

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 **semantics** 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

Structured Data Testing Tool

URL:

HTML

```
<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">
```

PREVIEW

Examples

Maximum 1500 characters allowed. Over-length text will be truncated.

Google search results

Google Custom Search

Preview

What's Sematic Web: Google Snippet

Extracted structured data

rdfa-node

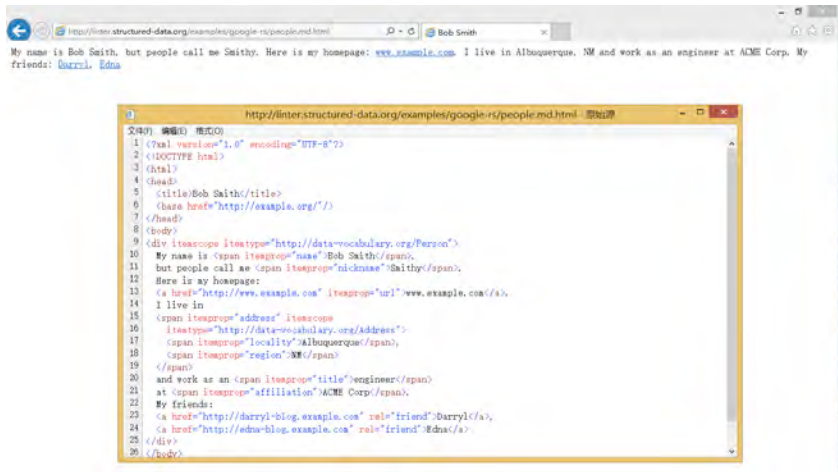
relationship:

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



The image shows a web browser window displaying a Google snippet for a person's homepage. The browser's address bar shows the URL `http://linter.structured-data.org/examples/google-rs/people.md.html`. The snippet text reads: "My name is Bob Smith, but people call me Smithy. Here is my homepage: [www.example.com](#). I live in Albuquerque, NM and work as an engineer at ACME Corp. My friends: [Darryl](#), [Edna](#)".

Below the browser window, a separate window displays the HTML code for the snippet. The code is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html>
3 <html>
4 <head>
5 <title>Bob Smith</title>
6 <base href="http://example.org/" />
7 </head>
8 <body>
9 <div iterscope itesstype="http://data-vocabulary.org/Person">
10 My name is <span itesprop="name">Bob Smith</span>.
11 but people call me <span itesprop="nickname">Smithy</span>.
12 Here is my homepage:
13 <a href="http://www.example.com" itesprop="url">www.example.com</a>.
14 I live in
15 <span itesprop="address" itescope
16 itesstype="http://data-vocabulary.org/Address">
17 <span itesprop="locality">Albuquerque</span>,
18 <span itesprop="region">NM</span>
19 </span>
20 and work as an <span itesprop="title">engineer</span>
21 at <span itesprop="affiliation">ACME Corp</span>.
22 My friends:
23 <a href="http://darryl-blog.example.com" rel="friend">Darryl</a>,
24 <a href="http://edna-blog.example.com" rel="friend">Edna</a>.
25 </div>
26 </body>
```

What's Sematic Web: Google Snippet

Enhanced search result preview

*Disclaimer: this preview is only shown as an 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

inter.structured-data.org/examples/google-ra/people.md.html

Albuquerque, NM - engineer, ACME Corp.

an actual search result **may** display other content **relating** to your search terms here.

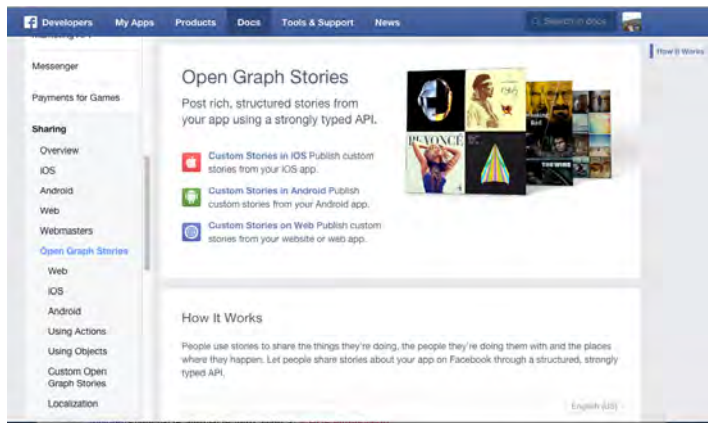
Raw structured data extracted from the page:

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://inter.structured-data.org/examples/google-ra/people.md.html	

Parser statistics

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

What's Sematic Web: Facebook Social Graph



The screenshot shows the Facebook Developers website with the 'Open Graph Stories' section highlighted in the left sidebar. The main content area features the title 'Open Graph Stories' and a description: 'Post rich, structured stories from your app using a strongly typed API.' Below this, there are three links with corresponding icons: 'Custom Stories in iOS', 'Custom Stories in Android', and 'Custom Stories on Web'. To the right of these links is a collage of various story images. At the bottom of the main content area, there is a section titled 'How It Works' with a brief explanation of how stories are used on Facebook. The footer of the page includes a language selector set to 'English (US)'.

Facebook Developers | My Apps | Products | Docs | Tools & Support | News

Search in docs

How It Works

Open Graph Stories

Post rich, structured stories from your app using a strongly typed API.

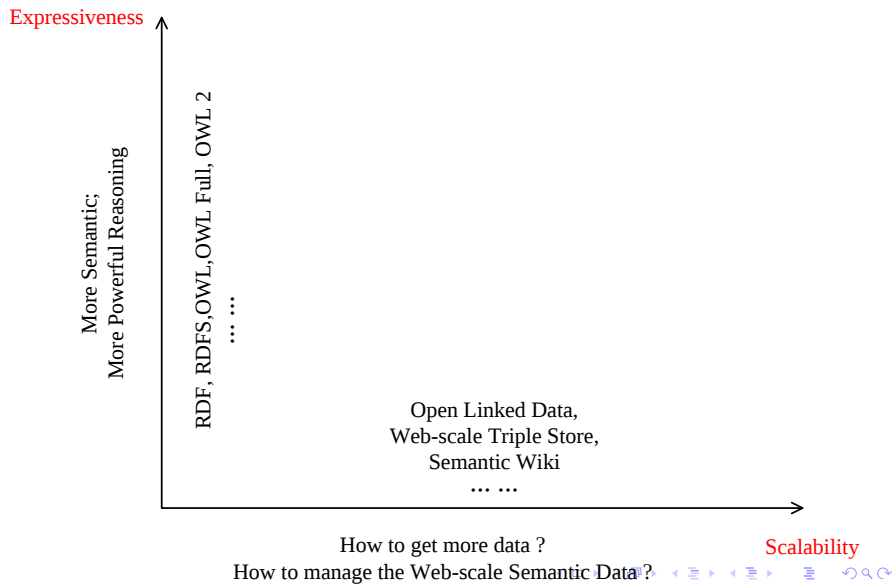
-  Custom Stories in iOS Publish custom stories from your iOS app.
-  Custom Stories in Android Publish custom stories from your Android app.
-  Custom Stories on Web Publish custom stories from your website or web app.

How It Works

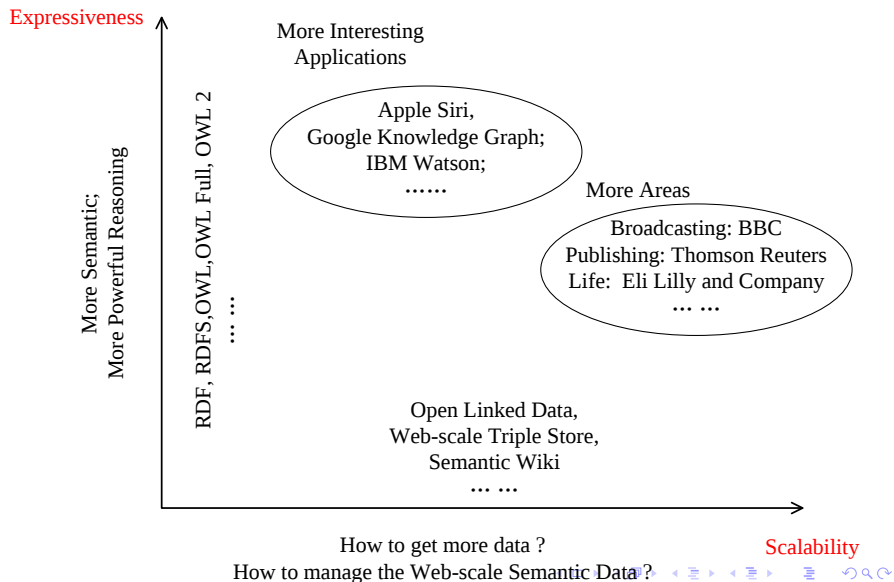
People use stories to share the things they're doing, the people they're doing them with and the places where they happen. Let people share stories about your app on Facebook through a structured, strongly typed API.

English (US)

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.

The screenshot shows the Amazon Echo interface. At the top, a question is entered: "Who is the wife of the current president of the us?". Below the question, two possible answers are displayed. The first answer is for "Michelle Obama", with a brief biography and a photo. The second answer is for "Anita Thigpen Perry", also with a brief biography. At the bottom, there are buttons to "Rate this answer", "Report Abuse", and "How do we know?".

Who is the wife of the current president of the us?

You asked: Who is the wife of the current president of the us?

Michelle Obama

Michelle LaVaughn Robinson Obama (born January 17, 1964), the wife of the forty-fourth President of the United States, Barack Obama, and is the first African-American First Lady of the United States

website wikipedia

Anita Thigpen Perry

Anita Thigpen Perry (born March 5, 1952), the current First Lady of Texas, and the wife of Governor Rick Perry

website wikipedia

Rate this answer: Report Abuse

How do we know?

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

SafeSearch

About: 127,908,908 results (0.25 seconds)

[Barack Obama - Wikipedia, the free encyclopedia](#)
in [wikipedia.org/wiki/Barack_Obama](#) - Cached
Barack Hussein **Obama II** is the 44th and current President of the United States. He is the first African American to hold the office. Born in Honolulu, Hawaii, ...
[Early life and career of Barack](#) ... [Family of Barack Obama](#) - [Obama administration](#)

[News for Obama](#)

 [Obama Embraces 'Obamacare' Says It's Here to Stay](#)
TIME - 2 hours ago
(TOLEDO, Ohio) - In his warm-up for the Democratic National Convention, President Barack **Obama** is tangling with a couple of mals, only one ...

[Obama heads for Isaac-soaked South, as convention delegates gather in North Carolina](#)
Montreal Gazette - 57 minutes ago

[Obama hits Romney-Obamacare slam, says 'I do care'](#)
The Associated Press - 11 hours ago

[Barack Obama](#)
[www.barackobama.com/](#) - Cached
BarackObama.com is the official re-election campaign website of President Barack **Obama**. Visit the site for the latest updates from the **Obama** campaign, ...
[Store](#) - [Contact Us](#) - [Jobs](#) - [Volunteer](#)

[President Barack Obama | The White House](#)
[www.whitehouse.gov/administration/president-obama](#) - Cached

Barack Obama



Barack Hussein Obama II is the 44th and current President of the United States. He is the first African American to hold the office. [Wikipedia](#)

Born: August 4, 1961 (age 51), Honolulu
Full name: Barack Hussein Obama II
Net worth: US\$ 11.8 million (2010)

Education: Harvard Law School (1990–1991), Columbia University (1983), More
Siblings: Maya Soetoro-Ng, George Obama, Mark Ndesandjo

Books

Dreams from My Father

The Audacity of Hope

Of Two I Sing

Change We Can Believe In

Barack Obama in His Own Words

RDF Uses

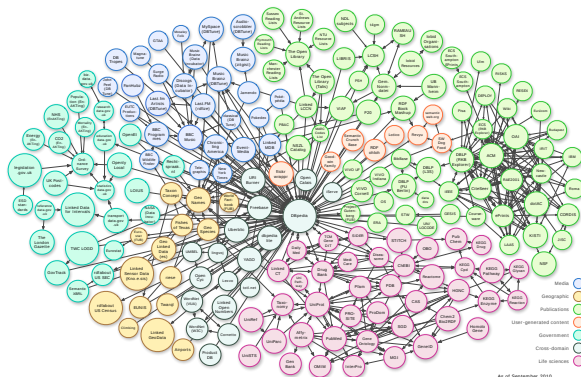
- ▶ Yago and DBPedia extract facts from Wikipedia & represent as RDF → structural queries
- ▶ Communities build RDF data
 - ▶ E.g., biologists: Bio2RDF and Uniprot RDF
- ▶ Web data integration
 - ▶ Linked Data Cloud
- ▶ ...

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

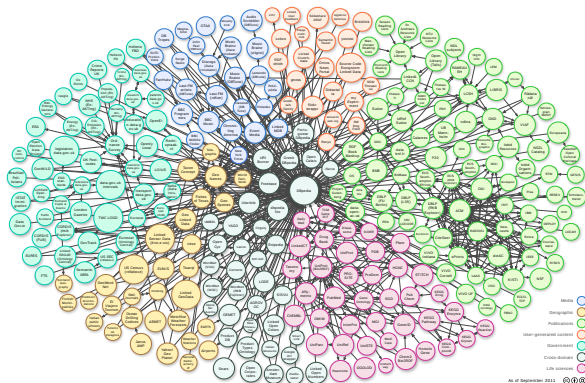


September '10:
203 datasets

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

RDF Data Volumes ...

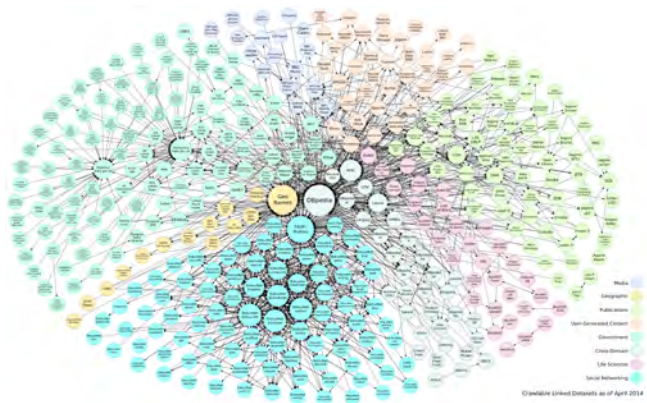
- ▶ ... are growing – and fast
 - ▶ Linked data cloud currently consists of 325 datasets with >25B triples
 - ▶ Size almost doubling every year



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

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"

Abraham.Lincoln:DiedIn



y:Washington.DC

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”

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"

Abraham.Lincoln:DiedIn



y:Washington.DC

RDF Data Model

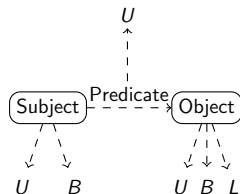
- ▶ Triple: Subject, Predicate (Property), Object (s, p, o)

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

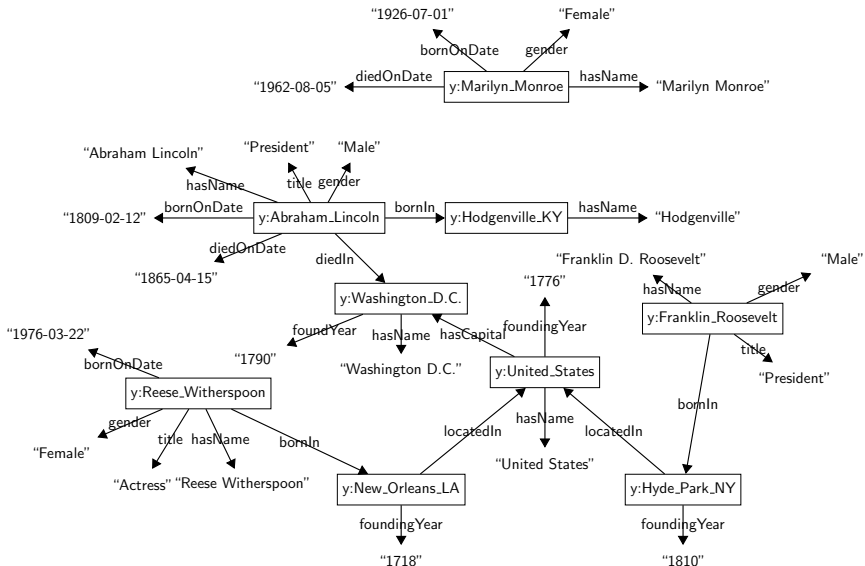
Subject	Predicate	Object
y: Abraham_Lincoln	hasName	"Abraham Lincoln"
y: Abraham_Lincoln	BornOnDate	"1809-02-12"
y: Abraham_Lincoln	DiedOnDate	"1865-04-15"
y: Abraham_Lincoln	bornIn	y: Hodgenville_KY
y: Abraham_Lincoln	DiedIn	y: Washington_DC
y: Abraham_Lincoln	title	"President"
y: Abraham_Lincoln	gender	"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



RDF Query Model

- ▶ Query Model - **SPARQL Protocol and RDF Query Language**
- ▶ 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

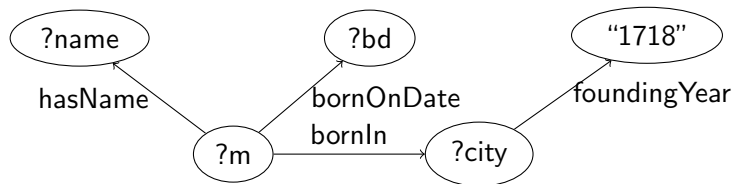
$$(U \cup V) \times (U \cup V) \times (U \cup V \cup L)$$

- ▶ $?x$ `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.property="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> ?y .
  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"

Too many self-joins!

```
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.property="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	openingYear
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

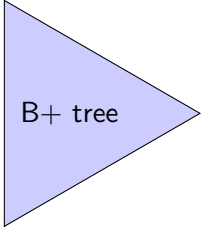
B+ tree

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



B+ tree

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

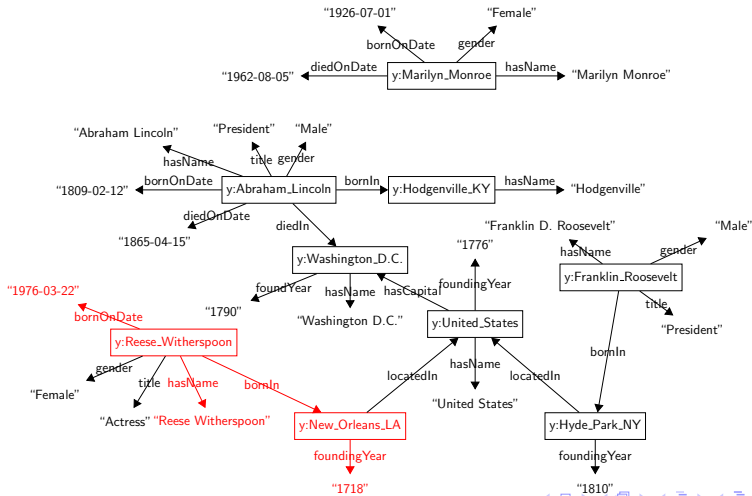
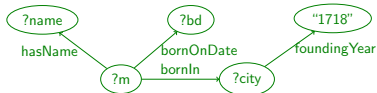
A Data Driven Approach [Zou et al., SIGMOD 2014; Zheng et al., SIGMOD 2015]

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

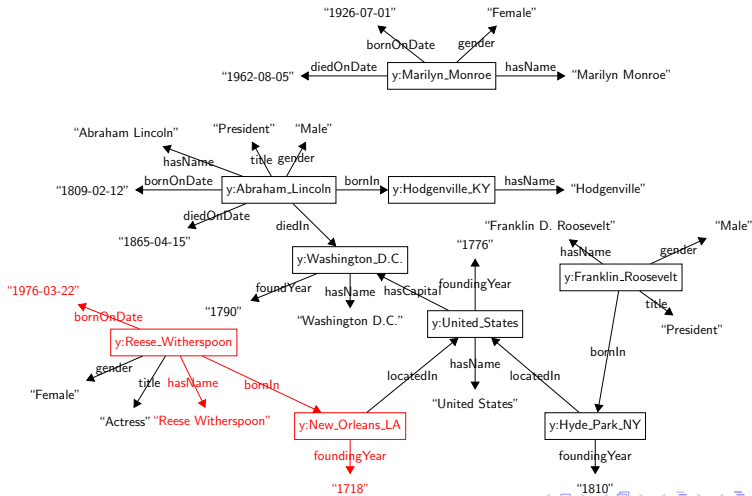
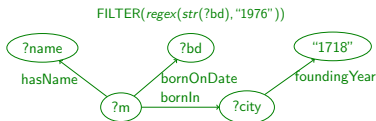
0. Start with RDF Graph G

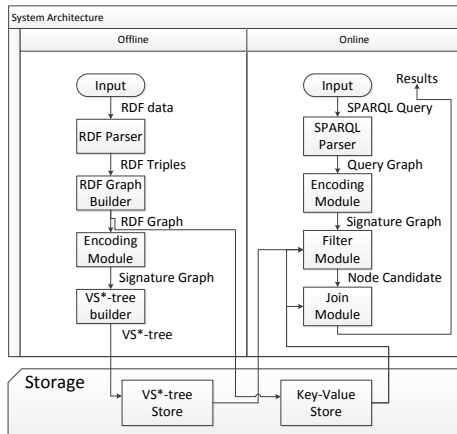
`FILTER(regex(str(?bd), "1976"))`



0. Start with RDF Graph G

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





gStore

C++ API

```
1 #include "GStoreConnector.h"
2 #include <stdio.h>
3 #include <iostream>
4
5 // Before run this example, you must start up the GStore server at first (use command ./gstore).
6 int main(int argc, char * argv[])
7 {
8     // Initialize the GStore server's IP address and port.
9     GStoreConnector gc("127.0.0.1", 8080);
10
11     // Build a new database by a TXT file.
12     // Note that the relative path is related to gstore.
13     gc.Build("db_100000", "example/conf_scripts/DB_100000.txt");
14
15     // When you can execute through query in this database.
16     std::string query = "SELECT * FROM T";
17
18     // 1.
19     gc.setType(GStoreConnector::GSTORETEXT);
20     gc.setName("GSTORETEXT");
21     gc.setTableName("T");
22     gc.setTableName("T");
23     gc.setTableName("T");
24     gc.setName("GSTORETEXT");
25     gc.setTableName("T");
26     gc.setName("GSTORETEXT");
27     gc.setName("GSTORETEXT");
28
29     std::string answer = gc.query(query);
30     std::cout << answer << std::endl;
31
32     // Build this database.
33     gc.Unload("db_100000");
34
35     // Also, you can load some exist database directly and then query.
36     gc.Load("db_100000");
37     answer = gc.query(query);
38     std::cout << answer << std::endl;
39     return 0;
40 }
```

JAVA API

```
1 import java.GStoreConnector;
2
3 // Before run this example, you must start up the GStore server at first (use command ./gstore).
4 public class JavaExample
5 {
6     // Initialize the GStore server's IP address and port.
7     GStoreConnector gc = new GStoreConnector("127.0.0.1", 8080);
8
9     // Build a new database by a TXT file.
10     // Note that the relative path is related to gstore.
11     gc.Build("db_100000", "example/conf_scripts/DB_100000.txt");
12
13     // When you can execute through query in this database.
14     String query = "SELECT * FROM T";
15
16     // 1.
17     gc.setType(GStoreConnector.GSTORETEXT);
18     gc.setName("GSTORETEXT");
19     gc.setTableName("T");
20     gc.setTableName("T");
21     gc.setTableName("T");
22     gc.setName("GSTORETEXT");
23     gc.setTableName("T");
24     gc.setName("GSTORETEXT");
25     gc.setName("GSTORETEXT");
26
27     String answer = gc.query(query);
28     System.out.println(answer);
29
30     // Build this database.
31     gc.Unload("db_100000");
32
33     // Also, you can load some exist database directly and then query.
34     gc.Load("db_100000");
35     answer = gc.query(query);
36     System.out.println(answer);
37 }
```

Queries:

subject	sublocation	predict	object	objLocation
?s		wasBornIn	?c	g36.1312200154265.4
temporal:Start	>=1950-XX-XX	End	<=1900-XX-XX	
Location				
?c		type_star	wordnet_city	
temporal:Start		End		
Location				

Time Line

-9999 -3000 0 1000 1500 1600 1700 1750 1800 1850 1900 1950 2000 2050 9999

Use exists examples:

Show

Query

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.

 **gAnswer**

3 results in total(10.76 seconds).

gAnswer

Pretty_Woman

Runaway_Bride_(1999_film)

Valentine's_Day_(film)

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_Prepare
Philadelphia_76ers	type	Basketball_team
An_Actor_Prepare	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_76ers
Constantin_Stanislavski	create	An_Actor_Prepare
Philadelphia_76ers	type	Basketball_team
An_Actor_Prepare	type	Book



Melanie Griffith

Existing Solutions

Who was married to an actor that play in Philadelphia ?

SELECT ?y  Translate NL Question to structured queries

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


Query Processing

Melanie Griffith

Existing Solutions

Ambiguity

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

Query Processing

Melanie Griffith

Philadelphia

Philadelphia_(film)

Philadelphia_76ers

Existing Solutions

Ambiguity

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

Query Processing

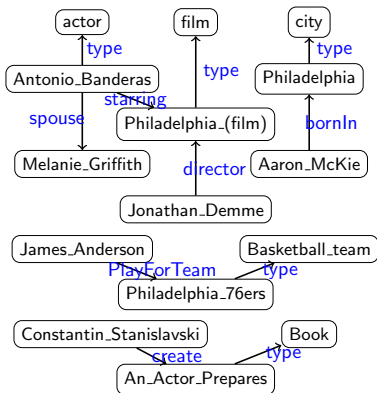
Melanie Griffith

playForTeam

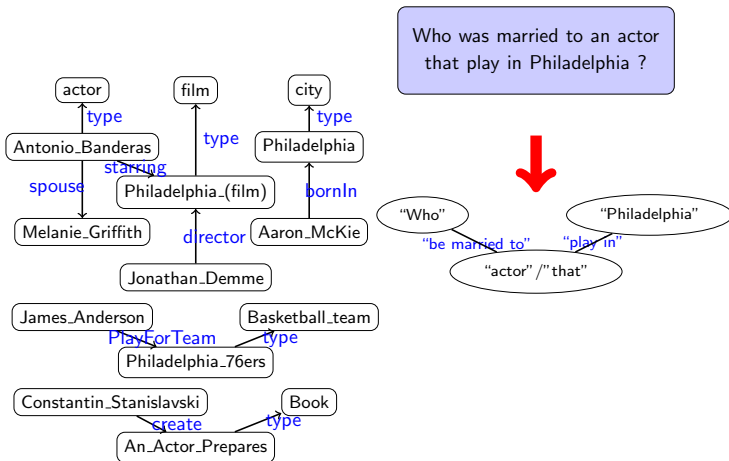
starring

director

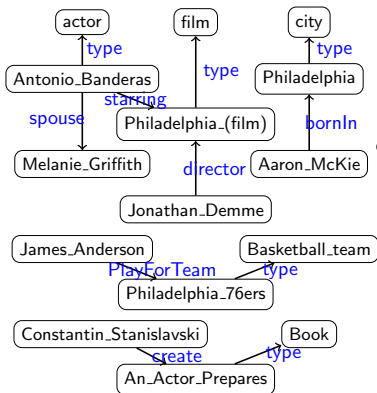
Our Method: Motivation–Data Driven



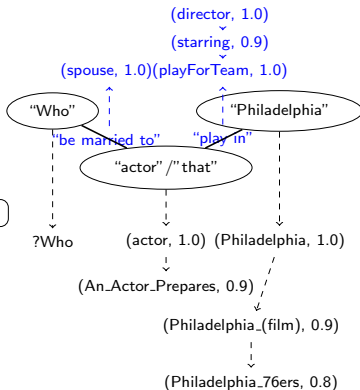
Our Method: Motivation–Data Driven



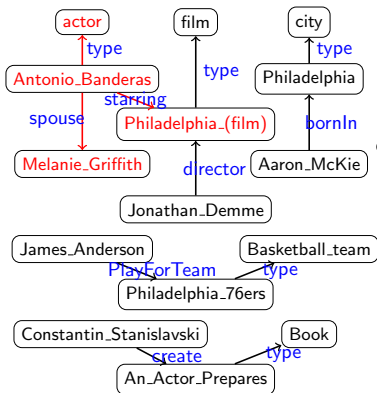
Our Method: Motivation–Data Driven



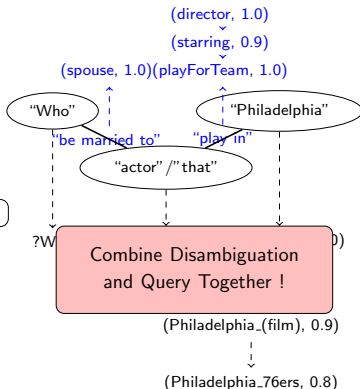
Who was married to an actor that play in Philadelphia ?



Our Method: Motivation–Data Driven



Who was married to an actor
that play in Philadelphia ?



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

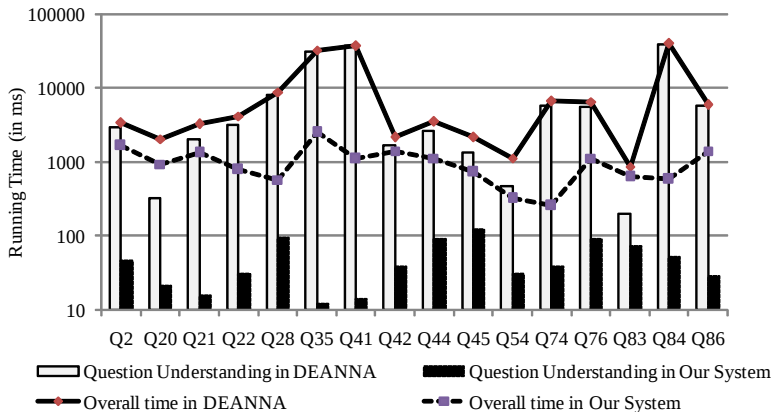


Figure : Online Running Time Comparison

Experiments: Online

QUESTION ANSWERING OVER LINKED DATA QALD-4: Evaluation results

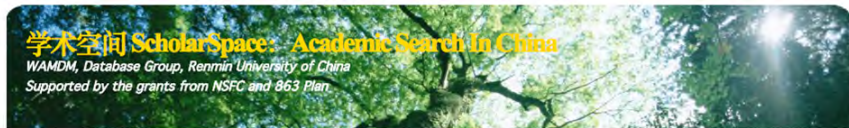
Workshop website: <http://www.sc.cit-ec.uni-bielefeld.de/qald>

Results¹ 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



欢迎关注学术空间微信号

孟小峰 云计算 论文

作者搜索

高级搜索

语义检索

NDW

常用链接: 学术空间ScholarSpace收录的作者列表, 学术主页自动生成系统EasyScholar

期刊与学术会议论文收录情况

- | | | |
|-------------------|-----------------|------------|
| → 软件学报 | → 计算机学报 | → 计算机研究与发展 |
| → 电子学报 | → 中国图象图形学报 | → 中文信息学报 |
| → 计算机科学 | → 小型微型计算机系统 | → 计算机科学与探索 |
| → 中国科学F辑 | → 计算机辅助设计与图形学学报 | |
| → 计算机工程与科学 | → 计算机工程(暂停收录) | |
| → 中国数据库学术会议(NDBC) | | |

计算机中文文献导读

2014年09月



领域版 | 期刊版

2014年

2014年08月



领域版 | 期刊版

2014年

2014年07月



领域版 | 期刊版

2014年

An Example: using gAnswer+gStore in CDBLP

学术空间ScholarSpace
WAMDM, Database Group, Renmin University of China

首页 检索 期刊 分类 机构 会议 关于

学术空间ScholarSpace(C-DBLP)是一个以作者为中心的面向计算机

检索词: 孟小峰 云计算 论文

共检索到结果4条

注: C-DBLP语义搜索功能图查询部分使用了北京大学邹磊老师提供的gAnswer系统, 在此表示感谢。 [了解更多相关信息请点击这里。](#)

4	慈祥 马友忠 孟小峰: 一种云环境下的大数据Top-K查询方法. 软件学报, 2014 (04): 813-826
3	史英杰 孟小峰: 云数据管理系统中查询技术研究综述. 计算机学报, 2013 (02): -
2	孟小峰 慈祥: 大数据管理:概念、技术与挑战. 计算机研究与发展, 2013 (01): -
1	霍峥 孟小峰 徐建良: 云计算中面向隐私保护的查询处理技术研究. 计算机科学与探索, 2012 (05): -

```
SELECT ?title
WHERE (?author :Name "孟小峰".
?paper :Title ?title.
?paper :Keywords ?k.
FILTER(regex(str(?k),"云计算"))
```

An Example: using gAnswer+gStore in CDBLP

**知网空间**
www.cnki.com.cn

主办: 中国知网
全球最大的数字图书馆

充值中心 购买充值卡 新手指南 帮助中心
送卡上门 下载阅读器 广告服务 English

收藏本站

首页 期刊全文数据库 学位论文库 会议论文库 年鉴全文库 学术百科 工具书 CNKI学问 注册 | 登录 | 我的账户

基础科学 | 工程技术I辑 | 工程技术II辑 | 医药卫生科技 | 信息技术 | 农业科技 | 哲学与人文科学 | 社会科学I辑 | 社会科学II辑 | 经济管理

输入关键词

高级搜索 用“Top-K查询云计算”期刊平台检索, 点击这里搜索更多...

中国知网 · 知识服务专家

《软件学报》 2014年04期 [加入收藏](#) [报错](#)

一种云环境下的大数据Top-K查询方法
蔡祥 马友忠 孟小峰

【摘要】: Top-K查询在搜索引擎、电子商务等领域有着广泛的应用.Top-K查询从海量数据中返回最符合用户需求的前K个结果,主要目的是消除信息过载带来的负面影响.大数据背景下的Top-K查询,给数据管理和分析等方面带来新的挑战.结合MapReduce的特点,从数据划分、数据筛选等方面对云环境下的大数据Top-K查询问题进行深入研究.实验结果表明,该方法具有良好的性能和扩展性.

【作者单位】: 中国人民大学信息学院;

【关键词】: Top-K查询 云计算 Map Reduce

【基金】: 国家自然科学基金(61371050,91224008) 国家高技术研究发展计划(863)(2013AA013204) 高等学校博士学科点专项科研基金(20130004130001)

【分类号】: TP311.13

【正文快照】:

知网广告投放
010-62982993
[查看详情>>](#)

相关期刊

- >电子技术应用
- >西安电子科技大学学报
- >材料工程
- >水声译丛
- >计算机工程与应用
- >华中科技大学学报(自然科学—)
- >计算机研究与发展
- >计算机工程
- >系统工程与电子技术

相关机构

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

