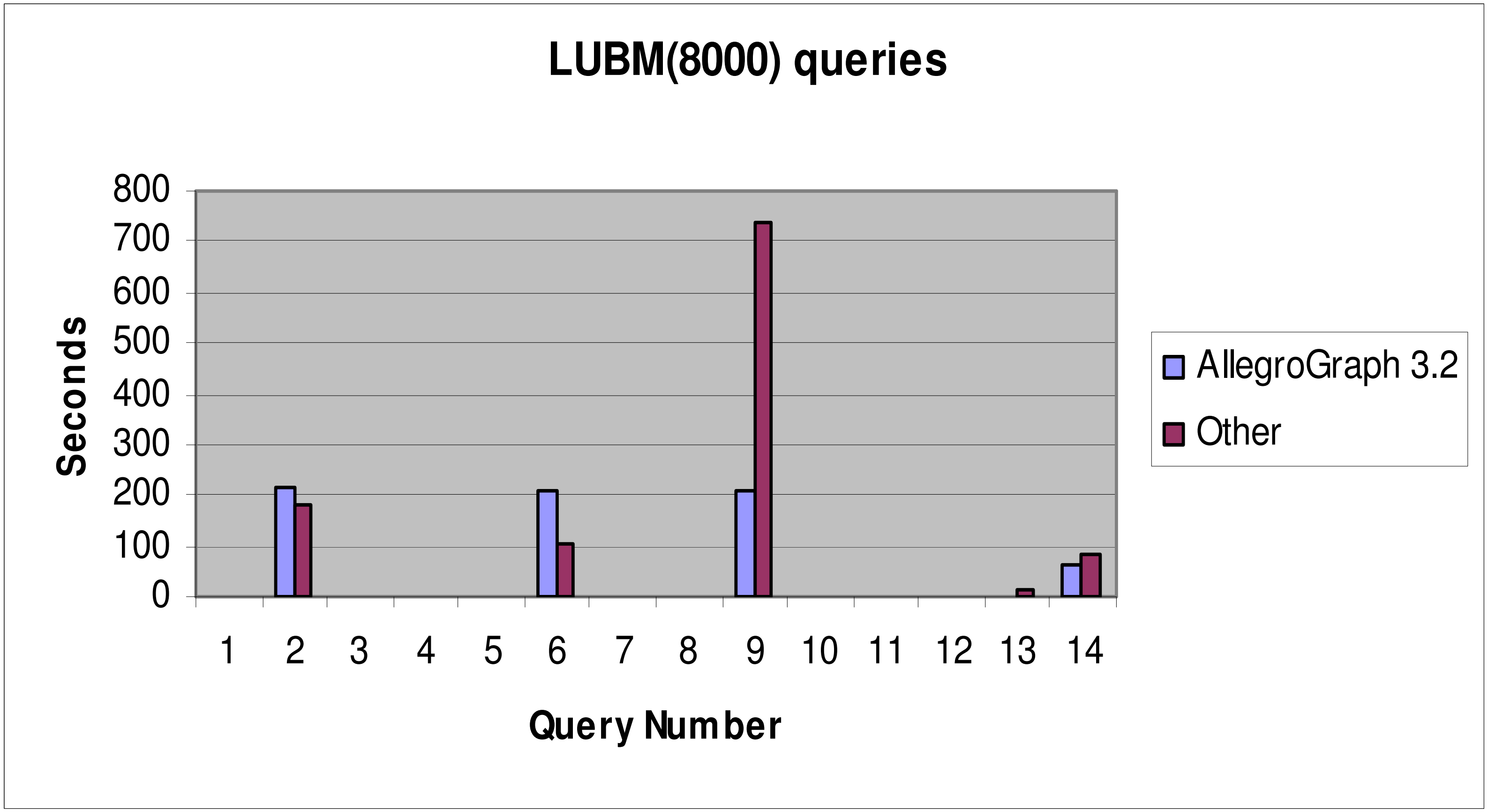

AllegroGraph LUBM(50) Comparison

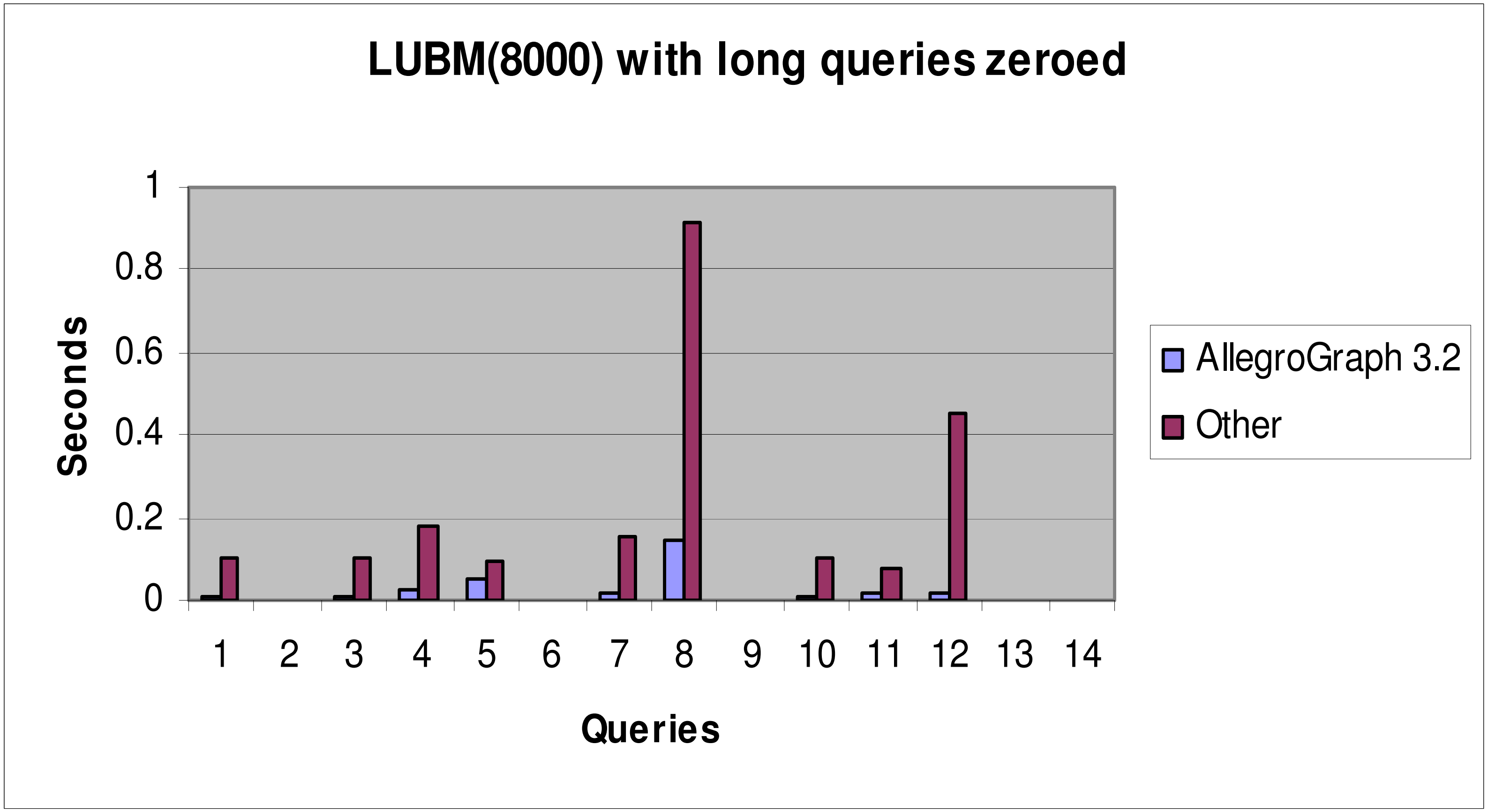

LUBM(50) large query zeroed
Seconds
6
5
4
3
2
1
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Queries
AllegroGraph 3.1
AllegroGraph 3.2


LUBM(50) with medium queries zeroed
Seconds
0.1
0.08
0.06
0.04
0.02
0
1
3
5
7
9
11
13
Queries
AllegroGraph 3.1
AllegroGraph 3.2

So does this work for huge triplestores?




LUBM(8000) queries
Seconds
800
700
600
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Query Number
AllegroGraph 3.2
Other



LUBM(8000) with long queries zeroed

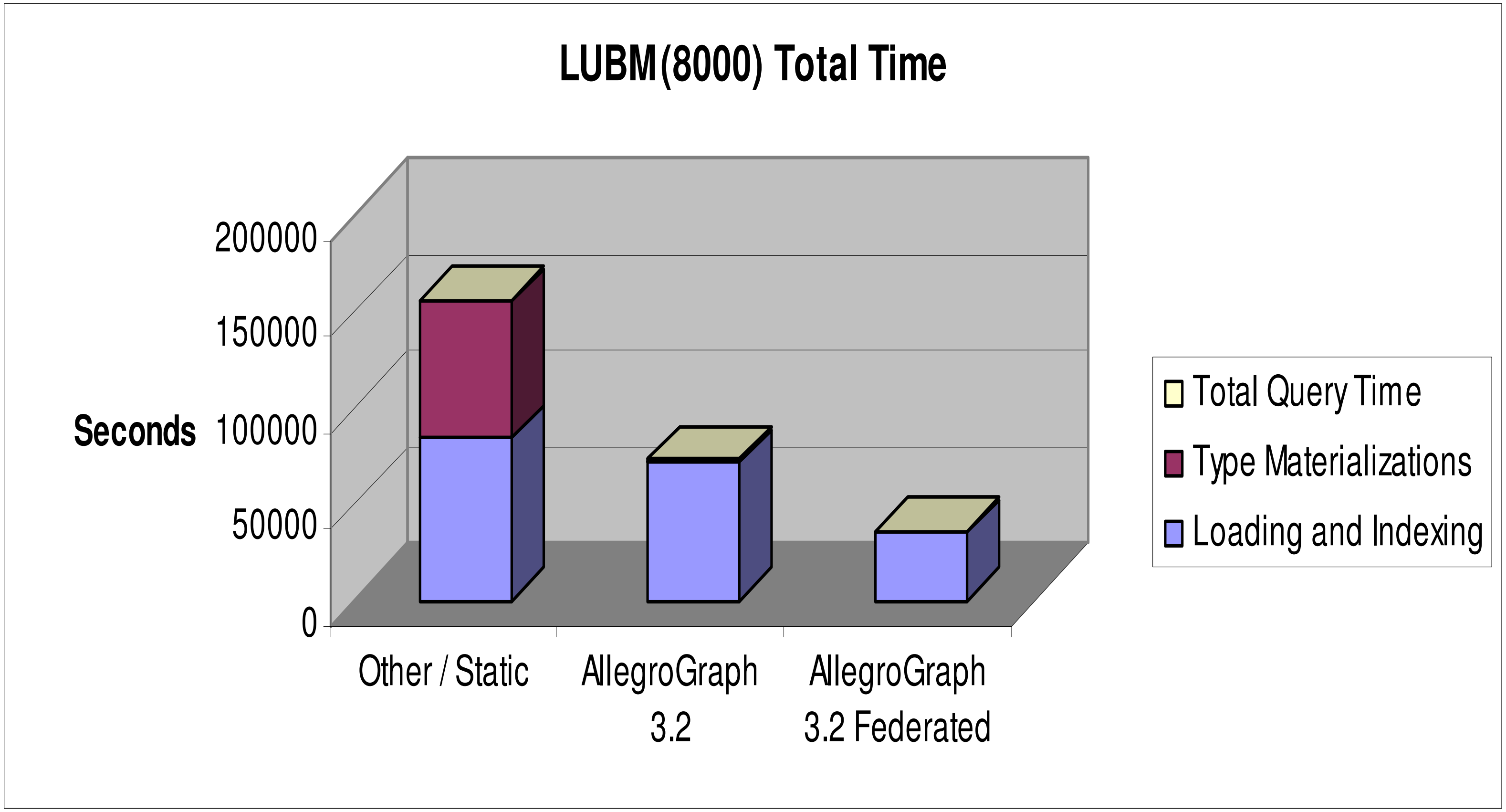
So what is the big deal? [1]

- AllegroGraph does not Materialize

- Typical triplestore:
  - Load & Index
  - Materialize: Do type inferences, some predicate normalizations
  - Index again

- With 3.2
  - Much more dynamic, add a few triples, delete or change an ontology
  - And back in the query business within a few minutes for a billion triples.

So what is the big deal? [2]



So what is the big deal? [1]

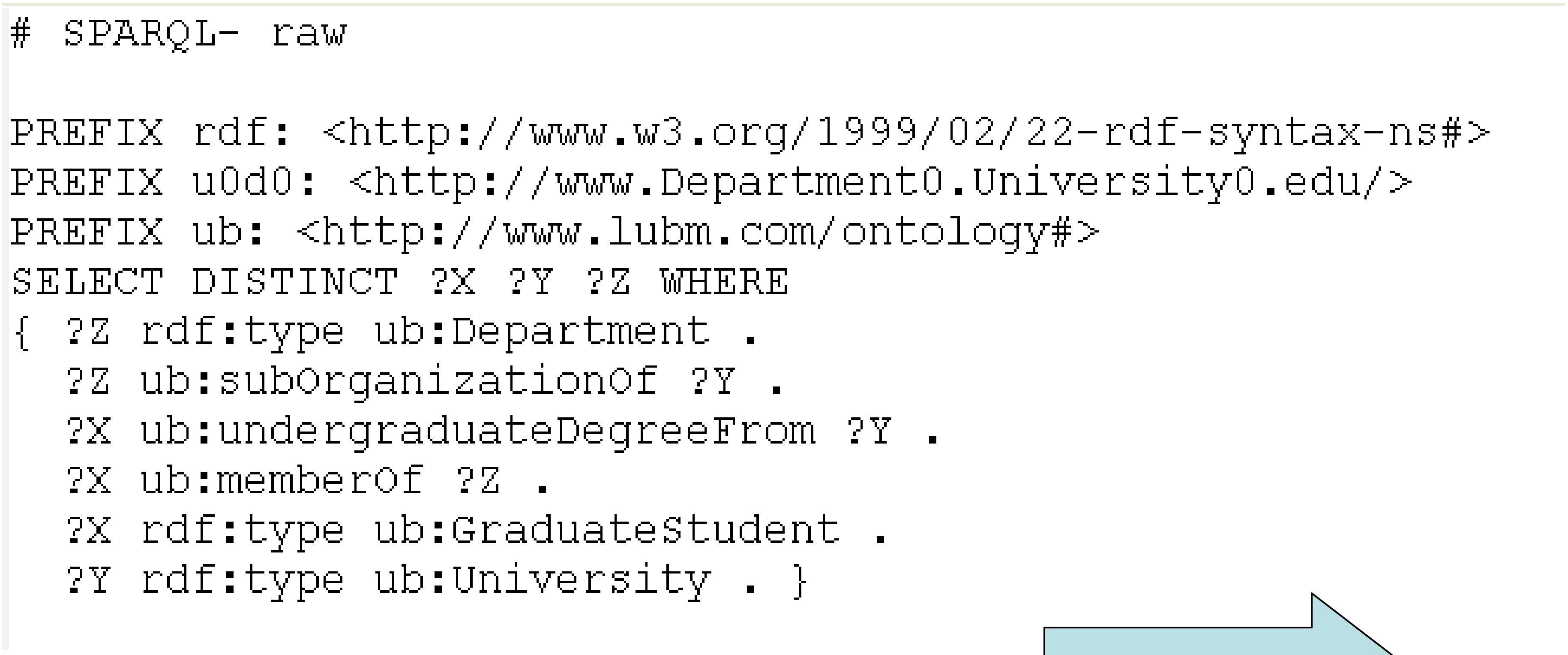
- AllegroGraph does not Materialize

- Typical triplestore:
  - Load & Index
  - Materialize: Do type inferences, some predicate normalizations
  - Index again

- With 3.2
  - Much more dynamic, add a few triples, delete or change an ontology
  - And back in the query business within a few minutes for a billion triples.

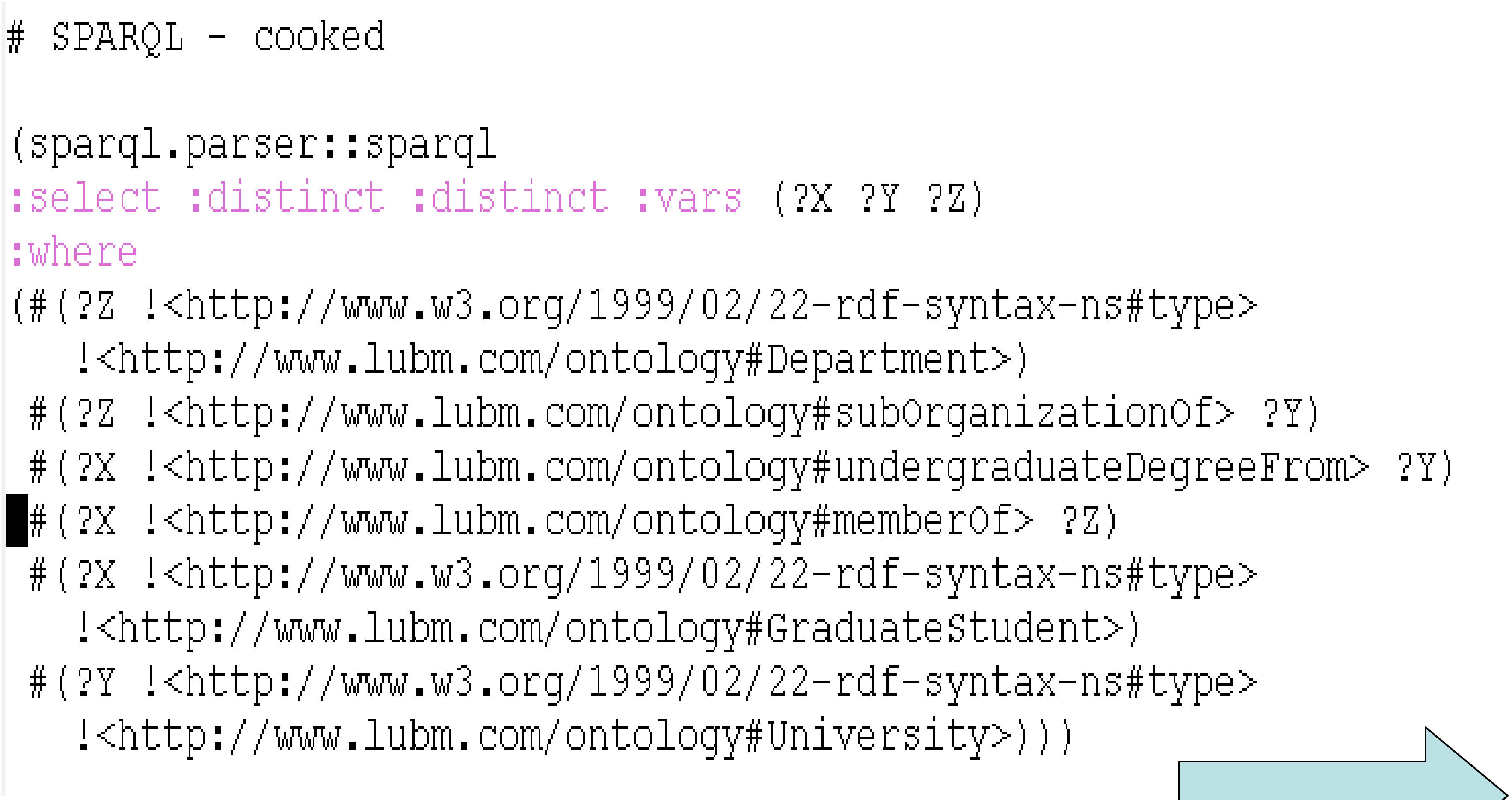
So how do we do this?

LUBM Query 2



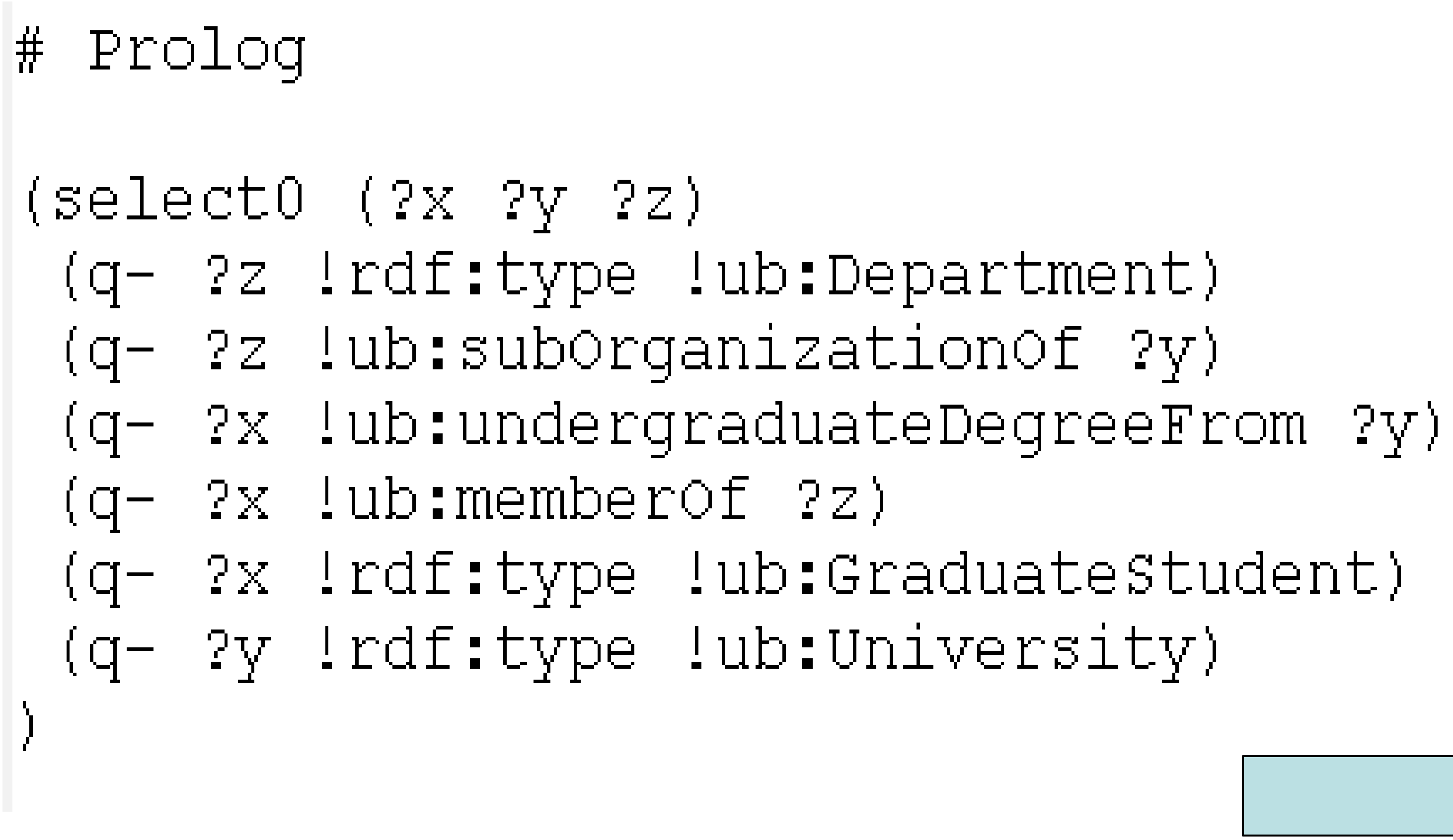
```
# SPARQL- raw

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX u0d0: <http://www.Department0.University0.edu/>
PREFIX ub: <http://www.lubm.com/ontology#>
SELECT DISTINCT ?X ?Y ?Z WHERE
{ ?Z rdf:type ub:Department .
  ?Z ub:subOrganizationOf ?Y .
  ?X ub:undergraduateDegreeFrom ?Y .
  ?X ub:memberOf ?Z .
  ?X rdf:type ub:GraduateStudent .
  ?Y rdf:type ub:University . }
```

Parsed



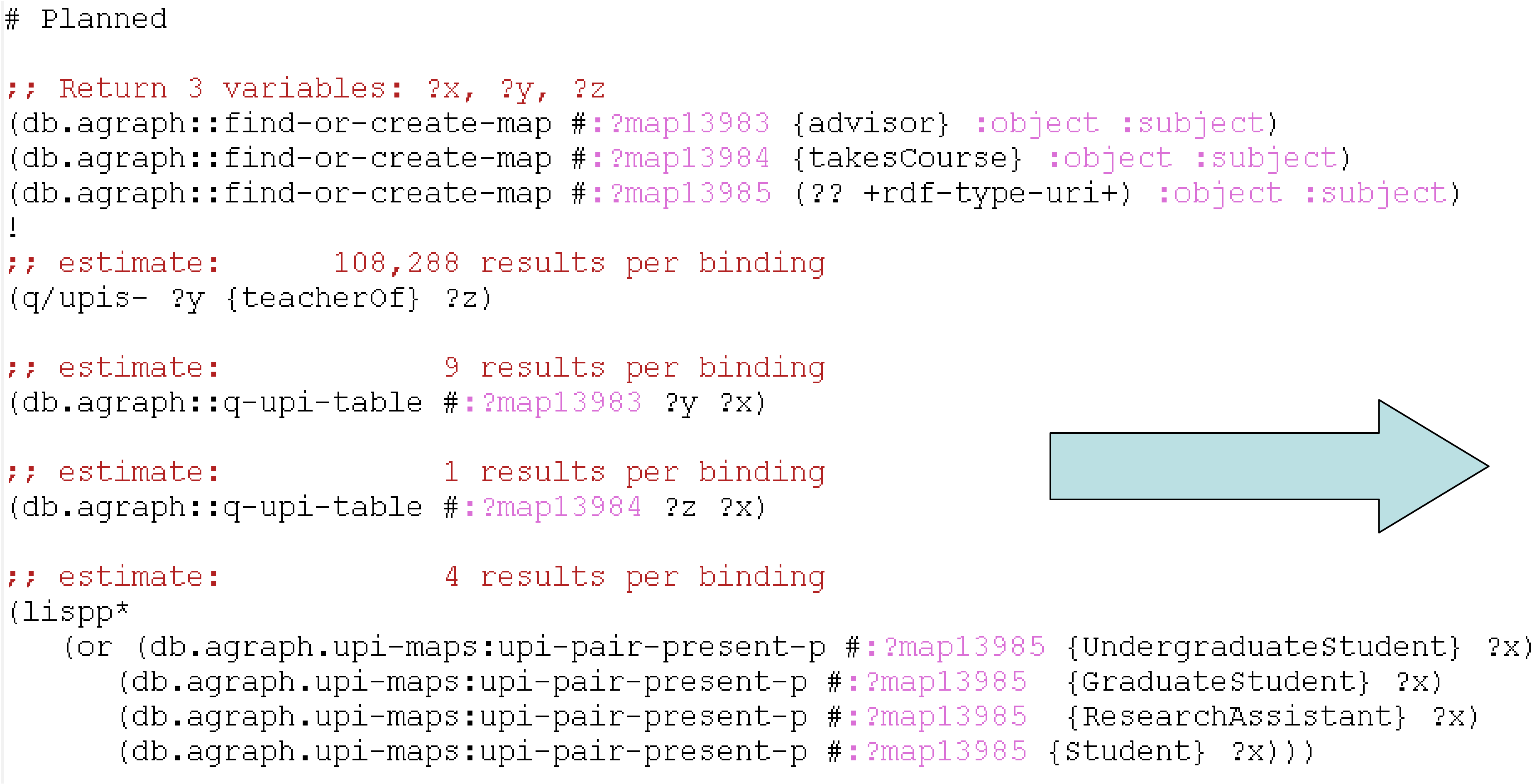
```
# SPARQL - cooked

(sparql.parser::sparql
:select :distinct :distinct :vars (?X ?Y ?Z)
:where
(#(?Z !<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   !<http://www.lubm.com/ontology#Department>)
 #(?Z !<http://www.lubm.com/ontology#subOrganizationOf> ?Y)
 #(?X !<http://www.lubm.com/ontology#undergraduateDegreeFrom> ?Y)
 #(?X !<http://www.lubm.com/ontology#memberOf> ?Z)
 #(?X !<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   !<http://www.lubm.com/ontology#GraduateStudent>)
 #(?Y !<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   !<http://www.lubm.com/ontology#University>)))
```

Rewritten in Prolog



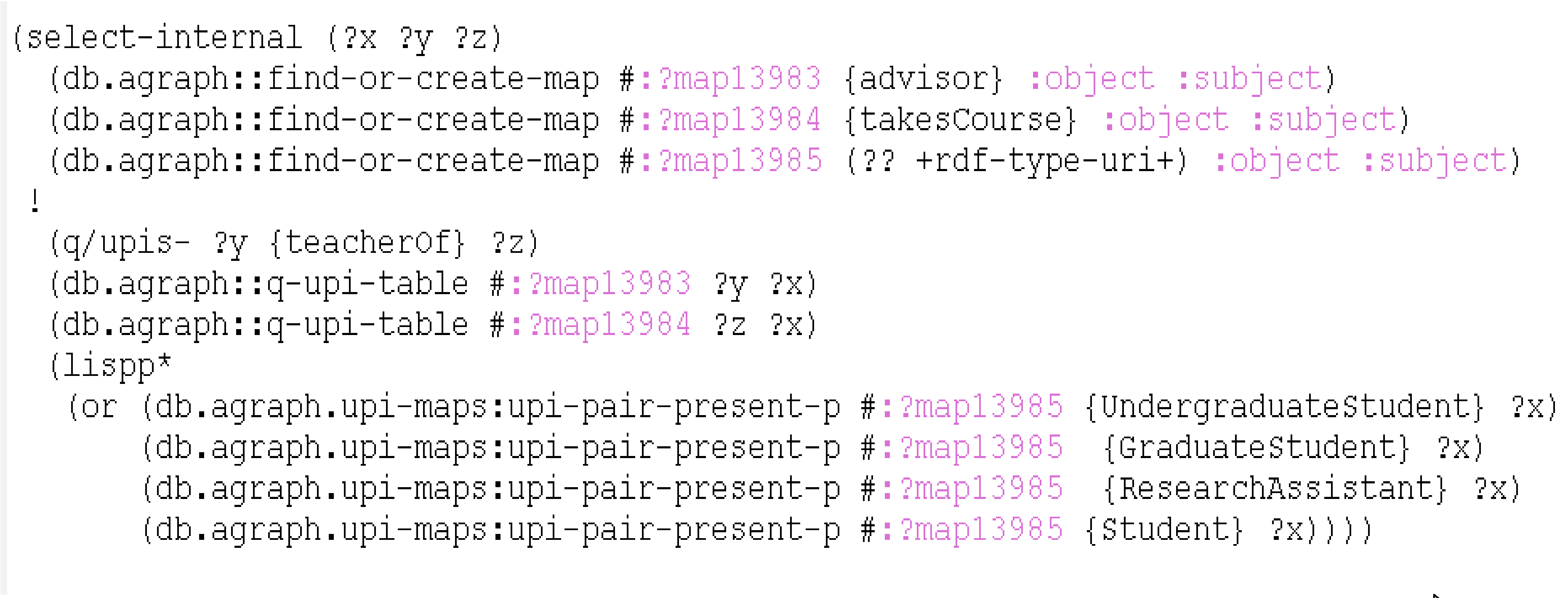
```
# Prolog

(select0 (?x ?y ?z)
 (q- ?z !rdf:type !ub:Department)
 (q- ?z !ub:subOrganizationOf ?y)
 (q- ?x !ub:undergraduateDegreeFrom ?y)
 (q- ?x !ub:memberOf ?z)
 (q- ?x !rdf:type !ub:GraduateStudent)
 (q- ?y !rdf:type !ub:University)
)
```

A statistics based plan with some reasoning simplifications



```
# Planned

;; Return 3 variables: ?x, ?y, ?z
(db.agraph::find-or-create-map #:?map13983 {advisor} :object :subject)
(db.agraph::find-or-create-map #:?map13984 {takesCourse} :object :subject)
(db.agraph::find-or-create-map #:?map13985 (?? +rdf-type-uri+) :object :subject)
!
;; estimate:      108,288 results per binding
(q/upis- ?y {teacherOf} ?z)

;; estimate:            9 results per binding
(db.agraph::q-upi-table #:?map13983 ?y ?x)

;; estimate:            1 results per binding
(db.agraph::q-upi-table #:?map13984 ?z ?x)

;; estimate:            4 results per binding
(lispp*
   (or (db.agraph.upi-maps:upi-pair-present-p #:?map13985 {UndergraduateStudent} ?x)
      (db.agraph.upi-maps:upi-pair-present-p #:?map13985  {GraduateStudent} ?x)
      (db.agraph.upi-maps:upi-pair-present-p #:?map13985  {ResearchAssistant} ?x)
      (db.agraph.upi-maps:upi-pair-present-p #:?map13985 {Student} ?x)))
```

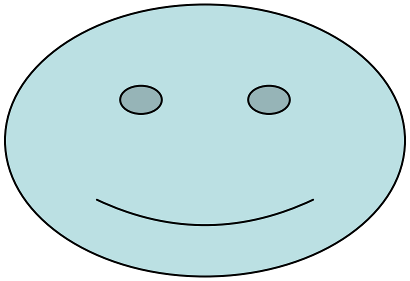
Internally executed as



```
(select-internal (?x ?y ?z)
  (db.agraph::find-or-create-map #:?map13983 {advisor} :object :subject)
  (db.agraph::find-or-create-map #:?map13984 {takesCourse} :object :subject)
  (db.agraph::find-or-create-map #:?map13985 (?? +rdf-type-uri+) :object :subject)
 !
  (q/upis- ?y {teacherOf} ?z)
  (db.agraph::q-upi-table #:?map13983 ?y ?x)
  (db.agraph::q-upi-table #:?map13984 ?z ?x)
  (lispp*
   (or (db.agraph.upi-maps:upi-pair-present-p #:?map13985 {UndergraduateStudent} ?x)
       (db.agraph.upi-maps:upi-pair-present-p #:?map13985  {GraduateStudent} ?x)
       (db.agraph.upi-maps:upi-pair-present-p #:?map13985  {ResearchAssistant} ?x)
       (db.agraph.upi-maps:upi-pair-present-p #:?map13985 {Student} ?x))))
```

Well, to be honest, really compiled down to machine instructions

```
;; code start: #x20f49904:
   0: 55             pushl ebp
   1: 8b ec         movl   ebp,esp
   3: 83 ec 30   subl      esp,$48
   6: 89 75 fc   movl      [ebp-4],esi
   9: 89 5d e4   movl      [ebp-28],ebx
  12: 39 a3 be 00 cmpl     [ebx+190],esp
      00 00
  18: 76 03          jbe      23
  20: ff 57 43   call      *[edi+67]        ; sys::trap-stack-ovfl
  23: 83 f9 01   cmpl      ecx,$1
  26: 74 03          jz       31
  28: ff 57 8b   call      *[edi-117]       ; sys::trap-wnaerr
  31: 8b 5d 00   movl      ebx,[ebp+0]
  34: 8b 5b ec   movl      ebx,[ebx-20]
  37: 8b 5b fa   movl      ebx,[ebx-6]
  40: 80 7f cb 00 cmpb     [edi-53],$0   ; sys::c_interrupt-pending
  44: 74 03          jz       49
  46: ff 57 87   call      *[edi-121]       ; sys::trap-signal-hit
  49: 80 7f cb 00 cmpb     [edi-53],$0   ; sys::c_interrupt-pending
  53: 74 03          jz       58
  55: ff 57 87   call      *[edi-121]       ; sys::trap-signal-hit
  58: 33 c0          xorl     eax,eax
  60: f8              clc
  61: 72 03          jb       66
  63: 33 c9          xorl     ecx,ecx
  65: 41              incl    ecx
  66: ff 73 fe   pushl     [ebx-2]
  69: 8f 45 dc   popl      [ebp-36]         ; excl::local-0
  72: 8b 5d dc   movl      ebx,[ebp-36]     ; excl::local-0
  75: 89 5d e8   movl      [ebp-24],ebx
  78: 8b 5d e4   movl      ebx,[ebp-28]
```



Concluding with some reality

- Expect 3.2 in a few days. Call if you want prelease now.
- The prolog query optimizer will work for you
- The Sparql will still run on our old reasoner, expect the faster Sparql on our next release

Thank you