Querying the Web of Data with SPARQL-LD

Pavlos Fafalios*
fafalios@ics.forth.gr

Thanos Yannakis
yannakis@ics.forth.gr

Yannis Tzitzikas
tzitzik@ics.forth.gr

Foundation for Research and Technology – Hellas (FORTH)
Institute of Computer Science (ICS)
Information Systems Laboratory (ISL)

University of Crete
Computer Science Department
Greece

** From 1st of June, postdoctoral researcher at L3S Research Center, Hannover*

The topic (in one **slide**)

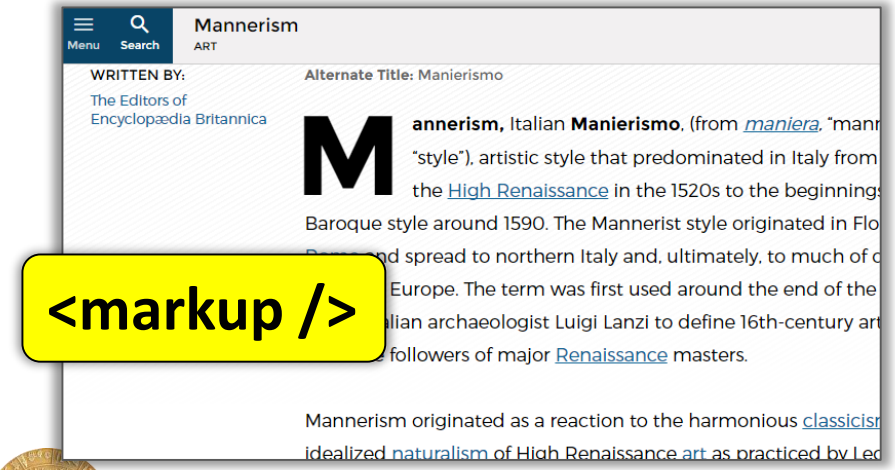
- How to query RDF data that exist on the **Web** in **any standard format**?
  - RDF/XML, N-Triples, N3/Turtle, RDFa, JSON-LD, Microdata
- How to query RDF data **dynamically-created** by Web Services?
- How to **integrate** (**at query-execution time**) data coming from **multiple** and **heterogeneous** web sources?
- How to do it in a **flexible** and **efficient** way?

- **SPARQL-LD (Linked Data)**: a generalization of SPARQL 1.1 Federated Query
  - Extension of **SERVICE** operator enabling to query any HTTP Web source containing RDF data (even derived at query-execution time)

The topic (in one **example**)

europeana
think culture

```
SELECT DISTINCT ?creator ?descr ?photo WHERE {
  SERVICE <http://europeana.ontotext.com/sparql> {
    ?work dc:subject dbr:Renaissance ; dc:creator ?creator }
  SERVICE <http://www.mannerism.org/painters> {
     ?creator dc:subject dbc:Mannerist_painters }
  SERVICE ?creator {
    ?creator foaf:depiction ?photo ;
             dbo:abstract ?descr FILTER(lang(?descr)= "it") } }
```



<markup />

Outline

- Introduction and Background — **5 min**
  - Web of Data, Linked Data
  - RDF, SPARQL
  - Web of Data and Digital Libraries
- Motivation — **5 min**
  - Current approaches on Querying the Web of Data
  - Limitations
- SPARQL-LD — **10 min**
  - Extended SERVICE definition
  - Implementation
  - Examples
  - Optimizations
- Evaluation — **5 min**
- Conclusion — **2 min**

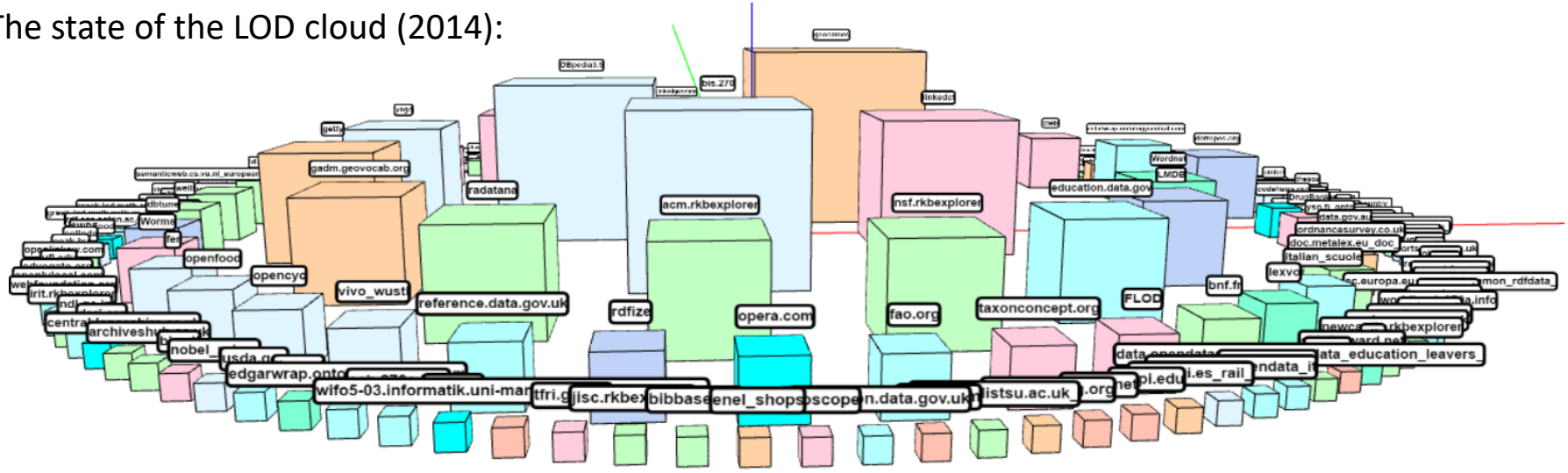
Background

- Web of Data, Linked Data
- RDF, SPARQL
- Web of Data and Digital Libraries

The Web of Data (or Semantic Web)

- Sharing information on the Web in a way that can be processed by machines
- **Linked Open Data (LOD)** describes a method of publishing structured data on the Web so that it can be **interlinked** and become more useful
  - HTTP, URI, RDF
- The **LOD** cloud: datasets published following the "Linked Data" principles

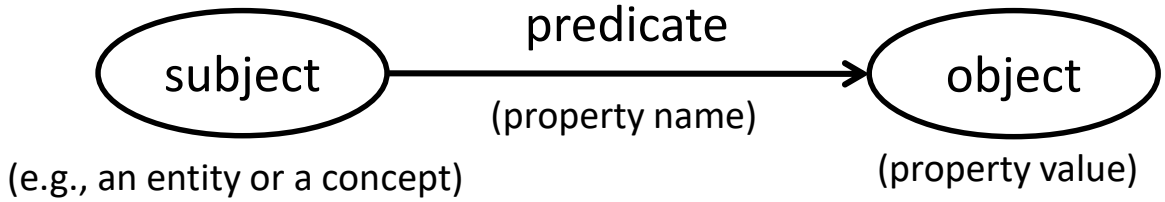
The state of the LOD cloud (2014):



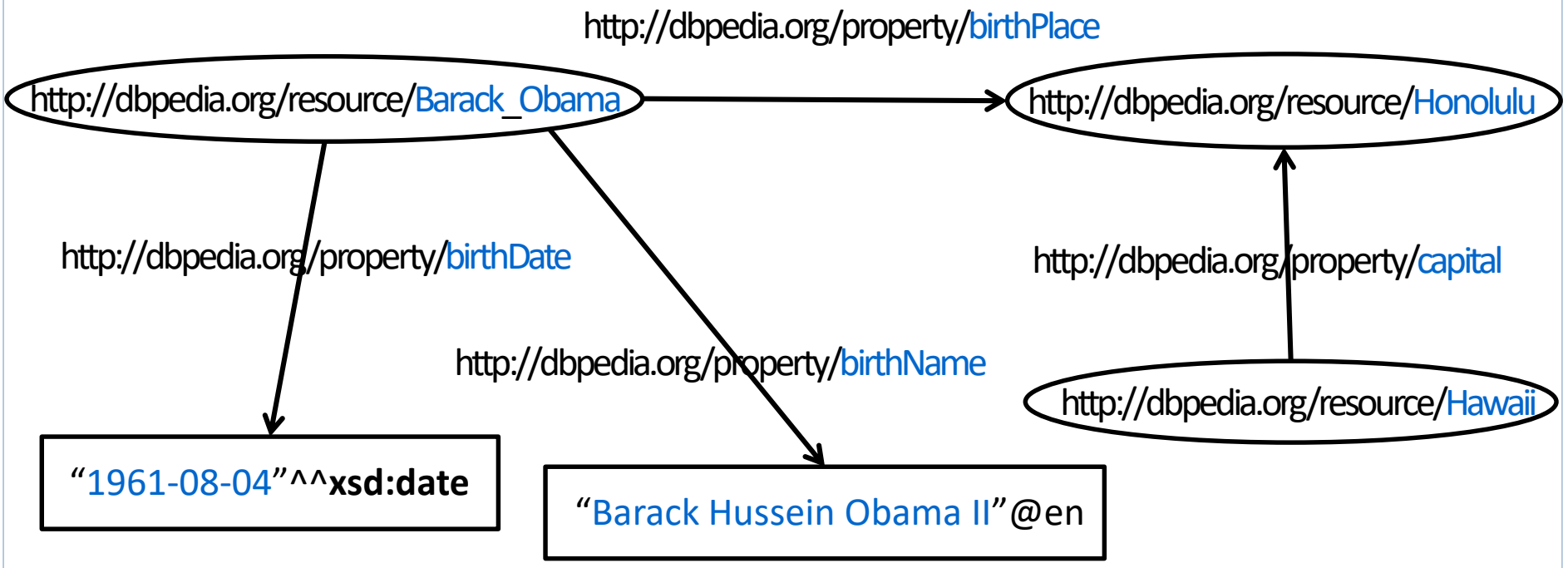
Interactive 3D Visualization of the LOD Cloud
http://www.ics.forth.gr/isl/**3DLod**

Main component: RDF (Resource Description Framework)

- Model for data
  - Syntax to allow exchange and use of information stored in various locations
  - Facilitate reading and correct use of information by computers
- RDF identifies resources with **URIs** (Uniform Resource Identifiers)
  - Often (though not always) the same as a URL
- RDF describes resources with **RDF triples**
  - Statements of the form SUBJECT – PREDICATE – OBJECT



RDF Graph

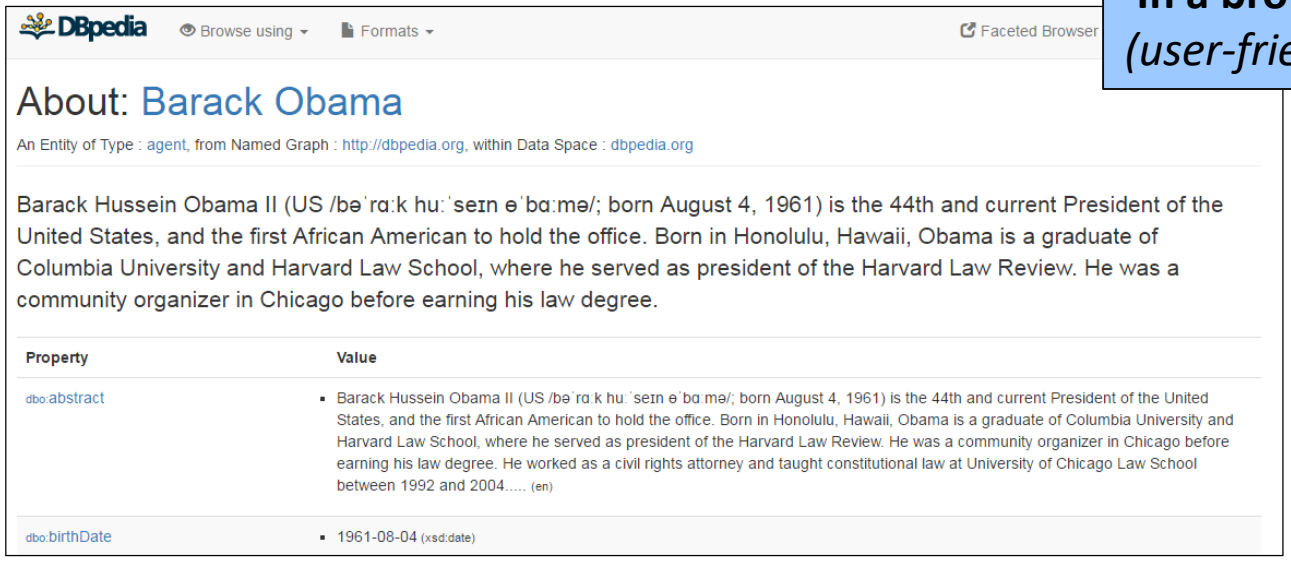


- Ontologies provide the "vocabulary" to describe data in RDF
- Linked Open Data (LOD): URIs should be dereferenceable (resolvable) and provide useful information in a standard format

Resolving a Web resource

http://dbpedia.org/resource/Barack_Obama

In a browser
(user-friendly)



Programmatically
(machine-friendly)

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dbpprop="http://dbpedia.org/property/"
  <rdf:Description rdf:about="http://dbpedia.org/resource/Barack_Obama">
    <dbpprop:birthDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">
        1961-08-04
    </dbpprop:birthDate>
    <dbpprop:birthPlace rdf:resource="http://dbpedia.org/resource/Honolulu" />
    <dbpprop:birthName xml:lang="en">
        Barack Hussein Obama II
```

RDF representation

- RDF/XML
- N-Triples
- Notation3 (N3) / Turtle
- JSON-LD
- Embedded in Web pages:
  - RDFa
  - JSON-LD, Turtle
  - Microdata, Microformats

RDF/XML

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dbpprop="http://dbpedia.org/property/"
  <rdf:Description rdf:about="http://dbpedia.org/resource/Barack_Obama">
    <dbpprop:birthDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">
        1961-08-04
    </dbpprop:birthDate>
    <dbpprop:birthPlace rdf:resource="http://dbpedia.org/resource/Honolulu" />
    <dbpprop:birthName xml:lang="en">
```

RDFa

```
<div typeof="foaf:Person" xmlns:foaf="http://xmlns.com/
foaf/0.1/">
   <p property="foaf:name">
     Alice Birpemswick
   </p>
   <p>
     Email: <a rel="foaf:mbox"
href="mailto:alice@example.com">alice@example.com</a>
   </p>
   <p>
     Phone: <a rel="foaf:phone" href="tel:
+1-617-555-7332">+1 617.555.7332</a>
   </p>
</div>
```

N-Triples

```
<http://dbpedia.org/resource/Barack_Obama>  <http://dbpedia.org/property/birthPlace>    <http://dbpedia.org/resource/Honolulu> .
<http://dbpedia.org/resource/Barack_Obama>  <http://dbpedia.org/property/birthDate> "1961-08-04"^^<http://www.w3.org/2001/XMLSchema#date> .
<http://dbpedia.org/resource/Barack_Obama>  <http://dbpedia.org/property/birthName> "Barack Hussein Obama II"@en .
<http://dbpedia.org/resource/Hawaii>    <http://dbpedia.org/property/capital>   <http://dbpedia.org/resource/Honolulu> .
```

Querying RDF data

- **SPARQL**
  - The standard language for querying RDF data

```
SELECT ?birthDate ?birthPlace
WHERE {
  <http://dbpedia.org/resource/Barack_Obama> dbo:birthDate ?birthDate .
  <http://dbpedia.org/resource/Barack_Obama> dbo:birthPlace ?birthPlace }
```

- **SPARQL endpoint**
  - Web protocol service enabling to query an RDF repository (triplestore) via SPARQL
  - Machine-friendly Web interface for querying a Knowledge Base
  - E.g., DBpedia's SPARQL endpoint: http://dbpedia.org/sparql

```
http://dbpedia.org/sparql?query=SELECT+%3FbirthDate … … …
```

Web of Data and Digital Libraries

- Great potential for Digital Libraries
  - Sharing knowledge on the Web | Dissemination
  - Information Integration and Enrichment
  - Support for complex information needs
  - Building relationships between DLs and external data sources

- CIDOC conceptual reference model [ISO 21127:2006]
- Europeana Data Model [Doerr et al., 2010] and Linked Open Data
- Bibliographic Framework Initiative (Library of Congress) [BIBFRAME]

- Adoption by global DLs
  - Library of France, Library of the Congress, British Library, National Library of Spain

Motivation

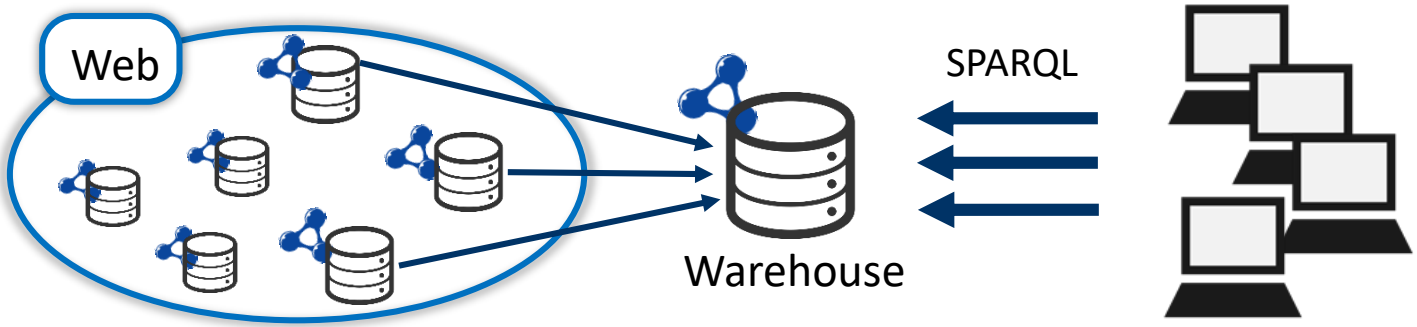
- Approaches on querying the Web of Data
- Limitations

Approaches on querying the Web of Data

- Data Centralization
  - Warehouse approach
- Link Traversal
  - “On the fly” data enrichment approach
- Query Federation
  - Mediator approach

Data Centralization (warehouse approach)

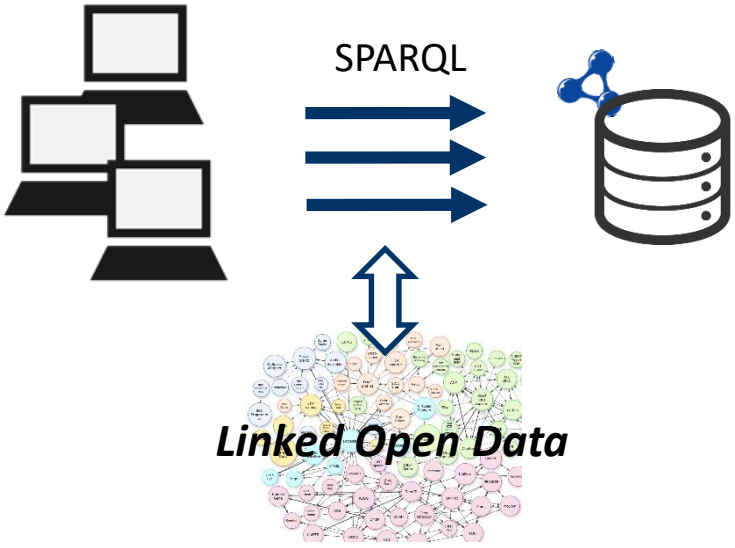
- Provide a query service over a collection of data
  - Copied (and probably transformed) from different sources on the Web
  - SPARQL endpoint over the warehouse



- Domain independent warehouses (e.g., SWSE [Hogan et al., Web Semantics 2011])
- Domain-specific warehouses (e.g., for the marine domain [Tzitzikas et. al, 2013])
- Digital Libraries (e.g., Europeana Linked Open Data)

Link Traversal

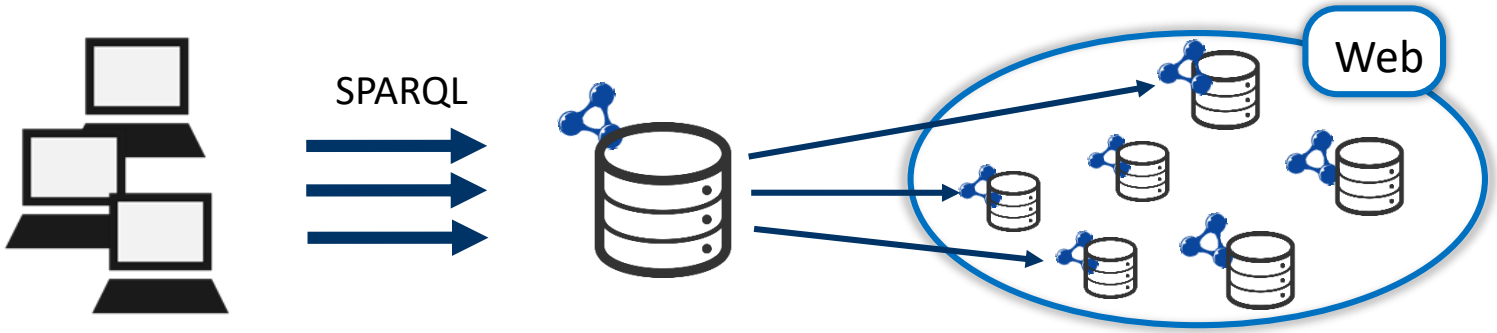
- Resolve URIs for discovering “on the fly” (at query-execution time) more data related to some resources



- Follow RDF links between resources based on URIs in the SPARQL query and in partial results
  - [Hartig, ISWC’12], Diamond [Miranker et al., AImWD’12]
  - LDQL [Hartig and Perez, ISWC’15]

Query Federation

- Provide integrated access to distributed sources on the Web



- Using a mediator service
  - DARQ [Quilitz and Leser, ESWC'08], SemWIQ [Langegger et al., ESWC'08]
- Directly through SPARQL
  - FROM/FROM NAMED and GRAPH operators
  - SPARQL 1.1 Federated Query (SERVICE operator)

Limitations – FROM and GRAPH

- **FROM/FROM NAMED** and **GRAPH** operators
  - It requires knowing in advance (during query formulation) the URIs of the remote resources (and declare them at FROM/FROM NAMED)
- The majority of SPARQL implementations uses FROM/FROM NAMED for querying specific named graphs already loaded in the local repository
  - They cannot retrieve and query a remote-dataset at query-execution time

How to query remote resources coming from partial results (derived at query-execution time)?

```
SELECT DISTINCT ?creator ?photo
FROM NAMED <?????>
WHERE {
  SERVICE <http://europeana.ontotext.com/sparql> {
    ?work dc:subject dbr:Renaissance ; dc:creator ?creator }
  GRAPH ?creator {
    ?creator foaf:depiction ?photo } }
```

Limitations - SERVICE

- SERVICE operator
  - We can invoke a portion of a query against a remote RDF repository
  - The URI should be the address of a SPARQL endpoint

We cannot query RDF data accessible on the Web but not available through an endpoint

```
SELECT DISTINCT ?creator ?photo WHERE {
  SERVICE <http://dbpedia.org/sparql> {
     ?creator dc:subject dbc:Mannerist_painters }
  ?creator foaf:depiction ?photo }
```

Limitations – Markup

- Markup languages are exploited by an ever-increasing number of publishers
  - **RDFa** [W3C Recom. 2015]
  - Embedded **JSON-LD** [W3C Recom. 2014] and **Turtle** [W3C Recom. 2014]
- The majority of SPARQL implementations do not support querying such RDF data!

- Web sites supporting RDFa
  - Yahoo.com
  - Hotels.com
  - ifood.tv
  - Food.com
  - Cnet.com
  - staples.com
  - nbcnews.com
  - Expedia.com
  - …

How to query such markup data directly through SPARQL?

Limitations - Reliability

- Reliability of public SPARQL endpoints
  - The major bottleneck towards the success of the “Semantic Web” realization
  - [Buil-Aranda et al., ISWC’13]
    - Only 32.2% of public endpoints have monthly **uptimes** of >99%
    - Their **performance** can vary by up to 3-4 orders of magnitude

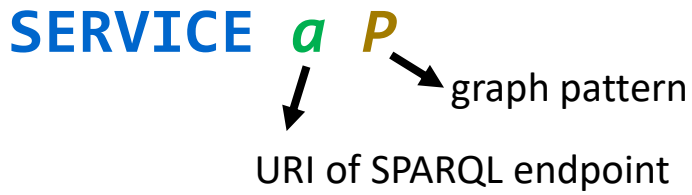
- Can we publish our data and **make them queryable through SPARQL** without needing to set up and maintain a (costly) SPARQL endpoint?

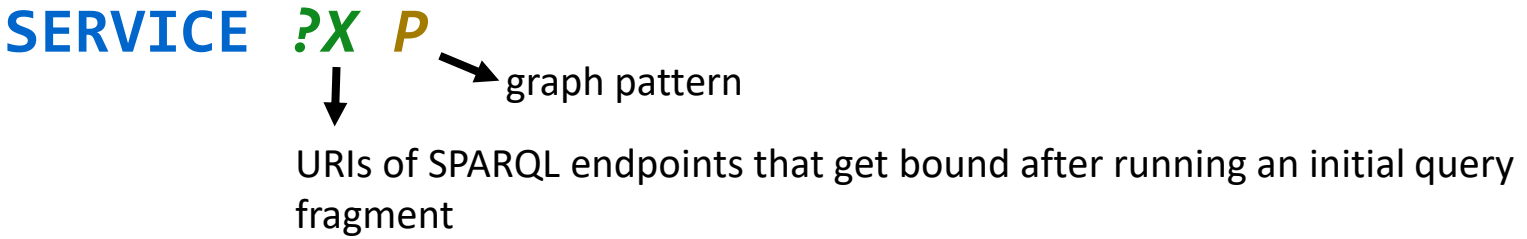
SPARQL-LD (**L**inked **D**ata)

- Extended SERVICE operator
- Implementation
- Examples
- Optimizations

Extended SERVICE operator

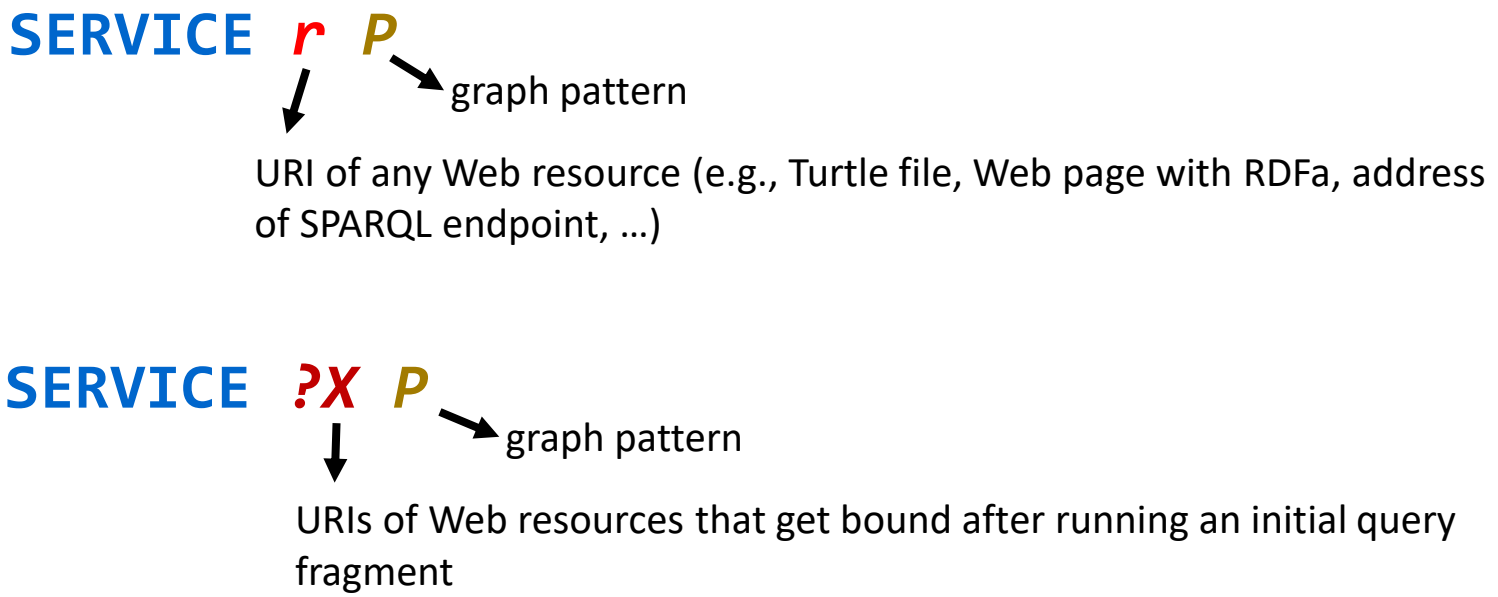
- Original SERVICE operator of SPARQL 1.1 Federated Query:



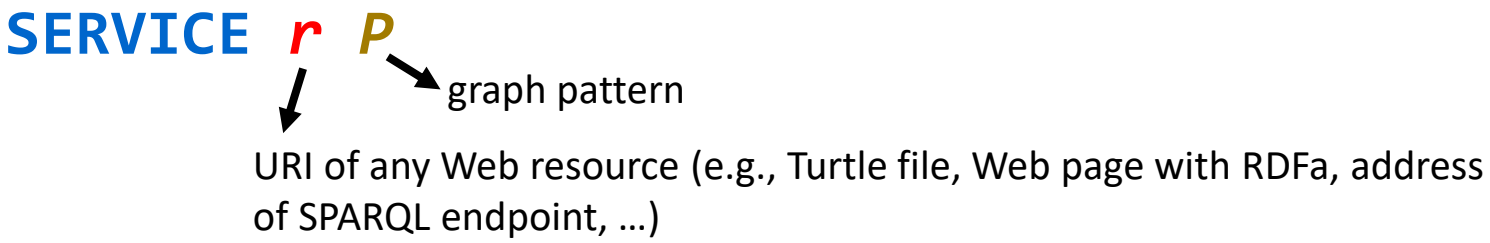


Extended SERVICE operator

- Extended SERVICE operator:



Extended SERVICE operator

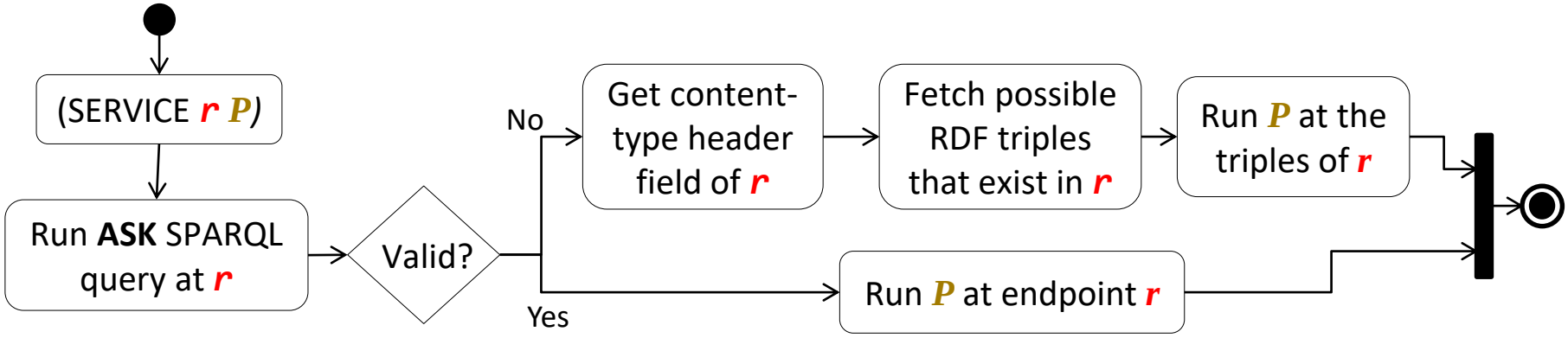


- If ***r*** is the address of a SPARQL endpoint:
  - Same as the original service operator: the remote endpoint evaluates the graph pattern ***P*** and returns the results
- Otherwise:
  - The RDF data that may exist in the Web resource ***r*** are fetched at real-time and queried for the graph pattern ***P***
  - If no RDF data exist in ***r***, no bindings are returned

Implementation

- Apache Jena Extension
  - Extension of Jena 2.13 ARQ component
  - Available on GitHub (+URLs of endpoints that implement SPARQL-LD)
    - **https://github.com/fafalios/sparql-ld**

- The query execution process



Query Examples

- Query:
  - Data embedded in a Web page as **RDFa**
  - Data from dereferenceable URIs (derived at query-execution time!)

The query returns all my co-authors together with their publications.

```
SELECT DISTINCT ?authorName ?paper WHERE {
    SERVICE <http://users.ics.forth.gr/~fafalios> {
        ?p <http://purl.org/dc/terms/creator> ?authorURI }
    SERVICE ?authorURI {
        ?authorURI foaf:name ?authorName .
        ?paper dc:creator ?authorURI } }
```

Query Examples

- Parameterize and call a named-entity recognition **Web Service** at query-execution time

The query first retrieves Web pages related to the fish genus Thunnus (using its dereferenceable URI), and then it calls a named-entity recognition service (X-Link) for identifying (at request time) names of fishes and countries in these Web pages.

```
SELECT DISTINCT ?detectedEntity ?categoryName (count(?position) as ?NumOfOccurrences)
WHERE {
  SERVICE <http://dbpedia.org/resource/Thunnus> {
    dbpedia:Thunnus dbpedia-owl:wikiPageExternalLink ?page }

  VALUES ?templ {
     <http://83.212.107.202/x-link-marine/api?categories=fish;country&url=PAGE> }
  BIND(REPLACE(str(?templ), 'PAGE', str(?page), 'i') as ?x) BIND(URI(?x) as ?service)
  SERVICE ?service {
    ?annot oa:hasBody ?ent .
    ?ent oae:regardsEntityName ?detectedEntity ;
       oae:position ?position ;
       oae:belongsTo ?category . ?category rdfs:label ?categoryName }
} GROUP BY ?detectedEntity ?categoryName ORDER BY DESC(?NumOfOccurrences)
```

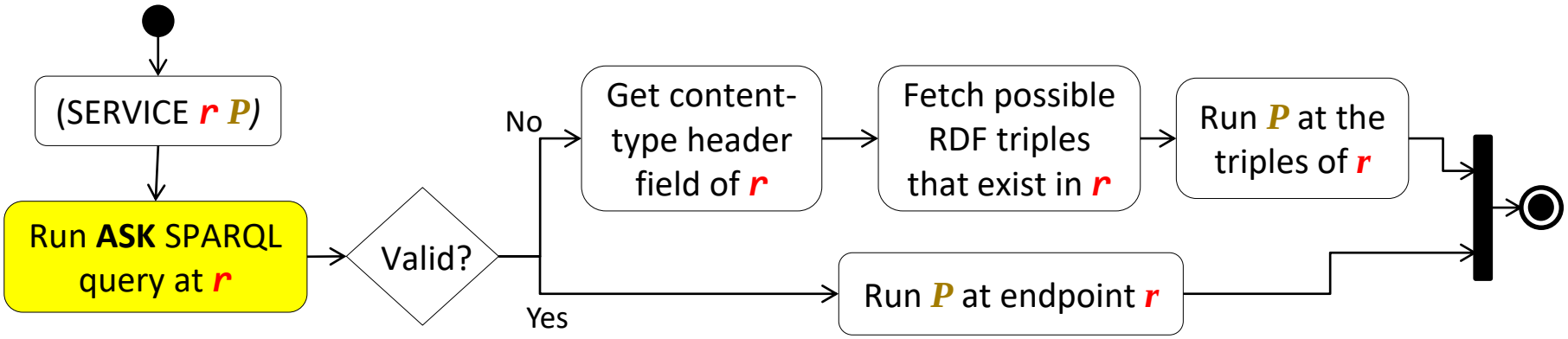
Optimizations

- Existing approaches
  - Optimizing the execution of SPARQL federated queries
    - Reordering triple patterns [Schwarte et al, ISWC'11]
    - Planning SERVICE queries [Montoya et al., COLD'12]
  - Caching
    - Improving the performance of SPARQL queries [Kjernsmo, ESWC'15]

- All existing approaches are also beneficial for SPARLQ-LD
- Extra points that need attention:
  - Reduce ASK queries
  - Avoid multiple fetching of same remote resource

Optimizations

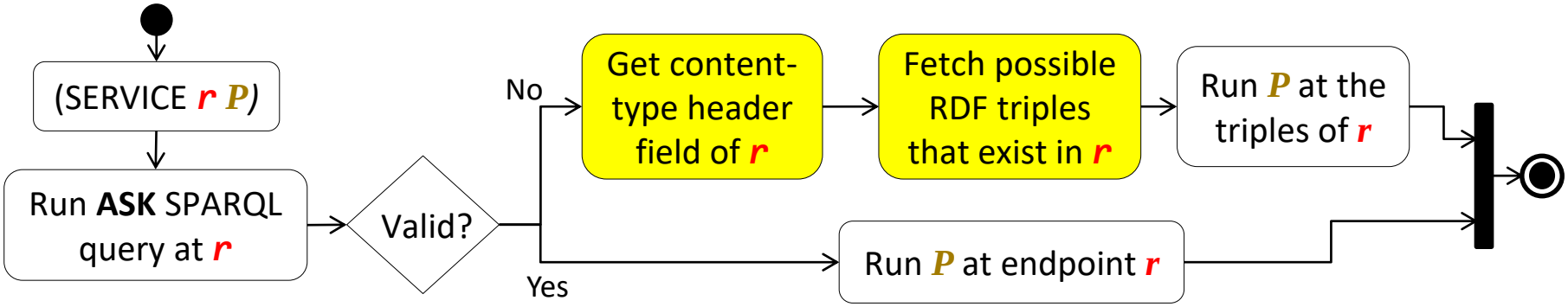
- Index of Known SPARQL endpoints
  - Small index with the URIs of “known” endpoints
  - Avoid running ASK queries to known or already-checked endpoints




```
SELECT DISTINCT ?painter ?work  WHERE {
  SERVICE <http://dbpedia.org/resource/Category:Greek_painters> {
    ?painter <http://purl.org/dc/terms/subject> ?greekPainter }
  SERVICE <http://europeana.ontotext.com/sparql> {
    ?objectInfo <http://purl.org/dc/elements/1.1/creator> ?painter .
    ?objectInfo <http://www.openarchives.org/ore/terms/proxyFor> ?work } }
```

Optimizations

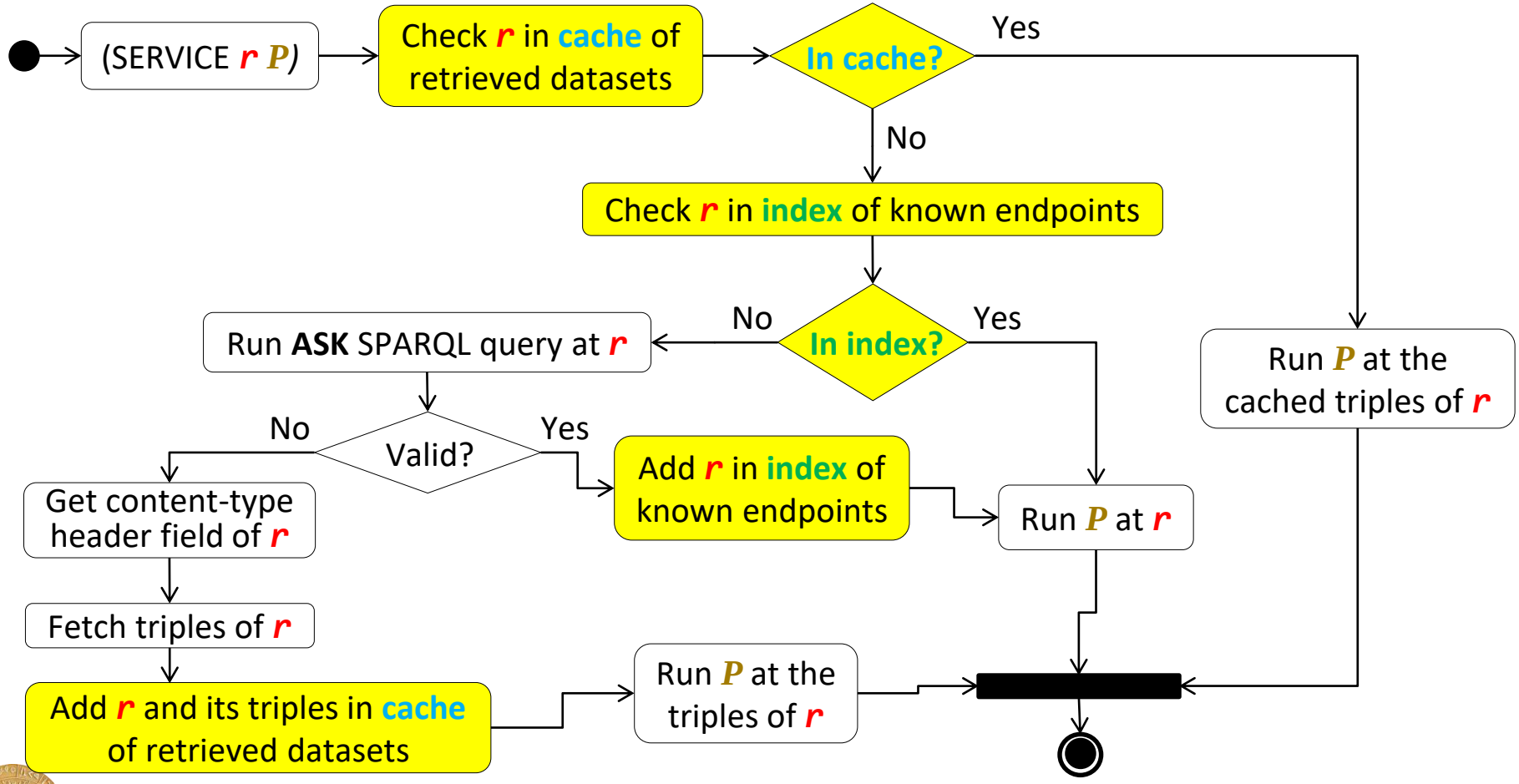
- Request-scope caching of fetched datasets
  - A query may contain multiple SERVICE invocations against the same Web resource
  - Avoid fetching remote resources that have been already fetched (in the context of a single query execution)




```
SELECT DISTINCT ?authorName ?paper WHERE {
  SERVICE <http://users.ics.forth.gr/~fafalios/> {
    ?p <http://purl.org/dc/terms/creator> ?author }
  SERVICE ?author {
    ?author <http://xmlns.com/foaf/0.1/name> ?authorName .
    ?paper <http://purl.org/dc/elements/1.1/creator> ?author } }
```

Optimizations

The optimized query execution process

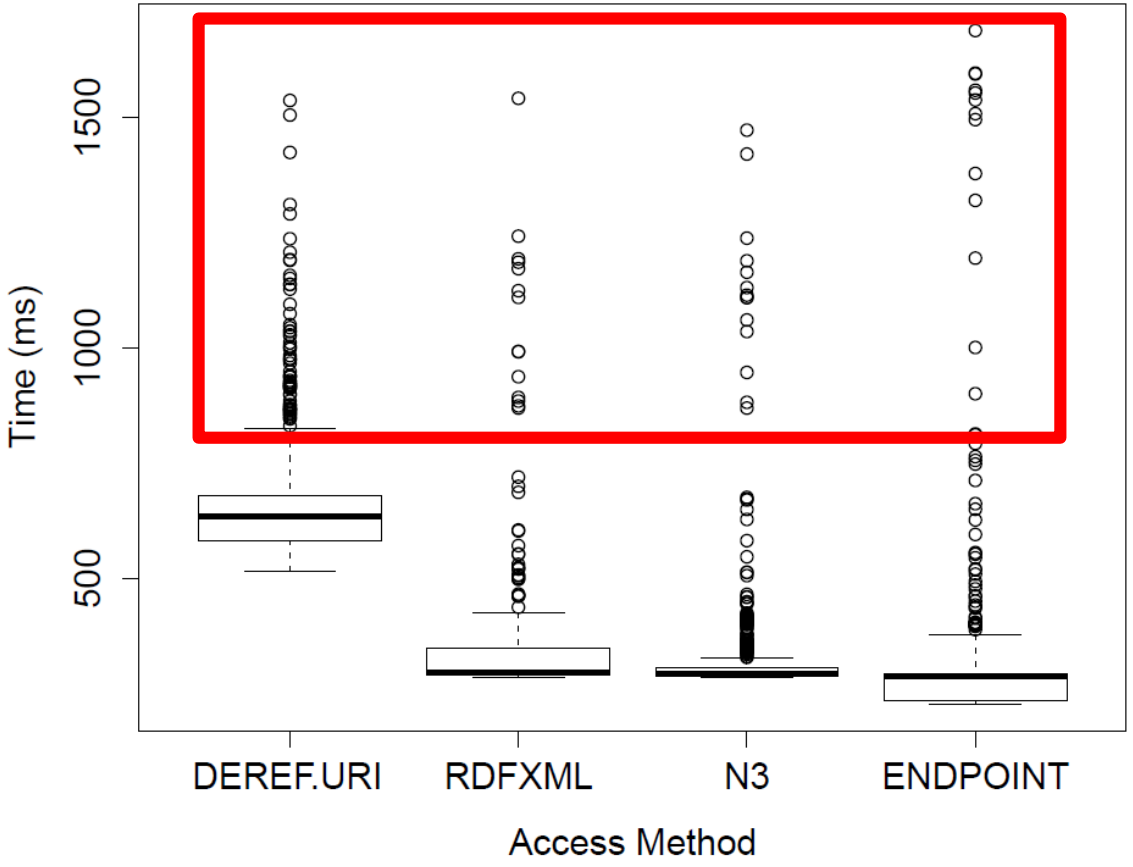


Evaluation

- Query execution time
- Accessing very large Web resources
- Effect of optimizations

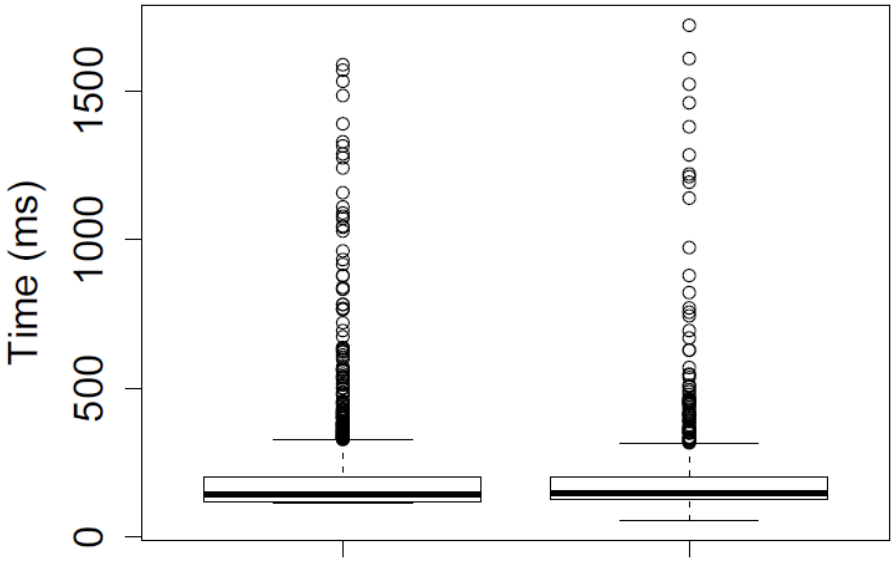
Query execution time

- Experiments for 1,000 randomly selected DBpedia URIs
- Time for retrieving the outgoing properties of each URI
- Using different access methods:
  - Dereferenceable URI
  - RDF/XML
  - Notation3 (N3)
  - SPARQL endpoint



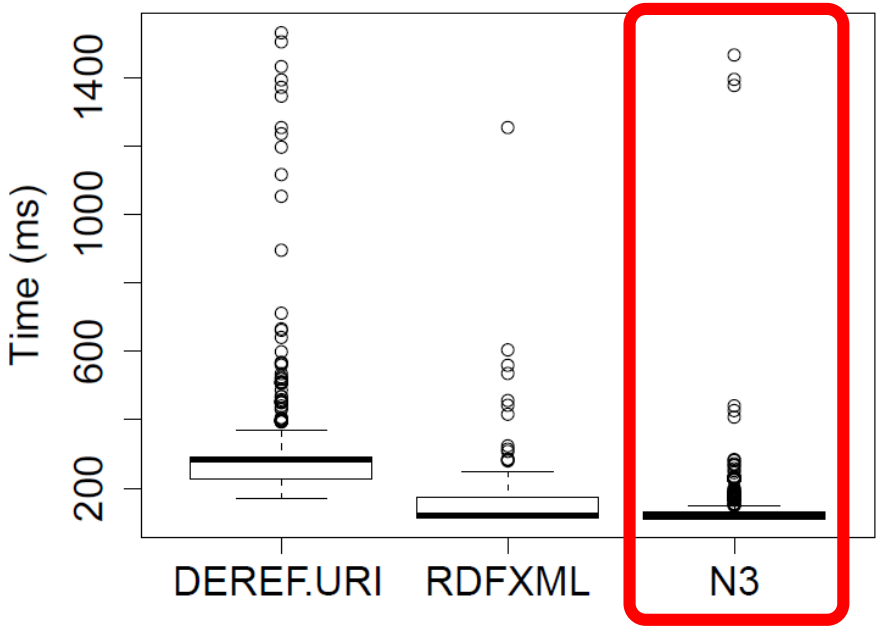
Query execution time

- Time required by the main subtasks of query execution



Checking if URI is an endpoint

Checking the URI content type



Fetching and loading the RDF data of the remote resource

Accessing very large Web resources

- N3 files of different size
  - 10,000 triples
  - 100,000 triples
  - 1,000,000 triples
  - 10,000,000 triples
- Run a query that requests the properties of a particular resource (existing in all files as subject in the triple)

Num of triples	Total Query Time	Is endpoint	Get content type	Fetch triples
10^4	900 ms	318 ms	7 ms	449 ms
10^5	3.2 sec	1,3 sec	8 ms	1.8 sec
10^6	31 sec	11 sec	8 ms	19 sec
10^7	546 sec	111 sec	51 ms	433 sec

Effect of optimizations

- Index of Known Endpoints
  - Experiments for different number of SERVICE calls to already-checked endpoints

Query	Num of calls to indexed endpoints	Time without Opt.	Time with Opt.	Speedup
Q1	10	3.5 sec	1.8 sec	1.9×
Q2	10^2	27.2 sec	16.5 sec	1.6×
Q3	10^3	9.6 min	2.5 min	3.9×
Q4	10^4	44.8 min	24 min	1.9×

Effect of optimizations

- Caching of Fetched Datasets
  - Experiments for:
    - Different number of triples in the remote resources
    - Different number of SERVICE calls to already-fetched resources

Query	Num of calls to cached datasets	Num of triples	Time without Opt.	Time with Opt.	Speedup
Q5	**16**	10^3	11.9 sec	1.4 sec	8.5×
Q6	**16**	10^4	72.9 sec	5.7 sec	12.7×
Q7	**16**	10^5	10.3 min	39.8 sec	15.5×
Q8	10	$\mathbf{10^2}$	11.8 sec	9.6 sec	1.2×
Q9	10^2	$\mathbf{10^2}$	35.9 sec	10.7 sec	3.4×
Q10	10^3	$\mathbf{10^2}$	4.8 min	11.6 sec	24.6×

Conclusion

- **SPARQL-LD**: a **generalization** of SPARQL 1.1's **SERVICE** operator
  - Fetch and query RDF data from any HTTP Web source (directly through SPARQL)
  - Query remote resources whose URIs are derived at query-execution time!
- **Integrate** (at query-execution time) data coming from heterogeneous sources:
  - local repository, other endpoints, dereferenceable URIs,
  - online RDF/XML, N3, Turtle, JSON-LD files
  - RDFa, embedded JSON-LD and Turtle
  - Data dynamically-created by Web Services
- **Motivate** Web publishers to enrich their digital contents and services with RDF!
  - Their data is made directly accessible and exploitable via SPARQL!
  - No need to set up and maintain a (costly) SPARQL endpoint
- A step towards the **Semantic Web** realization!

Conclusion

- Query-**execution time** highly depends on:
  - Number of triples existing in the resource
  - Status of the network between local and remote server
  - Status of remote server
- For "common" resources (<10^5 triples), total query time is very low

- Simple **optimizations** can highly reduce the query-execution time
  - Index of known and already-checked SPARQL endpoints
  - Request-scope caching of fetched triples

Future Work

- More **optimization** techniques
  - Query Planning
  - Caching

- **Query re-writing**
  - Queries to remote endpoints → queries to remote resources
  - Avoid querying the (often unreliable) endpoints

Thank you

demo and more:

https://**github**.com/fafalios/**sparql-ld**

Horizon 2020
European Union funding
for Research & Innovation

Foundation for Research and Technology – Hellas (FORTH)
Institute of Computer Science (ICS)
Information Systems Laboratory (ISL)

University of Crete
Computer Science Department
Greece