Graph-based RDF Data Managment

Lei Zou

Peking University
Institute of Computer Science and Technology

RDF and Semantic Web

- RDF is a language for the conceptual modeling of information about web resources
- A building block of semantic web
  - Facilitates exchange of information
  - Search engines can retrieve more relevant information
  - Facilitates data integration (mashes)
- Machine understandable
  - Understand the information on the web and the interrelationships among them

What's Sematic Web: A Simple Example (RDFa)

The traditional Web (HTML) only considers the display of the content.

How is the page displayed, such as which font and the format of the pictures ?

```
<html>
    <font size="3" color="red"> Lei Zou </font>
    </br>
    Email:<a href= "mailto: zoulei@pku.edu.cn">
zoulei@pku.edu.cn  </a>
    <p>
       <font size="3" color="black">Publications: </font>
    </p>
   <div>
        Lei Zou, Jinhui Mo, Lei Chen, M. Tamer Ozsu,
Dongyan Zhao, gStore: Answering SPARQL Queries Via
Subgraph Matching, VLDB, 2011
    </div>
</html>
```

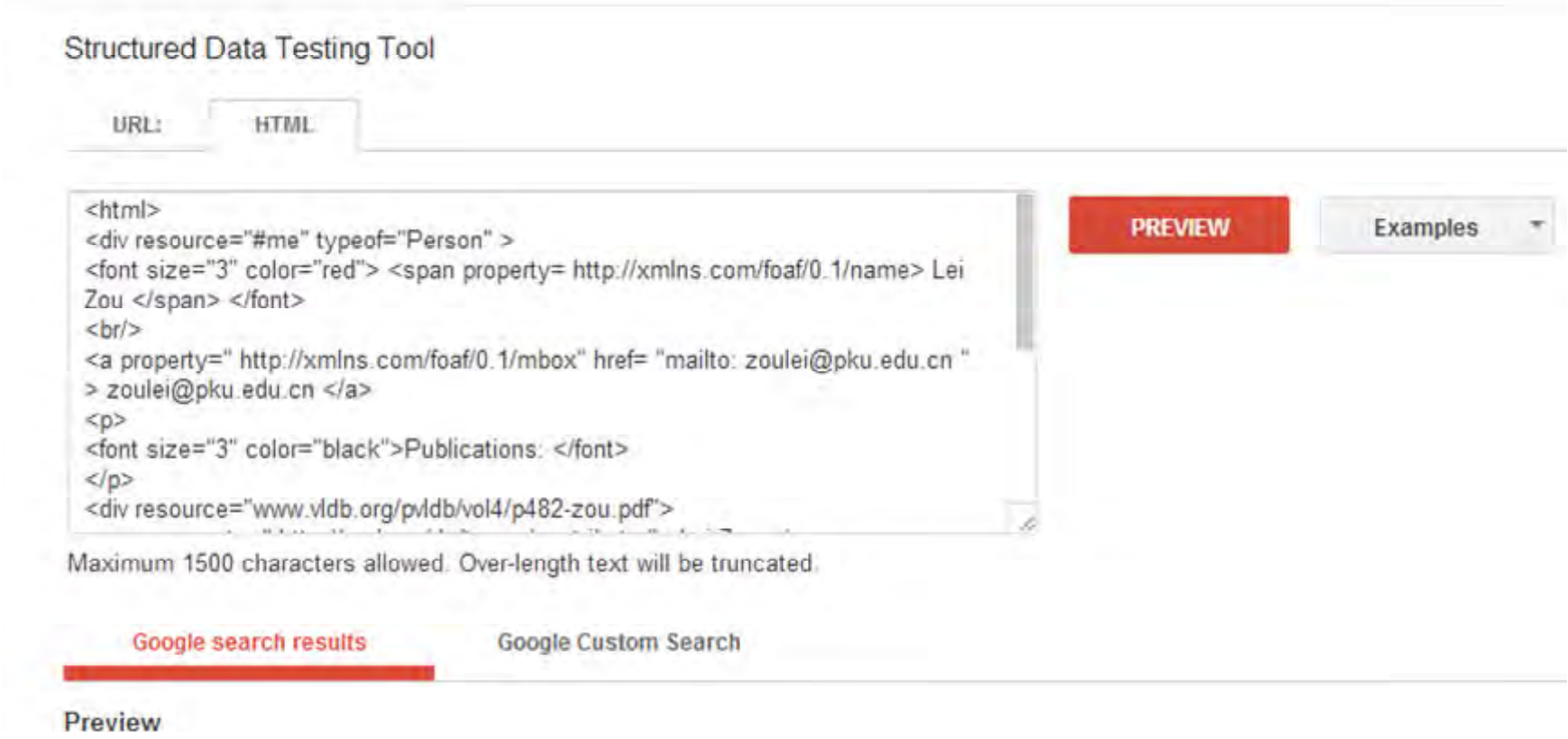
What's Sematic Web: A Simple Example (RDFa)

Sematic Web considers the sematics of the content.

What does the content in the page mean? e.g., What are the mean of "zoulei@pku.edu.cn" and "VLDB" ?

```
<html>
<div resource="#me" typeof="Person" >
<font size="3" color="red"> <span property= http://xmlns.com/foaf/0.1/name> Lei
Zou </span> </font>
<br/>
<a property=" http://xmlns.com/foaf/0.1/mbox" href= "mailto: zoulei@pku.edu.cn "
> zoulei@pku.edu.cn </a>
<p>
<font size="3" color="black">Publications: </font>
</p>
<div resource="www.vldb.org/pvldb/vol4/p482-zou.pdf">
    <span property=" http://purl.org/dc/terms/contributor"> Lei Zou </span>,
    <span property="http://purl.org/dc/terms/contributor"> Jinghui Mo </span>,
    <span property=" http://purl.org/dc/terms/contributor"> Lei Chen </span>,
    <span property=" http://purl.org/dc/terms/contributor"> M. Tamer Özsu</span>,
    <span property=" http://purl.org/dc/terms/contributor"> Dongyan Zhao</span>,
    <span property=" http://purl.org/dc/terms/title"> gStore: Answering SPARQL
Queries Via Subgraph Matching </span>,
    <span property=" http://purl.org/dc/terms/Publisher"> VLDB </span>
    <span property=" http://purl.org/dc/terms/Date">2011</span>
</div>
</html>
```

What's Sematic Web: Google Snippet



What's Sematic Web: Google Snippet

Extracted structured data

rdfa-node

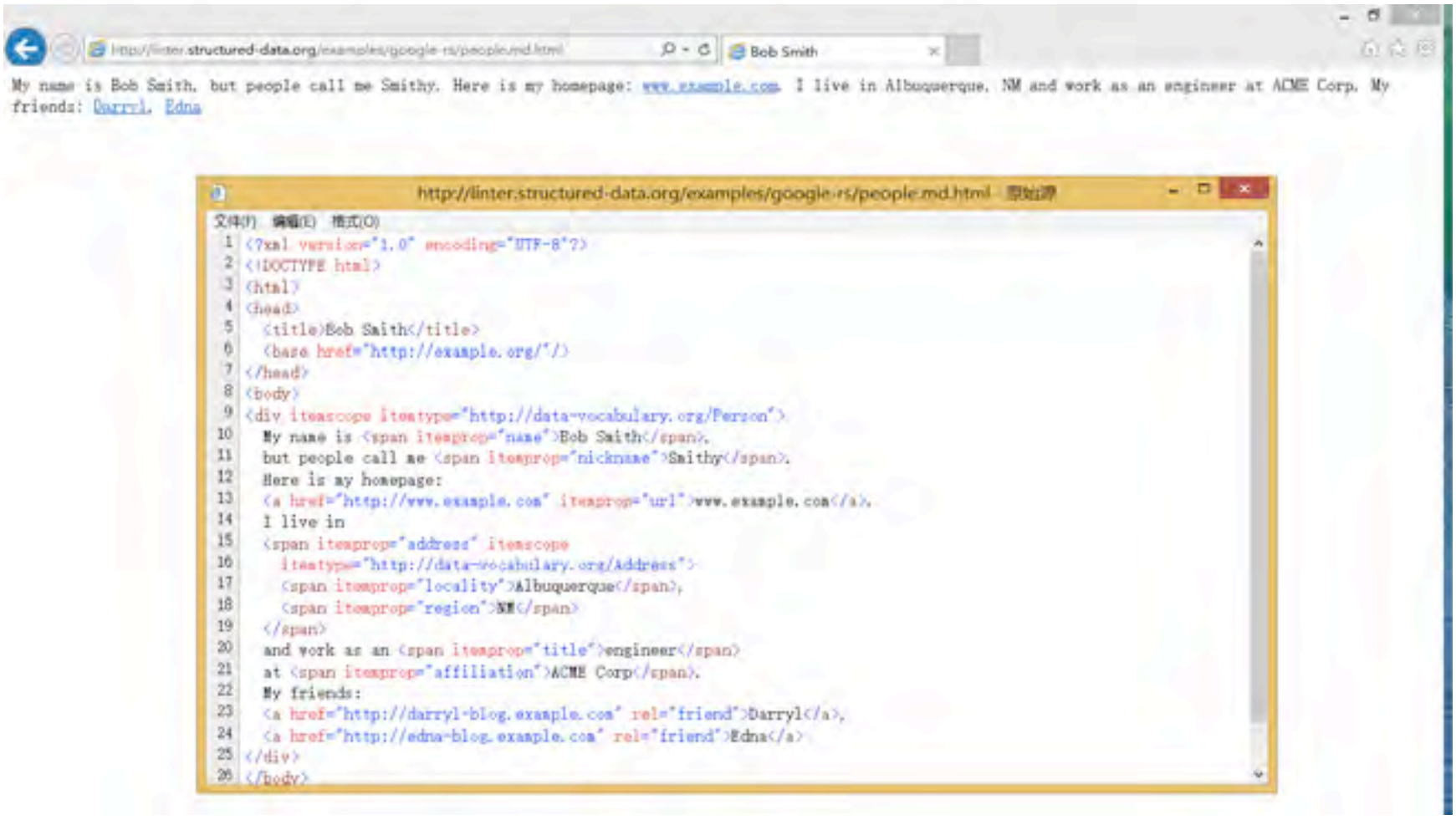
relationship:

name:	mbox
value:	zoulei@pku.edu.cn
href:	mailto:%20zoulei@pku.edu.cn

property:

name:	Lei Zou
contributor:	Lei Zou
contributor:	Jinghui Mo
contributor:	Lei Chen
contributor:	M. Tamer Özsu
contributor:	Dongyan Zhao
title:	gStore: Answering SPARQL Queries Via Subgraph Matching
Publisher:	VLDB
Date:	2011

What's Sematic Web: Google Snippet



What's Sematic Web: Google Snippet

Enhanced search result preview

*Disclaimer: this preview is only shown as a example of what a search engine **might** display. It is to the discretion of each search engine provider to decide whether your page will be displayed as an enhanced search result or not in their search results pages.*

Bob Smith
linter.structured-data.org/examples/google-rs/people.md.html
Albuquerque, NM - engineer, ACME Corp
an **actual** search result **may** display other content **relating** to your search terms here.

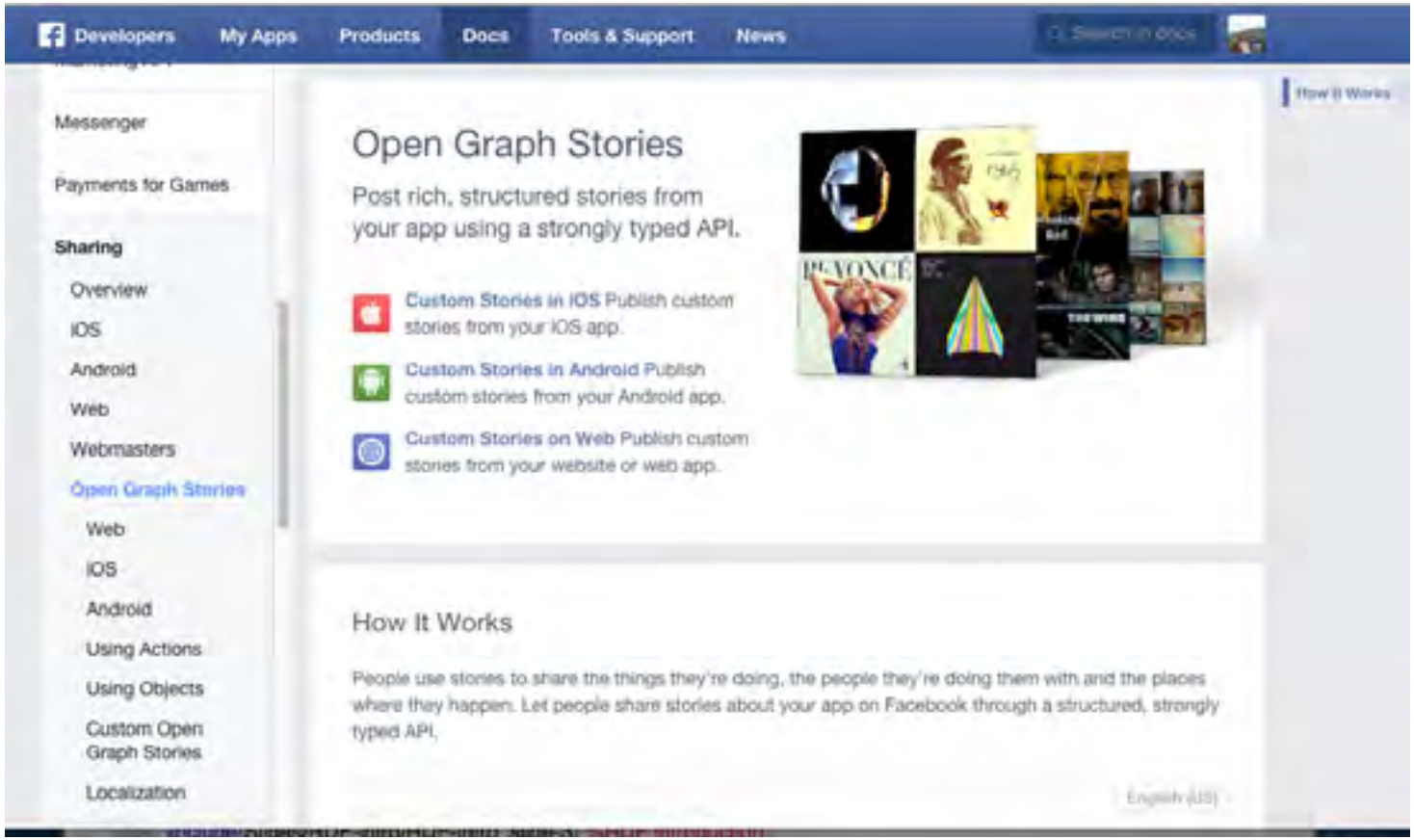
Raw structured data extracted from the page:

rdf:type	vmd:Person	
vmd:address	**rdf:type**	vmd:Address
	vmd:locality	Albuquerque
	vmd:region	NM
vmd:affiliation	ACME Corp	
vmd:name	Bob Smith	
vmd:nickname	Smithy	
vmd:title	engineer	
vmd:url	http://www.example.com	

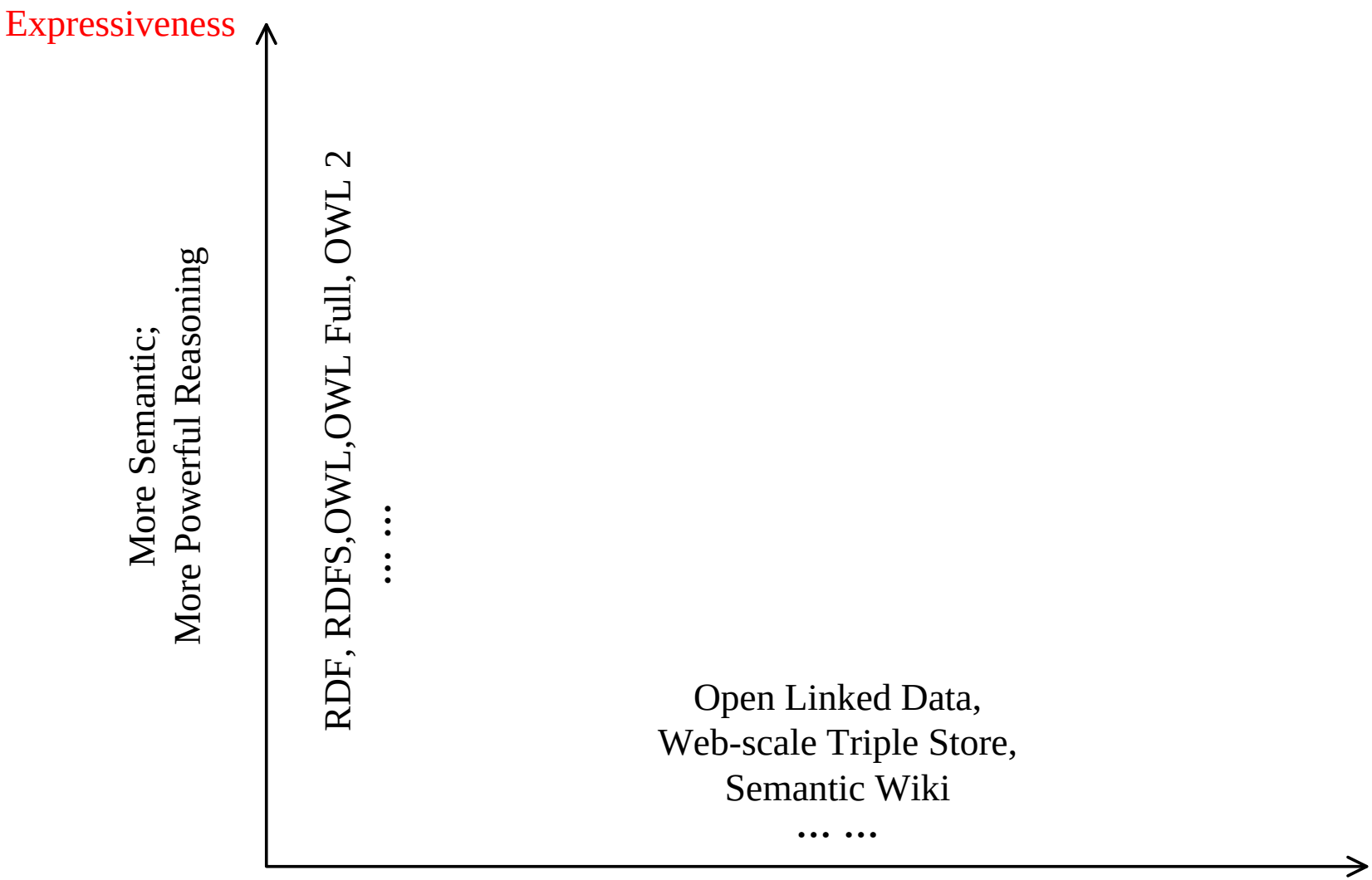
Parser statistics

Statements	12
Templates	http://data-vocabulary.org/Person http://data-vocabulary.org/Address

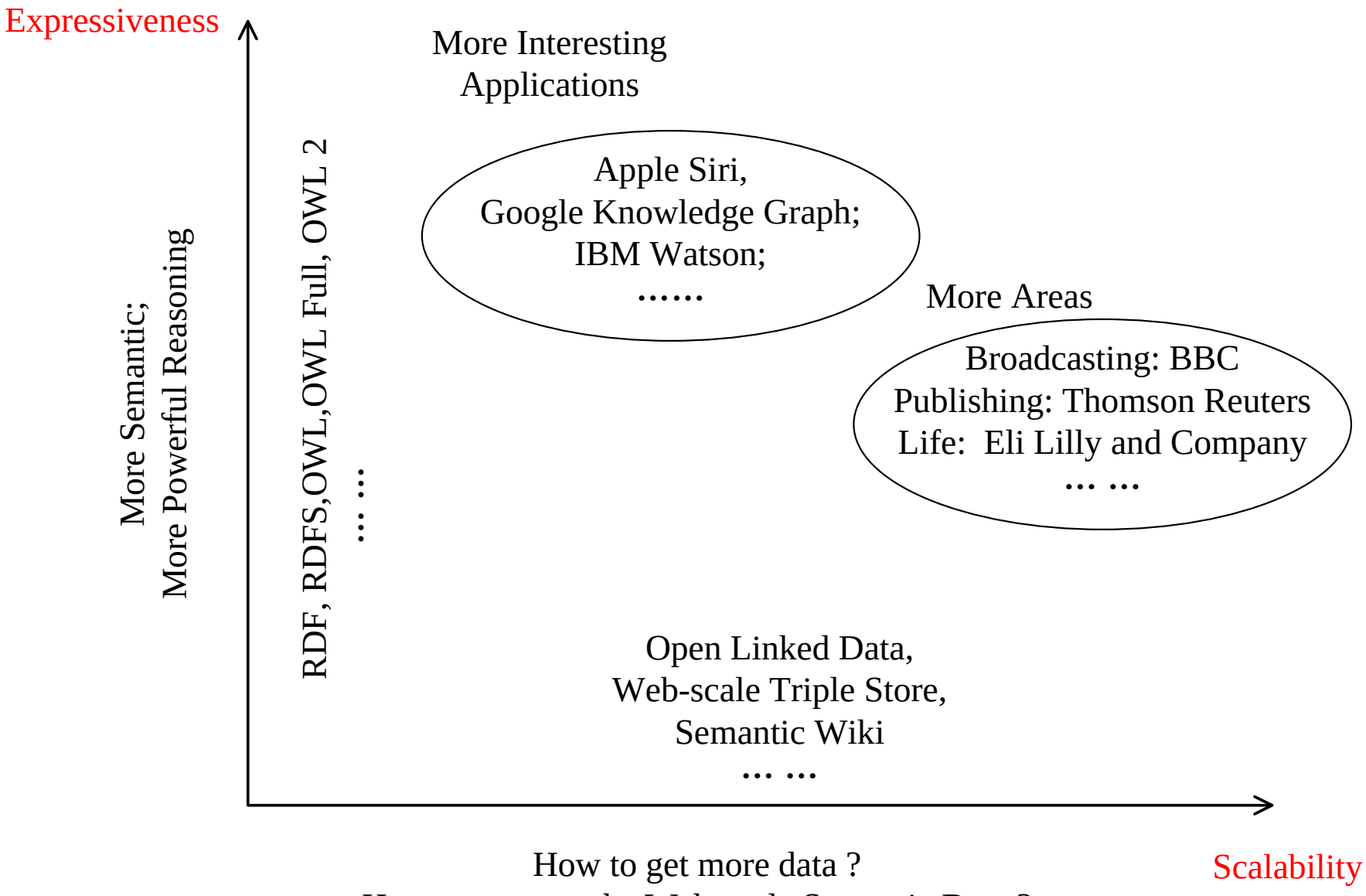
What's Sematic Web: Facebook Social Graph



What's Sematic Web: From Two Perspectives



What's Sematic Web: From Two Perspectives

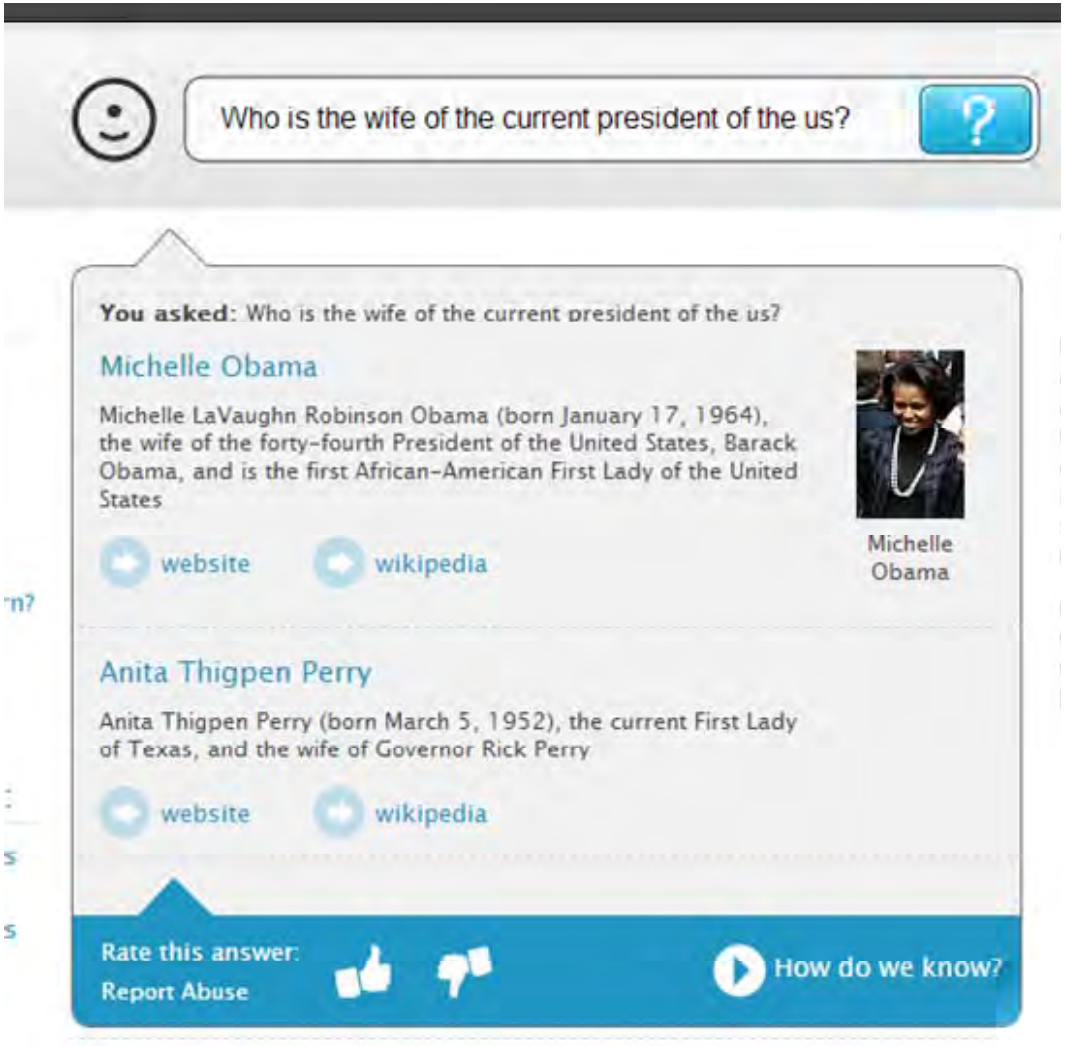


Some Interesting Products

IBM Watson

Some Interesting Products

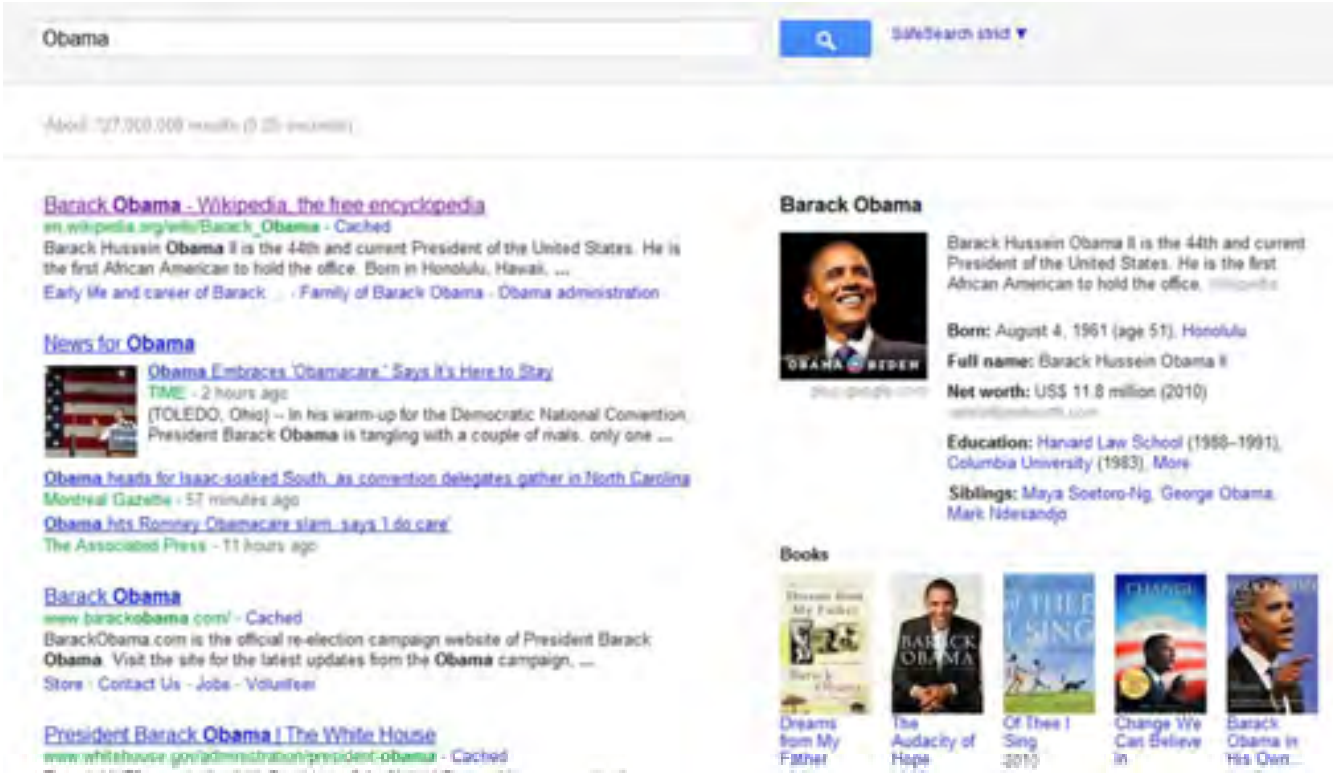
EVI— acquired by Amazon on October 2012.



William Tunstall-Pedoe: True Knowledge: Open-Domain Question Answering Using Structured Knowledge and Inference. AI Magazine 31(3): 80-92 (2010)

Some Interesting Products

Google Knowledge Graph



RDF Uses

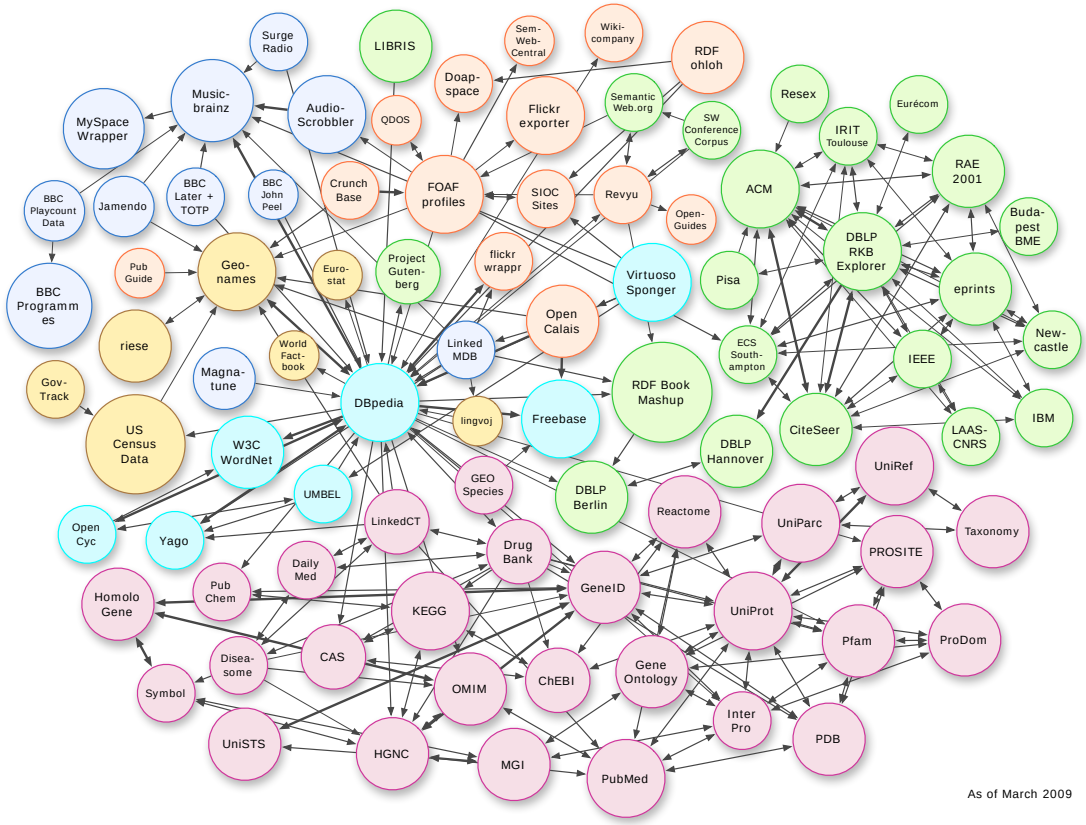
- Yago and DBPedia extract facts from Wikipedia & represent as RDF $\rightarrow$ structural queries
- Communities build RDF data
  - E.g., biologists: Bio2RDF and Uniprot RDF
- Web data integration
  - Linked Data Cloud
- ...

RDF Data Volumes . . .

- . . . are growing – and fast
  - Linked data cloud currently consists of 325 datasets with $>$25B triples
  - Size almost doubling every year

RDF Data Volumes . . .

- . . . are growing – and fast
  - Linked data cloud currently consists of 325 datasets with >25B triples
  - Size almost doubling every year

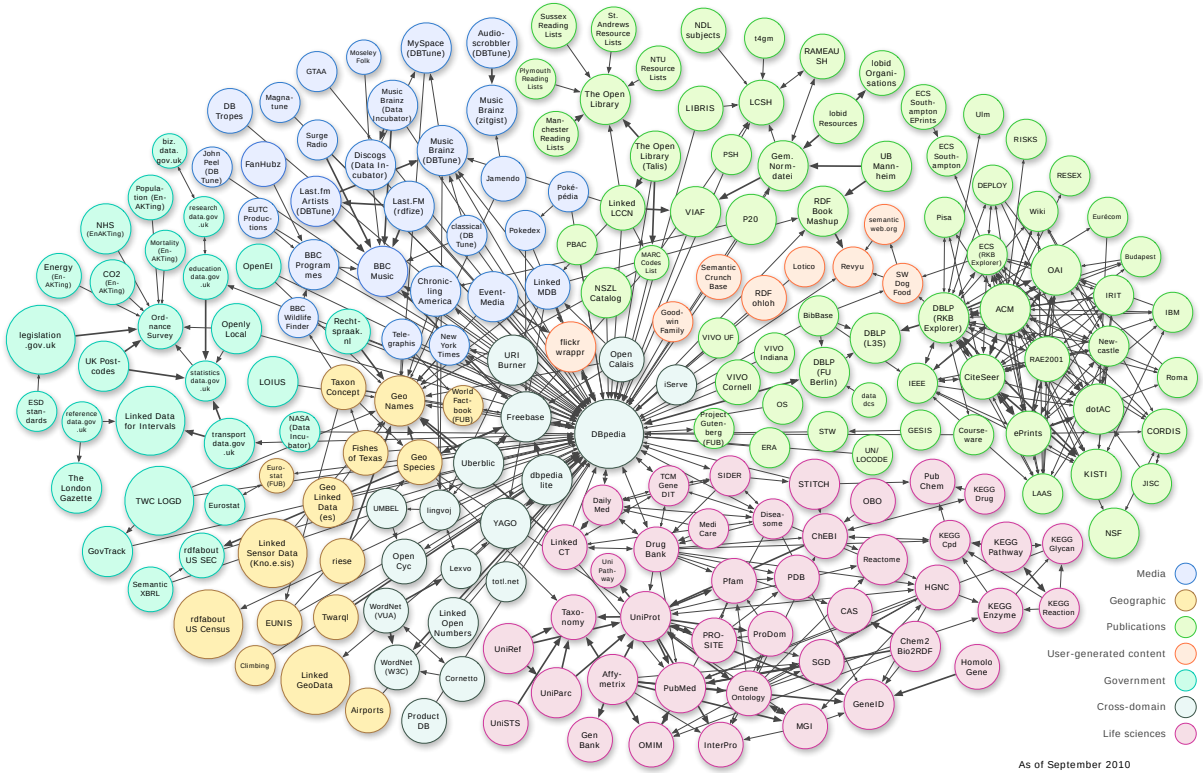


March '09:
89 datasets

Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch.
http://lod-cloud.net/

RDF Data Volumes . . .

- . . . are growing – and fast
  - Linked data cloud currently consists of 325 datasets with >25B triples
  - Size almost doubling every year

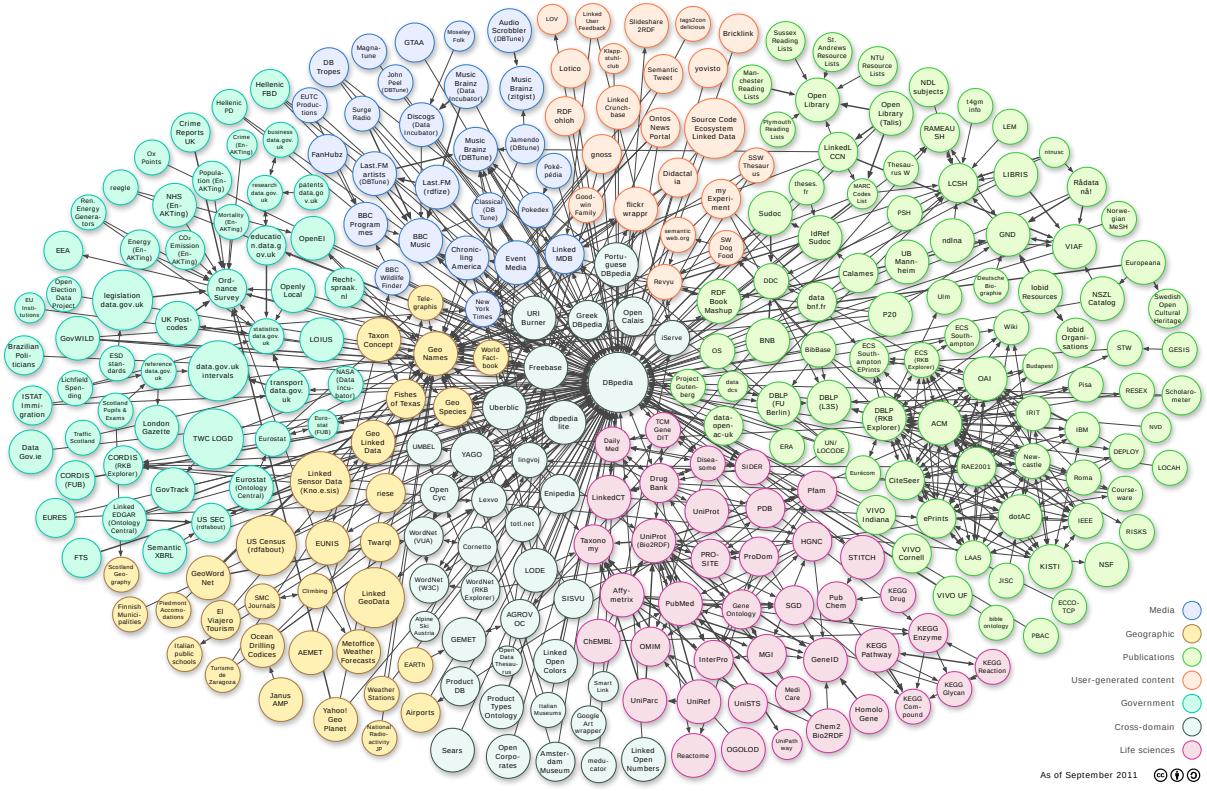


September '10:
203 datasets

Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch.
http://lod-cloud.net/

RDF Data Volumes . . .

- . . . are growing – and fast
  - Linked data cloud currently consists of 325 datasets with >25B triples
  - Size almost doubling every year

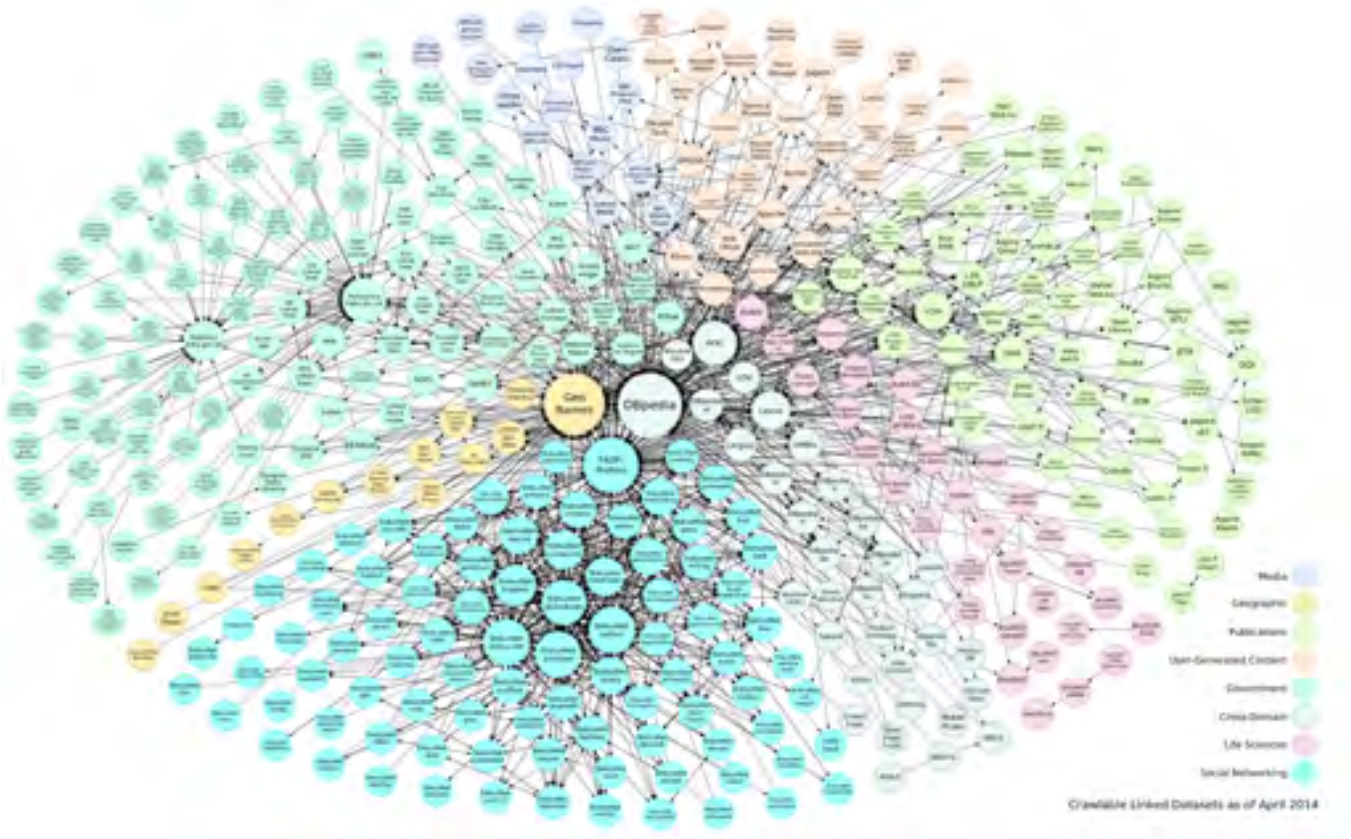


September '11:
295 datasets

Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch.
http://lod-cloud.net/

RDF Data Volumes . . .

- . . . are growing – and fast
  - Linked data cloud currently consists of 325 datasets with >25B triples
  - Size almost doubling every year



April '14:
1091 datasets, ???
triples

Max Schmachtenberg, Christian Bizer, and Heiko Paulheim: Adoption of Linked Data Best Practices in Different Topical Domains. In *Proc. ISWC*, 2014.

Outline

RDF Introduction

gStore: a graph-based SPARQL query engine

Answering SPARQL queries using graph pattern matching [Zou et al., PVLDB 2011, VLDB J 2014]

gAnswer: Natural Language Question Answering over RDF

A Data Driven Approach [Zou et al., SIGMOD 2014; Zheng et al., SIGMOD 2015]

Outline

RDF Introduction

gStore: a graph-based SPARQL query engine

Answering SPARQL queries using graph pattern matching [Zou et al., PVLDB 2011, VLDB J 2014]

gAnswer: Natural Language Question Answering over RDF

A Data Driven Approach [Zou et al., SIGMOD 2014; Zheng et al., SIGMOD 2015]

RDF Introduction

- Everything is an uniquely named resource

http://en.wikipedia.org/wiki/Abraham_Lincoln

RDF Introduction

- Everything is an uniquely named resource
- Namespaces can be used to scope the names

xmlns:y=http://en.wikipedia.org/wiki

y:Abraham_Lincoln

RDF Introduction

- Everything is an uniquely named resource
- Namespaces can be used to scope the names
- Properties of resources can be defined

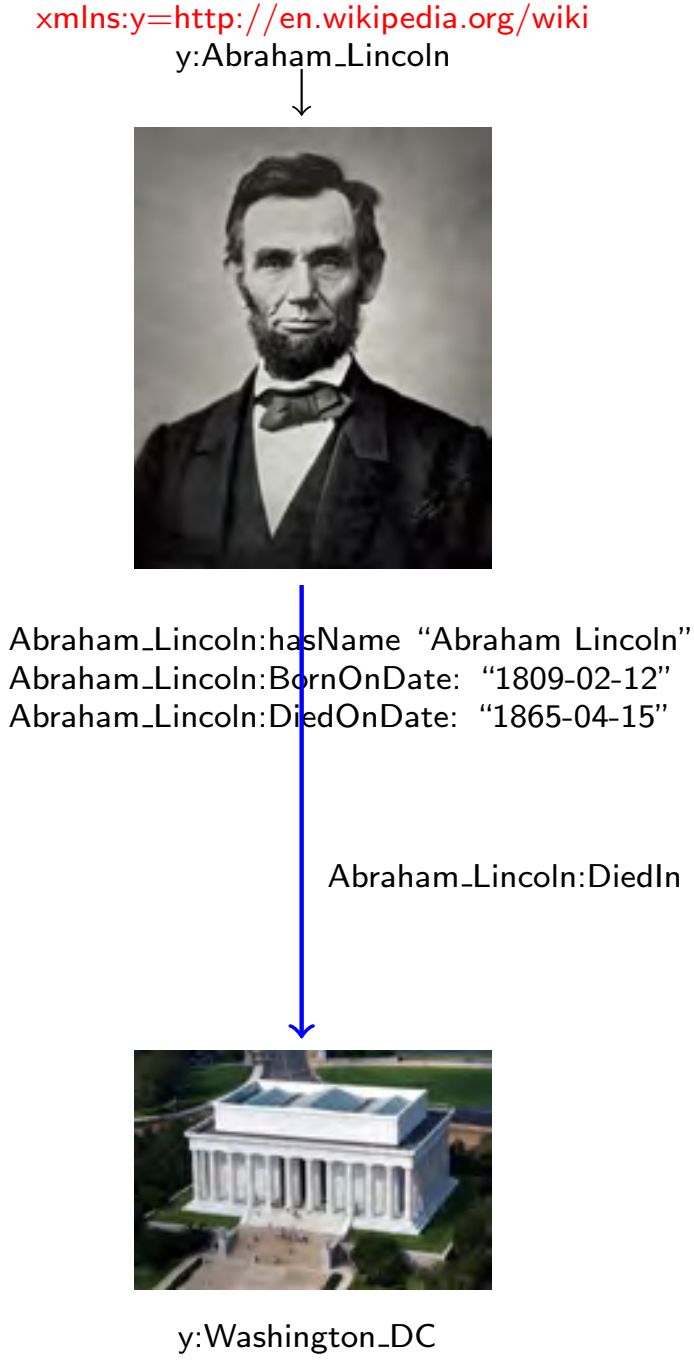
xmlns:y=http://en.wikipedia.org/wiki

y:Abraham_Lincoln

Abraham_Lincoln:hasName "Abraham Lincoln"
Abraham_Lincoln:BornOnDate: "1809-02-12"
Abraham_Lincoln:DiedOnDate: "1865-04-15"

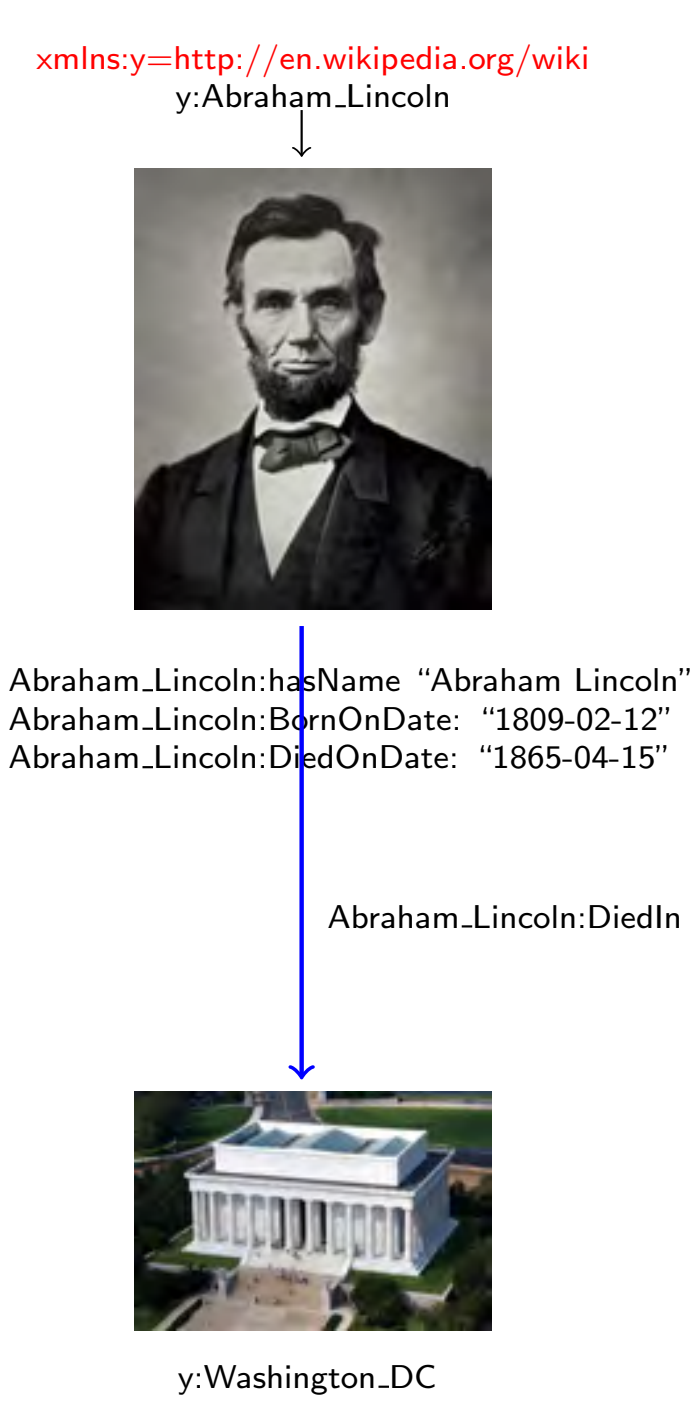
RDF Introduction

- Everything is an uniquely named resource
- Namespaces can be used to scope the names
- Properties of resources can be defined
- Relationships with other resources can be defined



RDF Introduction

- Everything is an uniquely named resource
- Namespaces can be used to scope the names
- Properties of resources can be defined
- Relationships with other resources can be defined
- Resources can be contributed by different people/groups and can be located anywhere in the web
  - Integrated web “database”



RDF Data Model

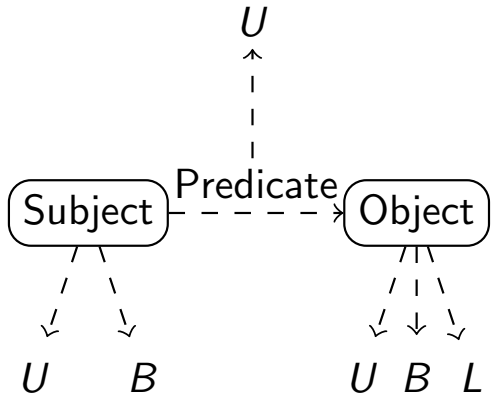
- Triple: Subject, Predicate (Property), Object (s, p, o)

  Subject: the entity that is described (URI or blank node)

  Predicate: a feature of the entity (URI)

  Object: value of the feature (URI, blank node or literal)

- $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$
- Set of RDF triples is called an RDF graph



U: set of URIs
B: set of blank nodes
L: set of literals

Subject	**Predicate**	**Object**
Abraham_Lincoln	hasName	"Abraham Lincoln"
Abraham_Lincoln	BornOnDate	"1809-02-12"
Abraham_Lincoln	DiedOnDate	"1865-04-15"

RDF Example Instance

Prefix: y=http://en.wikipedia.org/wiki

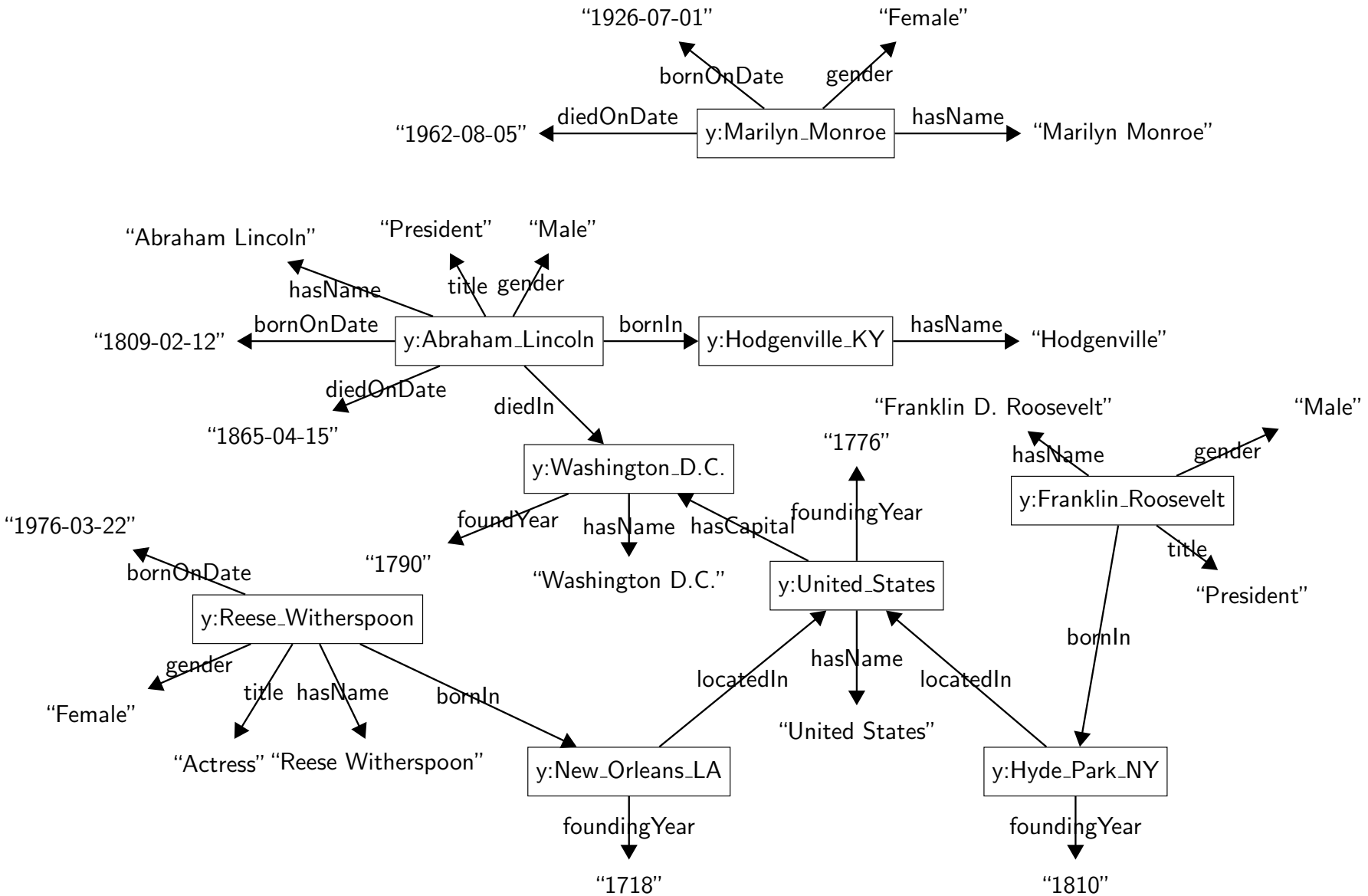
Subject	Predicate	Object
y: Abraham_Lincoln	hasName	"Abraham Lincoln"
y: Abraham_Lincoln	BornOnDate	"1809-02-12"'
y: Abraham_Lincoln	DiedOnDate	"1865-04-15"
y:Abraham_Lincoln	bornIn	y:Hodgenville_KY
y: Abraham_Lincoln	DiedIn	y: Washington_DC
y:Abraham_Lincoln	title	"President"
y:Abraham_Lincoln	gender	"Male"
y: Washington_DC	hasName	"Washington D.C."
y:Washington_DC	foundingYear	"1790"
y:Hodgenville_KY	hasName	"Hodgenville"
y:United_States	hasName	"United States"
y:United_States	hasCapital	y:Washington_DC
y:United_States	foundingYear	"1776"
y:Reese_Witherspoon	bornOnDate	"1976-03-22"
y:Reese_Witherspoon	bornIn	y:New_Orleans_LA
y:Reese_Witherspoon	hasName	"Reese Witherspoon"
y:Reese_Witherspoon	gender	"Female"
y:Reese_Witherspoon	title	"Actress"
y:New_Orleans_LA	foundingYear	"1718"
y:New_Orleans_LA	locatedIn	y:United_States
y:Franklin_Roosevelt	hasName	"Franklin D. Roosevelt"
y:Franklin_Roosevelt	bornIn	y:Hyde_Park_NY
y:Franklin_Roosevelt	title	"President"
y:Franklin_Roosevelt	gender	"Male"
y:Hyde_Park_NY	foundingYear	"1810"
y:Hyde_Park_NY	locatedIn	y:United_States
y:Marilyn_Monroe	gender	"Female"
y:Marilyn_Monroe	hasName	"Marilyn Monroe"
y:Marilyn_Monroe	bornOnDate	"1926-07-01"
y:Marilyn_Monroe	diedOnDate	"1962-08-05"

URI

Literal

URI

RDF Graph


"1926-07-01"
"Female"
bornOnDate
gender
diedOnDate
y:Marilyn_Monroe
hasName
"1962-08-05"
"Marilyn Monroe"
"Abraham Lincoln"
"President"
"Male"
hasName
title
gender
"1809-02-12"
bornOnDate
y:Abraham_Lincoln
bornIn
y:Hodgenville_KY
hasName
"Hodgenville"
diedOnDate
diedIn
"1865-04-15"
y:Washington_D.C.
"Franklin D. Roosevelt"
"Male"
"1776"
hasName
gender
y:Franklin_Roosevelt
foundYear
hasName
hasCapital
foundingYear
"1790"
"Washington D.C."
"1976-03-22"
bornOnDate
y:United_States
title
"President"
y:Reese_Witherspoon
bornIn
gender
hasName
locatedIn
locatedIn
"Female"
title
hasName
bornIn
"United States"
"Actress"
"Reese Witherspoon"
y:New_Orleans_LA
y:Hyde_Park_NY
foundingYear
foundingYear
"1718"
"1810"

RDF Query Model

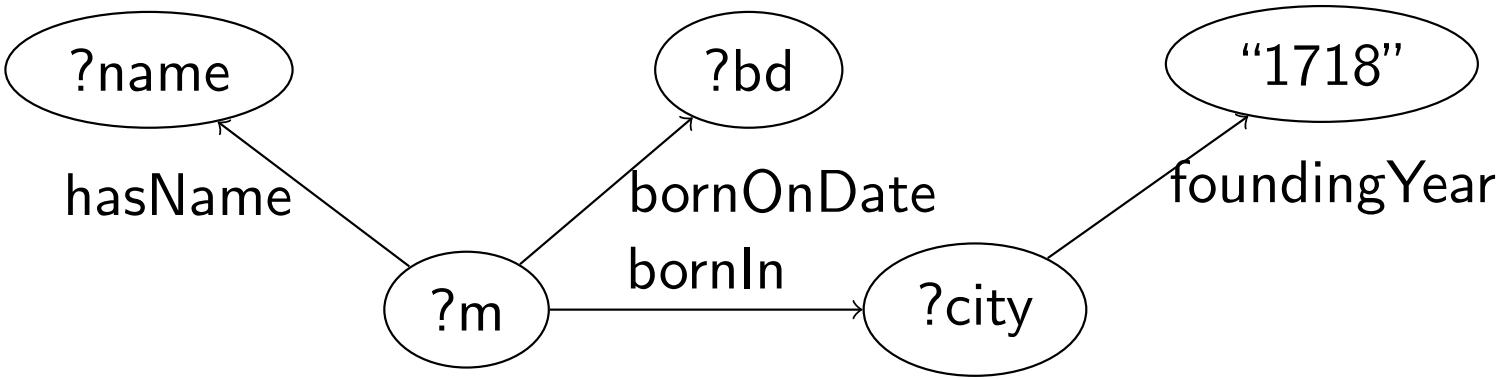
- Query Model - **S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage
- Given U (set of URIs), L (set of literals), and V (set of variables), a SPARQL expression is defined recursively:
  - an atomic triple pattern, which is an element of
    $$(U \cup V) \times (U \cup V) \times (U \cup V \cup L)$$
    - ?x hasName "Abraham Lincoln"
  - P FILTER R, where P is a graph pattern expression and R is a built-in SPARQL condition (i.e., analogous to a SQL predicate)
    - ?x price ?p FILTER(?p $<$ 30)
  - $P1$ AND/OPT/UNION $P2$, where $P1$ and $P2$ are graph pattern expressions
- Example:

```
SELECT ?name
WHERE {
   ?m <bornIn> ?city. ?m <hasName> ?name.
   ?m<bornOnDate> ?bd. ?city <foundingYear> ''1718''.
   FILTER(regex(str(?bd),''1976''))
}
```

SPARQL Queries

```
SELECT ?name
WHERE {
   ?m <bornIn> ?city. ?m <hasName> ?name.
   ?m<bornOnDate> ?bd. ?city <foundingYear> ''1718''.
   FILTER(regex(str(?bd),''1976''))
}
```

FILTER(*regex*(*str*(?bd),"1976"))



Naïve Triple Store Design

```
SELECT ?name
WHERE {
   ?m <bornIn> ?city. ?m <hasName> ?name.
   ?m<bornOnDate> ?bd. ?city <foundingYear> ''1718''.
   FILTER(regex(str(?bd),''1976''))
}
```

Subject	Property	Object
y:Abraham_Lincoln	hasName	"Abraham Lincoln"
y:Abraham_Lincoln	bornOnDate	"1809-02-12"
y:Abraham_Lincoln	diedOnDate	"1865-04-15"
y:Abraham_Lincoln	bornIn	y:Hodgenville_KY
y:Abraham_Lincoln	diedIn	y:Washington_DC
y:Abraham_Lincoln	title	"President"
y:Abraham_Lincoln	gender	"Male"
y:Washington_DC	hasName	"Washington D.C."
y:Washington_DC	foundingYear	"1790"
y:Hodgenville_KY	hasName	"Hodgenville"
y:United_States	hasName	"United States"
y:United_States	hasCapital	y:Washington_DC
y:United_States	foundingYear	"1776"
y:Reese_Witherspoon	bornOnDate	"1976-03-22"
y:Reese_Witherspoon	bornIn	y:New_Orleans_LA
y:Reese_Witherspoon	hasName	"Reese Witherspoon"
y:Reese_Witherspoon	gender	"Female"
y:Reese_Witherspoon	title	"Actress"
y:New_Orleans_LA	foundingYear	"1718"
y:New_Orleans_LA	locatedIn	y:United_States
y:Franklin_Roosevelt	hasName	"Franklin D. Roosevelt"
y:Franklin_Roosevelt	bornIn	y:Hyde_Park_NY
y:Franklin_Roosevelt	title	"President"
y:Franklin_Roosevelt	gender	"Male"
y:Hyde_Park_NY	foundingYear	"1810"
y:Hyde_Park_NY	locatedIn	y:United_States
y:Marilyn_Monroe	gender	"Female"
y:Marilyn_Monroe	hasName	"Marilyn Monroe"
y:Marilyn_Monroe	bornOnDate	"1926-07-01"
y:Marilyn_Monroe	diedOnDate	"1962-08-05"

Naïve Triple Store Design

```
SELECT ?name
WHERE {
   ?m <bornIn> ?city. ?m <hasName> ?name.
   ?m<bornOnDate> ?bd. ?city <foundingYear> ''1718''.
   FILTER(regex(str(?bd),''1976''))
}
```

Subject	Property	Object
y:Abraham_Lincoln	hasName	"Abraham Lincoln"
y:Abraham_Lincoln	bornOnDate	"1809-02-12"
y:Abraham_Lincoln	diedOnDate	"1865-04-15"
y:Abraham_Lincoln	bornIn	y:Hodgenville_KY
y:Abraham_Lincoln	diedIn	y:Washington_DC
y:Abraham_Lincoln	title	"President"
y:Abraham_Lincoln	gender	"Male"
y:Washington_DC	hasName	"Washington D.C."
y:Washington_DC	foundingYear	"1790"
y:Hodgenville_KY	hasName	"Hodgenville"
y:United_States	hasName	"United States"
y:United_States	hasCapital	y:Washington_DC
y:United_States	foundingYear	"1776"
y:Reese_Witherspoon	bornOnDate	"1976-03-22"
y:Reese_Witherspoon	bornIn	y:New_Orleans_LA
y:Reese_Witherspoon	hasName	"Reese Witherspoon"
y:Reese_Witherspoon	gender	"Female"
y:Reese_Witherspoon	title	"Actress"
y:New_Orleans_LA	foundingYear	"1718"
y:New_Orleans_LA	locatedIn	y:United_States
y:Franklin_Roosevelt	hasName	"Franklin D. Roosevelt"
y:Franklin_Roosevelt	bornIn	y:Hyde_Park_NY
y:Franklin_Roosevelt	title	"President"
y:Franklin_Roosevelt	gender	"Male"
y:Hyde_Park_NY	foundingYear	"1810"
y:Hyde_Park_NY	locatedIn	y:United_States
y:Marilyn_Monroe	gender	"Female"
y:Marilyn_Monroe	hasName	"Marilyn Monroe"
y:Marilyn_Monroe	bornOnDate	"1926-07-01"
y:Marilyn_Monroe	diedOnDate	"1962-08-05"

```
SELECT T2.object
FROM   T as T1, T as T2, T as T3,
       T as T4
WHERE T1.property="bornIn"
AND T2.property="hasName"
AND T3.property="bornOnDate"
AND T1.subject=T2.subject
AND T2.subject=T3.subject
AND T4.propety="foundingYear"
AND T1.object=T4.subject
AND T4.object="1718"
AND T3.object LIKE '%1976%'
```

Naïve Triple Store Design

```
SELECT ?name
WHERE {
   ?m <bornIn> ?city. ?m <hasName> ?name.
   ?m<bornOnDate> ?bd. ?city <foundingYear>
   FILTER(regex(str(?bd),``1976''))
}
```

Subject	Property	Object
y:Abraham_Lincoln	hasName	"Abraham Lincoln"
y:Abraham_Lincoln	bornOnDate	"1809-02-12"
y:Abraham_Lincoln	diedOnDate	"1865-04-15"
y:Abraham_Lincoln	bornIn	y:Hodgenville_KY
y:Abraham_Lincoln	diedIn	y:Washington_DC
y:Abraham_Lincoln	title	"President"
y:Abraham_Lincoln	gender	"Male"
y:Washington_DC	hasName	"Washington D.C."
y:Washington_DC	foundingYear	"1790"
y:Hodgenville_KY	hasName	"Hodgenville"
y:United_States	hasName	"United States"
y:United_States	hasCapital	y:Washington_DC
y:United_States	foundingYear	"1776"
y:Reese_Witherspoon	bornOnDate	"1976-03-22"
y:Reese_Witherspoon	bornIn	y:New_Orleans_LA
y:Reese_Witherspoon	hasName	"Reese Witherspoon"
y:Reese_Witherspoon	gender	"Female"
y:Reese_Witherspoon	title	"Actress"
y:New_Orleans_LA	foundingYear	"1718"
y:New_Orleans_LA	locatedIn	y:United_States
y:Franklin_Roosevelt	hasName	"Franklin D. Roosevelt"
y:Franklin_Roosevelt	bornIn	y:Hyde_Park_NY
y:Franklin_Roosevelt	title	"President"
y:Franklin_Roosevelt	gender	"Male"
y:Hyde_Park_NY	foundingYear	"1810"
y:Hyde_Park_NY	locatedIn	y:United_States
y:Marilyn_Monroe	gender	"Female"
y:Marilyn_Monroe	hasName	"Marilyn Monroe"
y:Marilyn_Monroe	bornOnDate	"1926-07-01"
y:Marilyn_Monroe	diedOnDate	"1962-08-05"


```
SELECT T2.object
FROM   T as T1, T as T2, T as T3,
       T as T4
WHERE T1.property="bornIn"
AND T2.property="hasName"
AND T3.property="bornOnDate"
AND T1.subject=T2.subject
AND T2.subject=T3.subject
AND T4.propety="foundingYear"
AND T1.object=T4.subject
AND T4.object="1718"
AND T3.object LIKE '%1976%'
```

Existing Solutions

1. Property table
   - Each class of objects go to a different table $\Rightarrow$ similar to normalized relations
   - Eliminates some of the joins
2. Vertically partitioned tables
   - For each property, build a two-column table, containing both subject and object, ordered by subjects
   - Can use merge join (faster)
   - Good for subject-subject joins but does not help with subject-object joins
3. Exhaustive indexing
   - Create indexes for each permutation of the three columns
   - Query components become range queries over individual relations with merge-join to combine
   - Excessive space usage

Property Tables

- Grouping by entities; Jena [Wilkinson et al., SWDB 03] ,FlexTable [Wang et al., DASFAA 10] , DB2-RDF [Bornea et al., SIGMOD 13]
- *Clustered property table*: group together the properties that tend to occur in the same (or similar) subjects
- *Property-class table*: cluster the subjects with the same type of property into one property table

Subject	Property	Object
y:Abraham_Lincoln	hasName	"Abraham Lincoln"
y:Abraham_Lincoln	bornOnDate	"1809-02-12"
y:Abraham_Lincoln	diedOnDate	"1865-04-15"
y:Washington_DC	hasName	"Washington D.C."
y:Washington_DC	foundingYear	"1790"
...	...	...

Subject	hasName	bornOnDate	diedOnDate
y:Abraham_Lincoln	"Abraham Lincoln"	1809-02-12	1865-04-15
y:Reese_Witherspoon	"Reese Witherspoon"	1976-03-22	

Subject	hasName	foundingYear
y:Washington_DC	"Washington D.C."	1790
y:Hyde_Park_NY	"Hyde Park"	1810'

Property Tables

- Grouping by entities; Jena [Wilkinson et al., SWDB 03] ,FlexTable [Wang et al., DASFAA 10] , DB2-RDF [Bornea et al., SIGMOD 13]
- *Clustered property table*: group together the properties that

Advantages

- Fewer joins
- If the data is structured, we have a relational system – similar to normalized relations

y:Abraham_Lincoln	bornOnDate	"1809-02-12
y:Abraham_Lincoln	diedOnDate	"1865-04-15"
y:Washington_DC	hasName	"Washington D.C."
y:Washington_DC	foundingYear	"1790"
...	...	...

Subject	hasName	bornOnDate	diedOnDate
y:Abraham_Lincoln	"Abraham Lincoln"	1809-02-12	1865-04-15
y:Reese_Witherspoon	"Reese Witherspoon"	1976-03-22	

Subject	hasName	foundingYear
y:Washington_DC	"Washington D.C."	1790
y:Hyde_Park_NY	"Hyde Park"	1810'

Property Tables

- Grouping by entities; Jena [Wilkinson et al., SWDB 03] ,FlexTable [Wang et al., DASFAA 10] , DB2-RDF [Bornea et al., SIGMOD 13]
- *Clustered property table*: group together the properties that

Advantages

- Fewer joins
- If the data is structured, we have a relational system – similar to normalized relations

Disadvantages

- Potentially a lot of NULLs
- Clustering is not trivial
- Multi-valued properties are complicated

Subject	hasName	bornOnDate	diedOnDate
y:Abraham_Lincoln	"Abraham Lincoln"	1809-02-12	1865-04-15
y:Reese_Witherspoon	"Reese Witherspoon"	1976-03-22	

Subject	hasName	foundingYear
y:Washington_DC	"Washington D.C."	1790
y:Hyde_Park_NY	"Hyde Park"	1810'

Binary Tables

- Grouping by properties: For each property, build a two-column table, containing both subject and object, ordered by subjects [Abadi et al., VLDB 07]
- Also called vertical partitioned tables
- n two column tables (n is the number of unique properties in the data)

Subject	Property	Object
y:Abraham_Lincoln	hasName	"Abraham Lincoln"
y:Abraham_Lincoln	bornOnDate	"1809-02-12"
y:Abraham_Lincoln	diedOnDate	"1865-04-15"
y:Washington_DC	hasName	"Washington D.C."
y:Washington_DC	foundingYear	"1790"
...	...	...

hasName

Subject	Object
y:Abraham_Lincoln	"Abraham Lincoln"
y:Washington_DC	"Washington D.C."

bornOnDate

Subject	Object
y:Abraham_Lincoln	1809-02-12
y:Reese_Witherspoon	1976-03-22

foundingYear

Subject	Object
y:Washington_DC	1790
y:Hyde_Park_NY	1810

Binary Tables

- Grouping by properties: For each property, build a two-column table, containing both subject and object, ordered by subjects

Advantages

- Supports multi-valued properties
- No NULLs
- No clustering
- Read only needed attributes (i.e. less I/O)
- Good performance for subject-subject joins

y:Washington_DC	hasName	"Washington D.C."
y:Washington_DC	foundingYear	"1790"
...	...	...

hasName

Subject	Object
y:Abraham_Lincoln	"Abraham Lincoln"
y:Washington_DC	"Washington D.C."

bornOnDate

Subject	Object
y:Abraham_Lincoln	1809-02-12
y:Reese_Witherspoon	1976-03-22

foundingYear

Subject	Object
y:Washington_DC	1790
y:Hyde_Park_NY	1810

Binary Tables

- Grouping by properties: For each property, build a two-column table, containing both subject and object, ordered by subjects

Advantages

- Supports multi-valued properties
- No NULLs
- No clustering
- Read only needed attributes (i.e. less I/O)
- Good performance for subject-subject joins

Disadvantages

- Not useful for subject-object joins
- Expensive inserts

Subject	Object
y:Abraham_Lincoln	"Abraham Lincoln"
y:Washington_DC	"Washington D.C."

Subject	Object
y:Abraham_Lincoln	1809-02-12
y:Reese_Witherspoon	1976-03-22

Subject	Object
y:Washington_DC	1790
y:Hyde_Park_NY	1810

Exhaustive Indexing

- RDF-3X [Neumann and Weikum, PVLDB 08] , Hexastore [Weiss et al., PVLDB 08]
- Strings are mapped to ids using a mapping table

Original triple table

Subject	Property	Object
y:Abraham_Lincoln	hasName	"Abraham Lincoln"
y:Abraham_Lincoln	bornOnDate	"1809-02-12"
y:Abraham_Lincoln	diedOnDate	"1865-04-15"
y:Washington_DC	hasName	"Washington D.C."
y:Washington_DC	foundingYear	"1790"

Encoded triple table

Subject	Property	Object
0	1	2
0	3	4
0	5	6
7	1	8
7	9	10

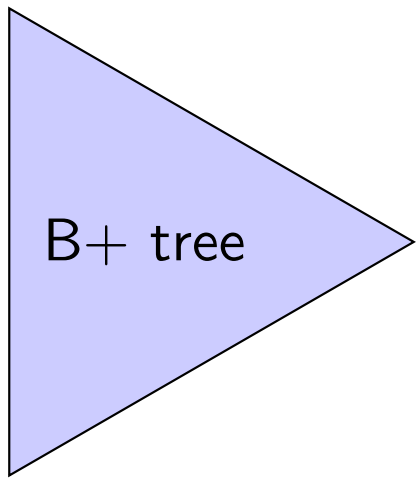
Mapping table

ID	Value
0	y:Abraham_Lincoln
1	hasName
2	"Abraham Lincoln"
3	bornOnDate
4	"1809-02-12"
5	diedOnDate
6	"1865-04-15"
7	y:Washington_DC
8	"Washington D.C."
9	foundingYear
10	"1790"

Exhaustive Indexing

- RDF-3X [Neumann and Weikum, PVLDB 08] , Hexastore [Weiss et al., PVLDB 08]
- Strings are mapped to ids using a mapping table
- Triples are indexed in a clustered B+ tree in lexicographic order

Subject	Property	Object
0	1	2
0	3	4
0	5	6
7	1	8
7	9	10

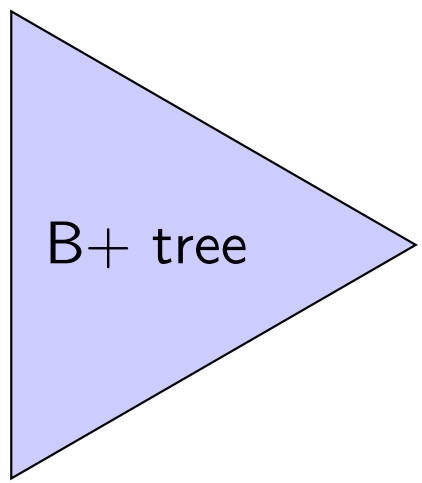


Easy querying through mapping table

Exhaustive Indexing

- RDF-3X [Neumann and Weikum, PVLDB 08] , Hexastore [Weiss et al., PVLDB 08]
- Strings are mapped to ids using a mapping table
- Triples are indexed in a clustered B+ tree in lexicographic order
- Create indexes for permutations of the three columns: SPO, SOP, PSO, POS, OPS, OSP

Subject	Property	Object
0	1	2
0	3	4
0	5	6
7	1	8
7	9	10



Easy querying through mapping table

Exhaustive Indexing–Query Execution

- Each triple pattern can be answered by a range query
- Joins between triple patterns computed using merge join
- Join order is easy due to extensive indexing

Subject	Property	Object
0	1	2
0	3	4
0	5	6
7	1	8
7	9	10
⋮	⋮	⋮

ID	Value
0	y:Abraham_Lincoln
1	hasName
2	"Abraham Lincoln"
3	bornOnDate
4	"1809-02-12"
5	diedOnDate
6	"1865-04-15"
7	y:Washington_DC
8	"Washington D.C."
9	foundingYear
10	"1790"

Exhaustive Indexing–Query Execution

- Each triple pattern can be answered by a range query
- Joins between triple patterns computed using merge join
- Join order is easy due to extensive indexing

Advantages

- Eliminates some of the joins – they become range queries
- Merge join is easy and fast

7	1	8
7	9	10
⋮	⋮	⋮

3	bornOnDate
4	"1809-02-12"
5	diedOnDate
6	"1865-04-15"
7	y:Washington_DC
8	"Washington D.C."
9	foundingYear
10	"1790"

Exhaustive Indexing–Query Execution

- Each triple pattern can be answered by a range query
- Joins between triple patterns computed using merge join
- Join order is easy due to extensive indexing

Advantages

- Eliminates some of the joins – they become range queries
- Merge join is easy and fast

Disadvantages

- Space usage

6	"1865-04-15"
7	y:Washington_DC
8	"Washington D.C."
9	foundingYear
10	"1790"

Outline

RDF Introduction

gStore: a graph-based SPARQL query engine

Answering SPARQL queries using graph pattern matching [Zou et al., PVLDB 2011, VLDB J 2014]

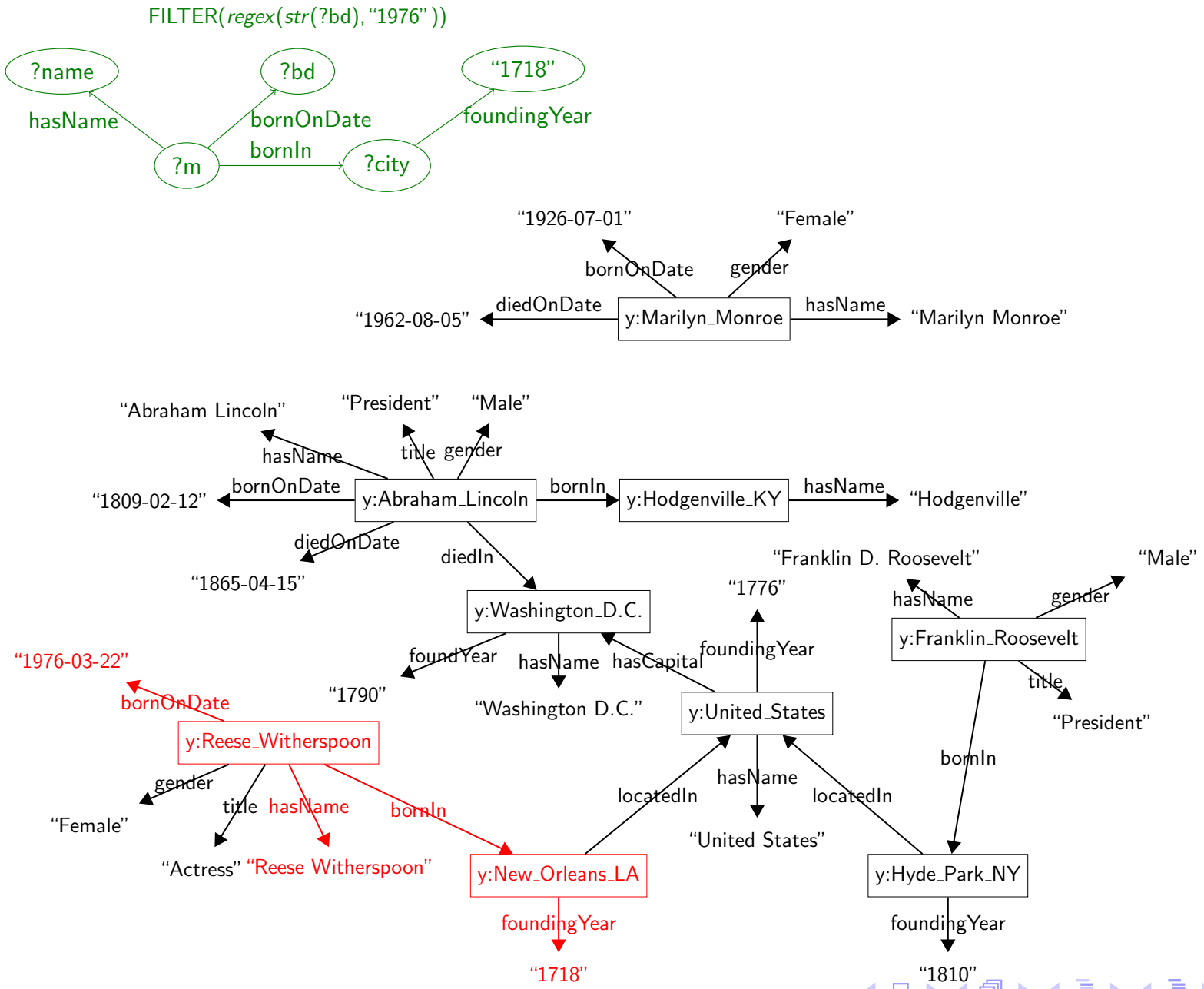
gAnswer: Natural Language Question Answering over RDF

A Data Driven Approach [Zou et al., SIGMOD 2014; Zheng et al., SIGMOD 2015]

gStore – General Idea

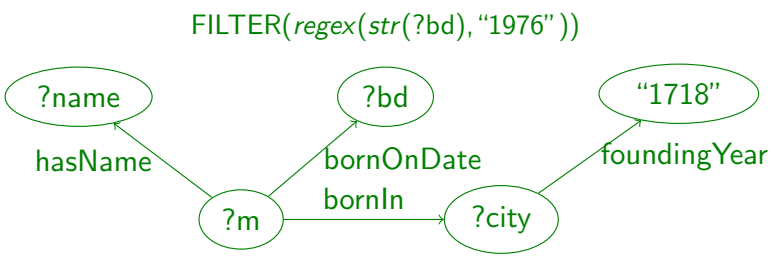
- We work directly on the RDF graph and the SPARQL query graph
  - Answering SPARQL query $\equiv$ subgraph matching
  - Subgraph matching is computationally expensive
- Use a signature-based encoding of each entity and class vertex to speed up matching
- Filter-and-evaluate
  - Use a false positive algorithm to prune nodes and obtain a set of candidates; then do more detailed evaluation on those
- We develop an index (VS*-tree) over the data signature graph (has light maintenance load) for efficient pruning

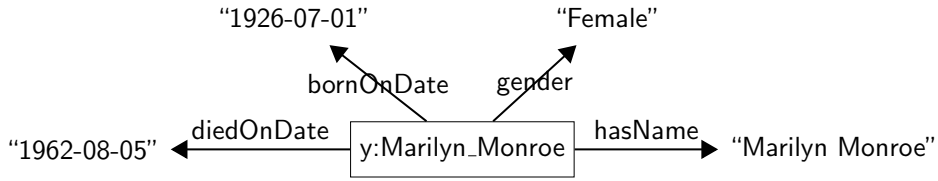
0. Start with RDF Graph *G*

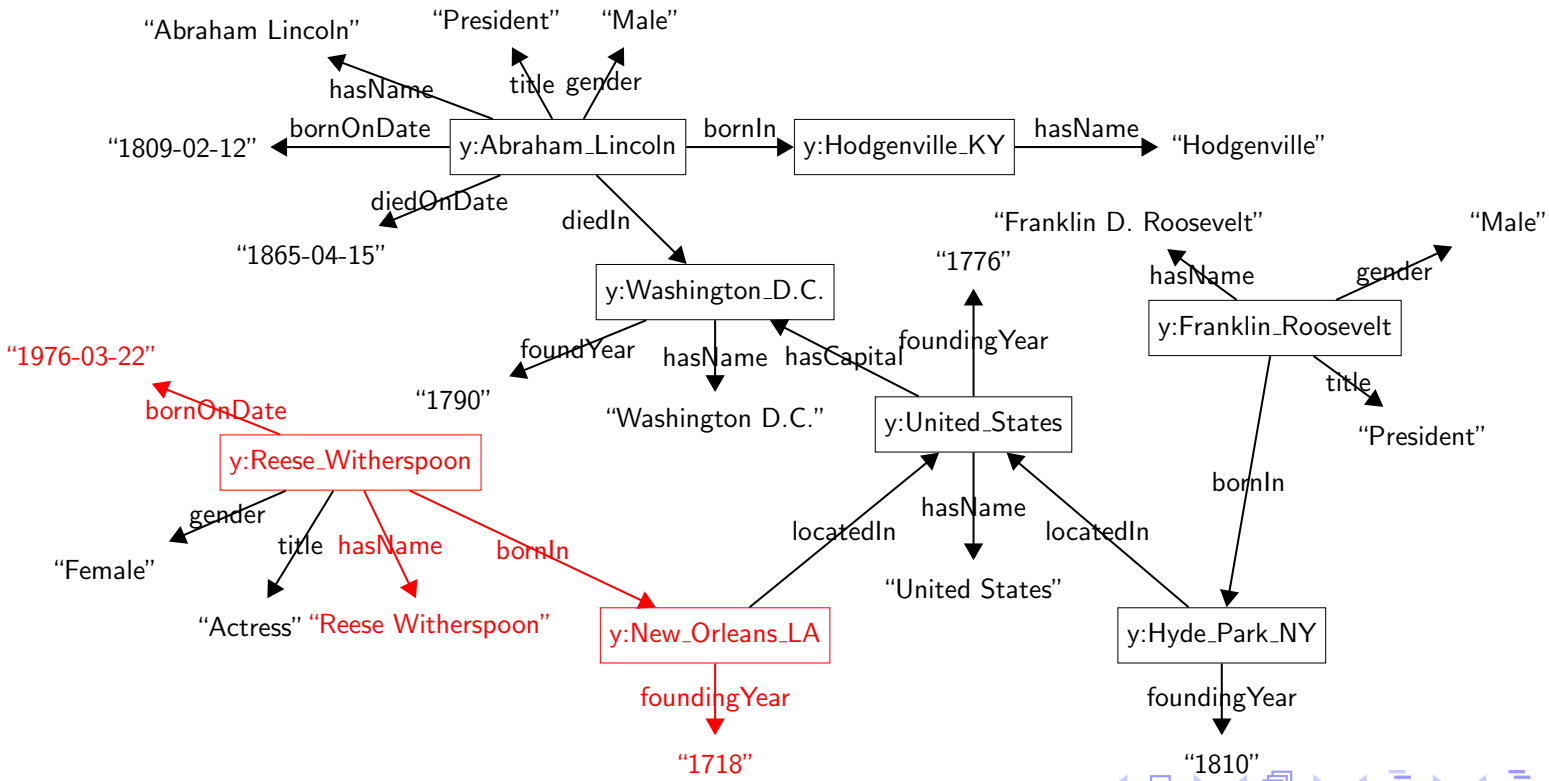


0. Start with RDF Graph G

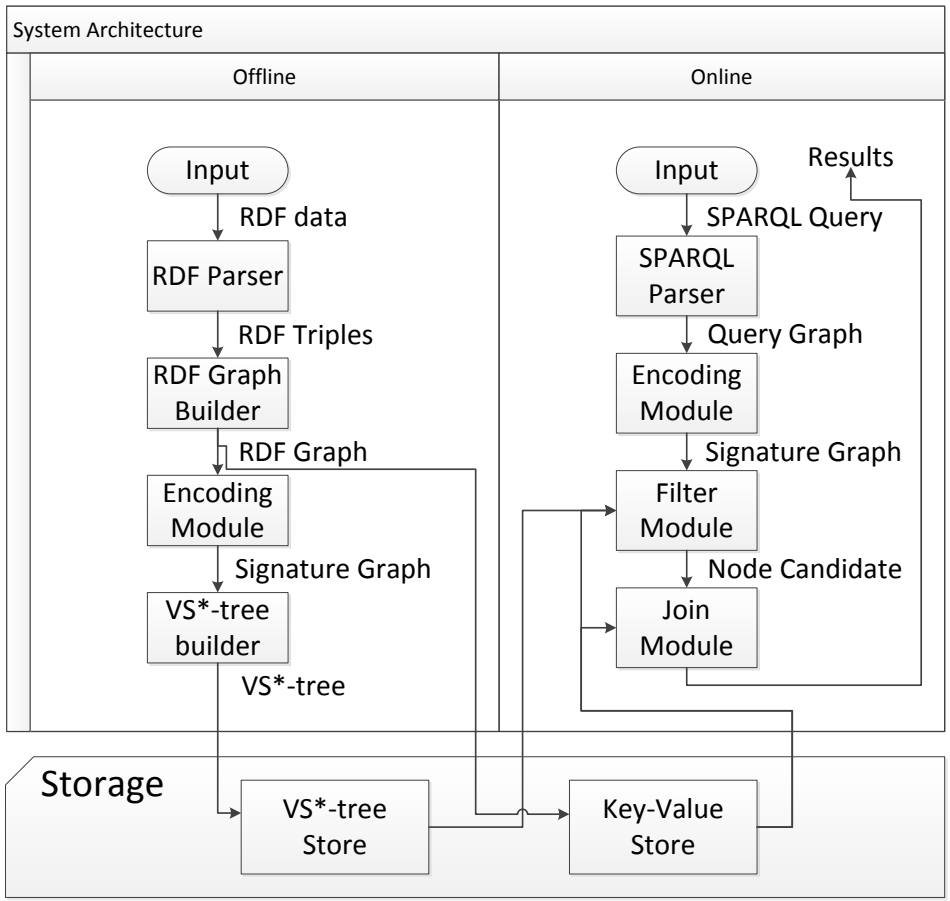
Finding matches over a large graph is not a trivial task!







gStore


System Architecture
Offline
Online
Input
RDF data
RDF Parser
RDF Triples
RDF Graph Builder
RDF Graph
Encoding Module
Signature Graph
VS*-tree builder
VS*-tree
Input
SPARQL Query
Results
SPARQL Parser
Query Graph
Encoding Module
Signature Graph
Filter Module
Node Candidate
Join Module
Storage
VS*-tree Store
Key-Value Store

gStore

C++ API

JAVA API

gStore


Queries:
subject
subLocation
predict
object
objLocation
?p
Draw it!
wasBornIn
?c
Draw it!
temporal:Start
>=1850-XX-XX
End
<=1900-XX-XX
Put it!
Location
Draw it!
?c
Draw it!
type_star
wordnet_city
Draw it!
temporal:Start
End
Put it!
Location
Draw it!
+
-
Time Line
-9999
-3000
0
1000
1500
1600
1700
1750
1800
1850
1900
1950
2000
2050
9999
Use exists examples:
Show
answers in one page
Query
意大利
地图
卫星
混合地图

Peer Review Comments

Charu Aggarwal, ACM Fellow, IBM T. J. Watson Researcher

In the realm of RDF, SPARQL is widely used as the query processing language. However, writing of a SPARQL query is often too challenging, because it requires the exact knowledge of structure, node labels and types. gStore [43], which is the first study that considers a subgraph matching-based query answering technique in RDF data, allows approximate node label matching, but adheres to strict structural matches. In contrast, our NeMa framework permits both structural and node label mismatches.

——NeMa: Fast Graph Search with Label Similarity, Proc. of VLDB: 181-192 (2013)

Outline

RDF Introduction

gStore: a graph-based SPARQL query engine

Answering SPARQL queries using graph pattern matching [Zou et al., PVLDB 2011, VLDB J 2014]

gAnswer: Natural Language Question Answering over RDF

A Data Driven Approach [Zou et al., SIGMOD 2014; Zheng et al., SIGMOD 2015]

gAnswer: Natural Language Question Answering Over Knowledge Graph–A Graph Data Driven Approach

- An Easy-to-Use Interface to Access Knowledge Graph
  - It is interesting to both academia and industry.
  - Interdisciplinary research between database and NLP (natural language processing) communities.

gAnswer: Natural Language Question Answering Over Knowledge Graph–A Graph Data Driven Approach

- An Easy-to-Use Interface to Access Knowledge Graph
  - It is interesting to both academia and industry.
  - Interdisciplinary research between database and NLP (natural language processing) communities.

Running Example

Question: Who was married to an actor that play in Philadelphia ?

Subject	**Property**	**Object**
Antonio_Banderas	type	actor
Antonio_Banderas	spouse	Melanie_Griffith
Antonio_Banderas	starring	Philadelphia_(film)
Philadelphia_(film)	type	film
Jonathan_Demme	director	film
Philadelphia	type	city
Aaron_McKie	bornIn	Philadelphia
James_Anderson	playForTeam	Philadelphia_76ers
Constantin_Stanislavski	create	An_Actor_Prepares
Philadelphia_76ers	type	Basketball_team
An_Actor_Prepares	type	Book

Running Example

Question: Who was married to an actor that play in Philadelphia ?

Subject	**Property**	**Object**
Antonio_Banderas	type	actor
Antonio_Banderas	spouse	Melanie_Griffith
Antonio_Banderas	starring	Philadelphia_(film)
Philadelphia_(film)	type	film
Jonathan_Demme	director	film
Philadelphia	type	city
Aaron_McKie	bornIn	Philadelphia
James_Anderson	playForTeam	Philadelphia_7
Constantin_Stanislavski	create	An_Actor_Prepa
Philadelphia_76ers	type	Basketball_team
An_Actor_Prepares	type	Book

Existing Solutions

Who was married to an actor that play in Philadelphia ?

Translate NL Question to structured queries

```
SELECT ?y
WHERE {
?x starring Philadelphia_(film).
?x type actor.
?x spouse ?y . }
```

Query Processing

Melanie Griffith

Existing Solutions

Who was married to an actor that play in Philadelphia ?

```
SELECT ?y
WHERE {
?x starring Philadelphia_(film).
?x type actor.
?x spouse ?y . }
```

Translate NL Question to structured queries

Philadelpha

↓

Philadelpha_(film)

↓

Philadelpha_76ers

Query Processing

Melanie Griffith

Existing Solutions

Who was married to an actor that play in Philadelphia?

```
SELECT ?y
WHERE {
?x starring Philadelphia_(film).
?x type actor.
?x spouse ?y . }
```

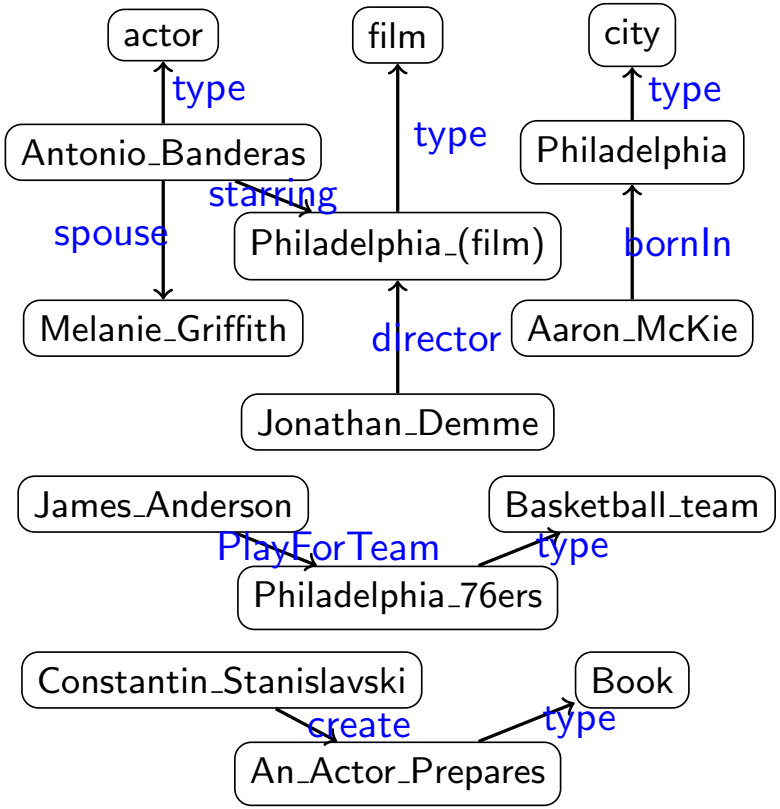
Translate NL Question to structured queries

playForTeam
↓
starring
↓
director

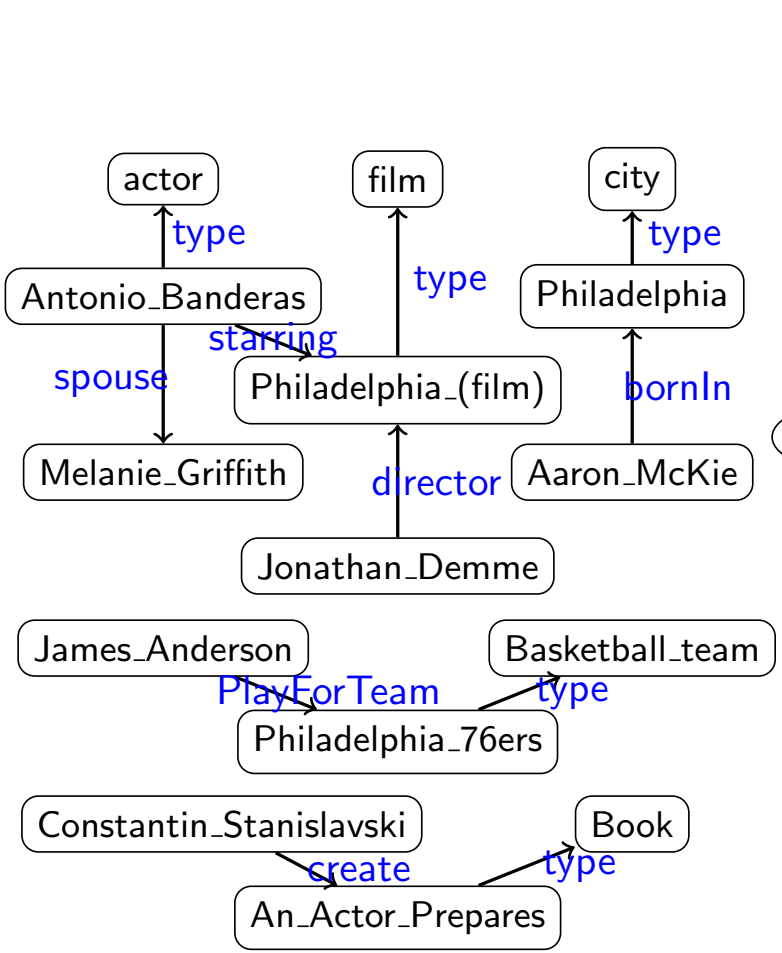
Query Processing

Melanie Griffith

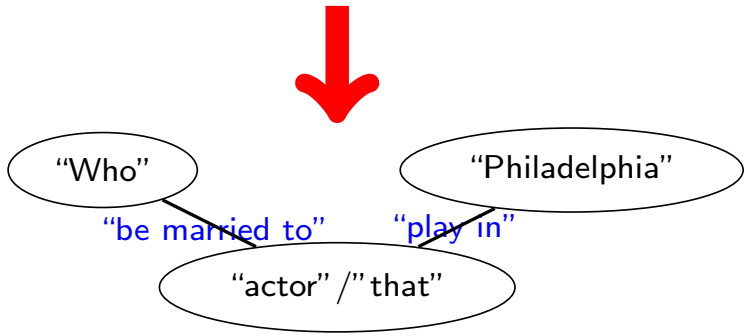
Our Method: Motivation–Data Driven



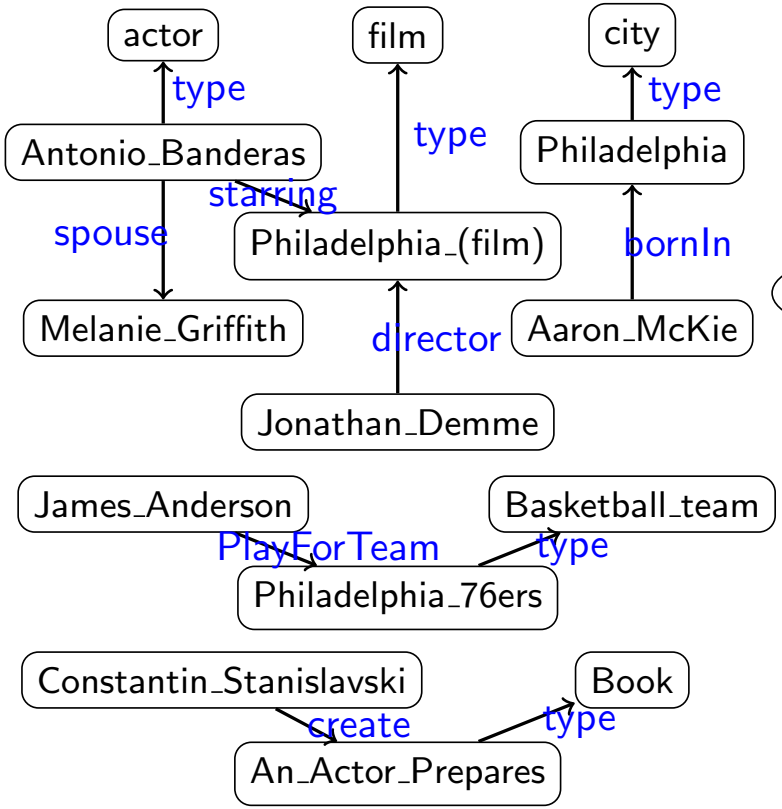
Our Method: Motivation–Data Driven

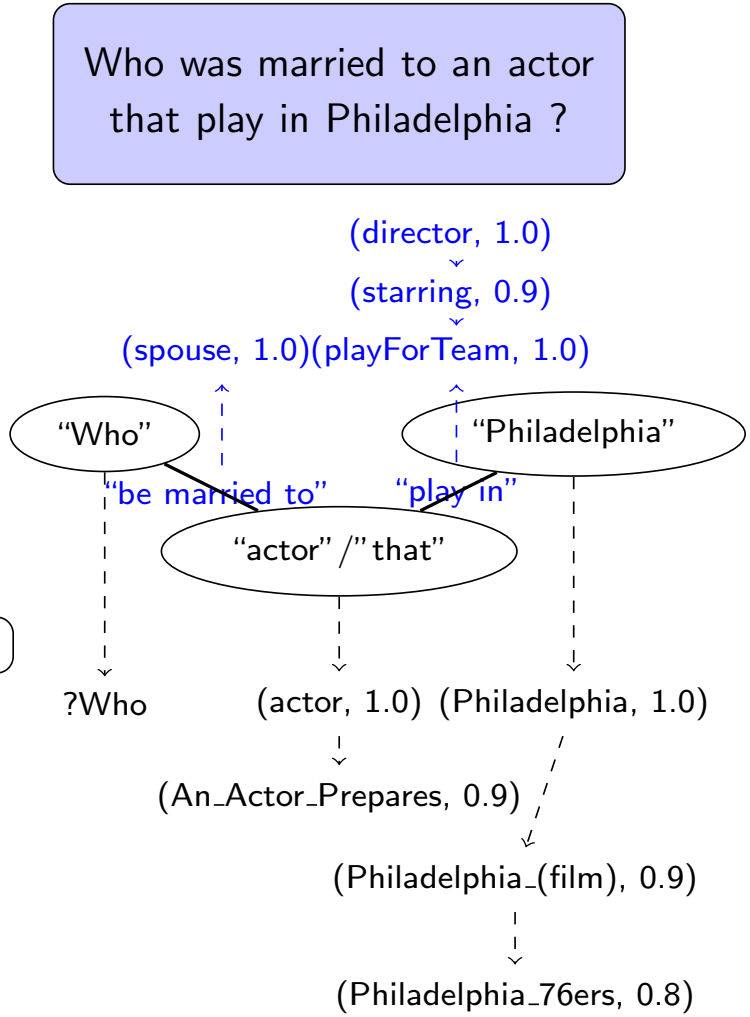


Who was married to an actor that play in Philadelphia ?

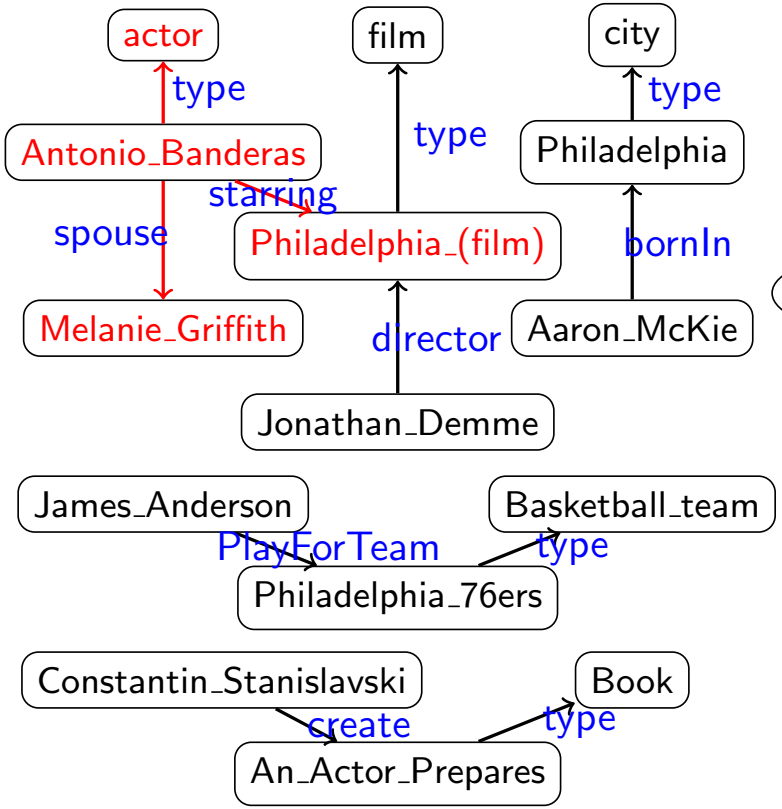


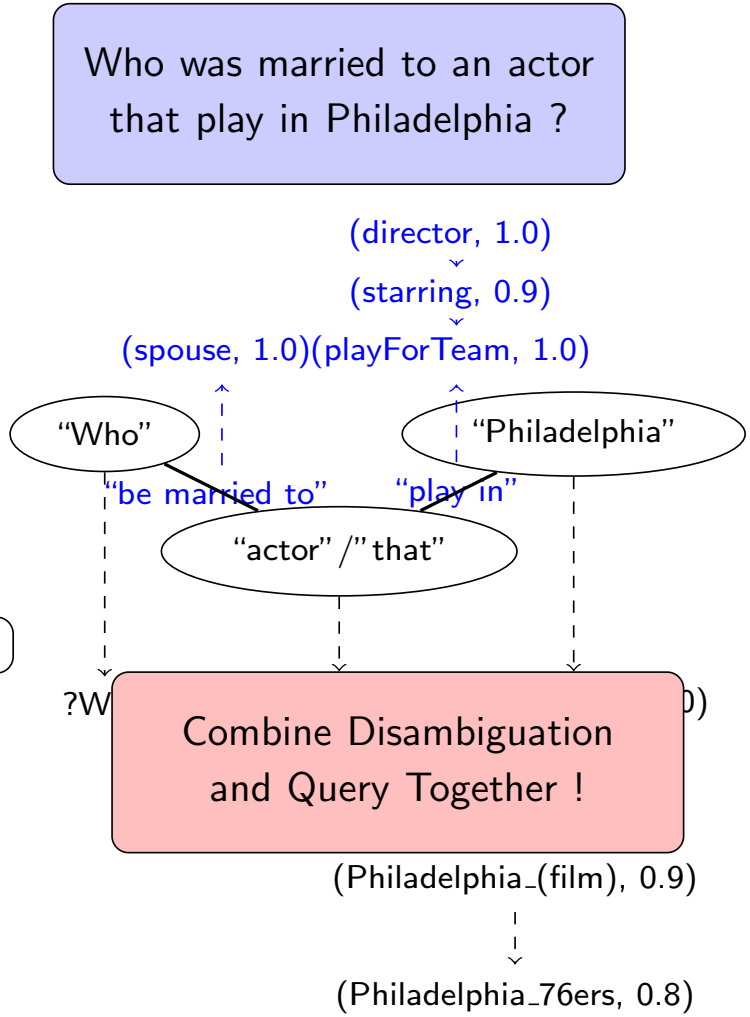
Our Method: Motivation–Data Driven





Our Method: Motivation–Data Driven





Experiments: Datasets

- RDF repository: DBPedia

Table : Statistics of RDF Graph

	DBpedia
Number of Entities	5.2 million
Number of Triples	60 million
Number of Predicates	1643
Size of RDF Graphs (in GB)	6.1

- Relation Phrase Dictionary: Patty

Table : Statistics of Relation Phrase Dataset

	wordnet-wikipedia	**freebase-wikipedia**
Number of Textual Patterns	350,568	1,631,530
Number of Entity Pairs	3,862,304	15,802,947
Average Entity Pair Number For Each Pattern	11	9

Experiments: Online

Benchmark: QALD-3, 99 Natural Language Questions

Table : Evaluating QALD-3 Testing Questions (on DBpedia)

	Processed	Right	Partially	Recall	Precision	F-1
Our Method	76	32	11	0.40	0.40	0.40
squall2sparql	96	77	13	0.85	0.89	0.87
CASIA	52	29	8	0.36	0.35	0.36
Scalewelis	70	1	38	0.33	0.33	0.33
RTV	55	30	4	0.34	0.32	0.33
Intui2	99	28	4	0.32	0.32	0.32
SWIP	21	14	2	0.15	0.16	0.16
DEANNA	27	21	0	0.21	0.21	0.21

Experiments: Online

ID	Questions	Response Time (in ms)
Q2	Who was the successor of John F. Kennedy?	1699
Q3	Who is the mayor of Berlin?	677
Q14	Give me all members of Prodigy?	811
Q17	Give me all cars that are produced in Germany ?	297
Q19	Give me all people that were born in Vienna and died in Berlin ?	2557
Q20	How tall is Michael Jordan ?	942
Q21	What is the capital of Canada ?	1342
Q22	Who is the governor of Wyoming ?	796
Q24	Who was the father of Queen Elizabeth II?	538
Q27	Sean Parnell is the governor of which U.S. state ?	1210
Q28	Give me all movies directed by Francis Ford Coppola.	577
Q30	What is the birth name of Angela Merkel ?	250
Q35	Who developed Minecraft ?.	2565
Q39	Give me all companies in Munich.	1312
Q41	Who founded Intel?	1105
Q42	Who is the husband of Amanda Palmer ?	1418
Q44	Which cities does the Weser flow through ?	1139
Q45	Which countries are connected by the Rhine ?	736
Q54	What are the nicknames of San Francisco ?	321
Q58	What is the time zone of Salt Lake City ?	316
Q63	Give me all Argentine films.	427
Q70	Is Michelle Obama the wife of Barack Obama ?	316
Q74	When did Michael Jackson die ?	258
Q76	List the children of Margaret Thatcher.	1139
Q77	Who was called Scarface?	719
Q81	Which books by Kerouac were published by Viking Press ?	796
Q83	How high is the Mount Everest ?	635
Q84	Who created the comic Captain America ?	589
Q86	What is the largest city in Australia ?	1419
Q89	In which city was the former Dutch queen Juliana buried ?	1700
Q98	Which country does the creator of Miffy come from ?	2121
Q100	Who produces Orangina ?	367

Experiments: Online

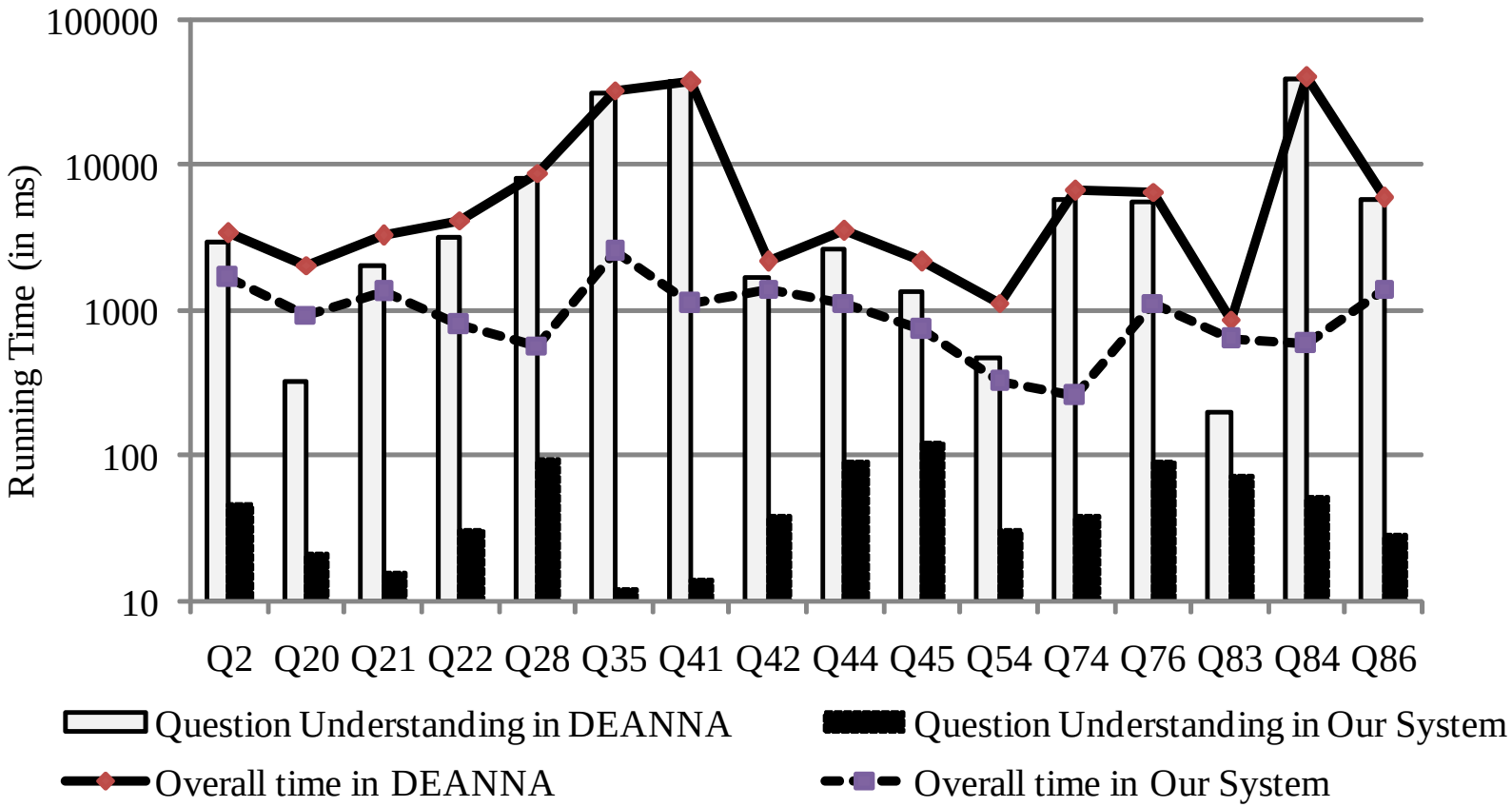


Figure : Online Running Time Comparison

Experiments: Online

QUESTION ANSWERING OVER LINKED DATA

QALD-4: Evaluation results

Workshop website: http://www.sc.cit-ec.uni-bielefeld.de/qald

Results[1] *for Task 1: Multilingual question answering over DBpedia*

	Total	Processed	Right	Partially	Recall	Precision	F-measure
Xser	50	40	34	6	0.71	0.72	**0.72**
gAnswer	50	25	16	4	0.37	0.37	**0.37**
CASIA	50	26	15	4	0.40	0.32	**0.36**
Intui3	50	33	10	4	0.25	0.23	**0.24**
ISOFT	50	28	10	3	0.26	0.21	**0.23**
RO_FII	50	50	6	0	0.12	0.12	**0.12**

Figure : QALD-4 Results

An Example: using gAnswer+gStore in CDBLP

An Example: using gAnswer+gStore in CDBLP

An Example: using gAnswer+gStore in CDBLP

Conclusions

- Graph Database is a Possible Way for RDF Knowledge Base Management.

Conclusions

- Graph Database is a Possible Way for RDF Knowledge Base Management.
- Subgraph Matching is a Strong Tool.

Conclusions

- Graph Database is a Possible Way for RDF Knowledge Base Management.
- Subgraph Matching is a Strong Tool.
- Using RDF repository, how to Provide Knowledge Services for Applications and Common Users?

Thank you!

gStore

gAnswer