SPARQL Protocol And RDF Query Language
WS 2011/12: XML Technologies

John Julian Carstens

Department of Computer Science
Communication Systems Group
Christian-Albrechts-Universität zu Kiel

March 1, 2012
Overview

1. Introduction
2. Linked Open Data
3. RDF
4. SPARQL
5. Usage and Examples
6. Conclusion
Overview

1. Introduction
2. Linked Open Data
3. RDF
4. SPARQL
5. Usage and Examples
6. Conclusion
Motivation

- Web is largely made for people, not machines
- Importance of machine-to-machine communication shown by wide use of web services
- Information retrieval is very difficult for a machine if data is unstructured
- When data is structured, a language for querying the data for certain information is necessary
How to use Structured Data?

1. Example, Semantic phone voice control:
   "Call my girlfriend!"
   1. Query: "x = FIND girlfriend OF user";
   2. Query: "FIND phone_no OF x"

2. Example, Information retrieval, e.g. from Wikipedia:
   "Show me the people born in Kiel in 1950!"
   Query: "FIND people WHERE place_of_birth = "Kiel" AND year_of_birth = 1950"
Goals

- Collect information about everything
- Combine and connect information to see hidden or unobvious relations
- *Standardizing* information storage and the way to access them
- Many more ...

Introduction
Linked Open Data
RDF
SPARQL
Usage and Examples
Conclusion
Overview

1. Introduction
2. Linked Open Data
3. RDF
4. SPARQL
5. Usage and Examples
6. Conclusion
What is Semantic Web?

- "New" concept for further developing the world wide web
- Problem: Web-media like HTML-pages, videos, songs etc. are designed for human consumption
- Solution: Connect resources with information, so that a machine can process it
- Especially information about resources → metadata
- Advantages for several use cases: consumers, scientists, health, search engines etc.
What is Semantic Web?

**Figure: Human view**¹

**Figure: Machine view**

Linked Open Data

- Basic idea for machine readable web: Connect resources to give them meaning ⇒ Linked Data
- On the web, readable by everyone ⇒ Linked Open Data

Thus, we need:

1. Language for connecting data and resources
2. Language for retrieving information from Linked Data
Linked Open Data Cloud

Image by Anja Jentzsch in http://en.wikipedia.org/wiki/Linked_data
1 Introduction

2 Linked Open Data

3 RDF

4 SPARQL

5 Usage and Examples

6 Conclusion
Reminder

We needed:

1. Language for connecting data and resources
2. Language for retrieving information from Linked Data

To solve 1, the Resource Description Framework (RDF) was introduced
RDF Basics

- Recommendation by the W3C
- XML-based framework to describe and connect resources
- Every describable thing is a *resource*, even things outside the web
- Resources have *properties* which have *values*
- Unique reference to each resource, property or value by URI
The basic view on RDF data is triples.

Each information is coded as triple, having:

- **Resource**, the resource to be described (also called subject)
- **Property**, a property of the resource (also called predicate)
- **Value**, the value of the property (also called object)

Use "blank nodes" to group properties and values for one resource.

Resource, Property and Value can be something referred to by a URI, Value can also be a literal.
RDF Syntax

- Different syntaxes are given for RDF: XML and triple
- Data can also be regarded as a graph, introducing RDF graphs

Example:
Car8231 Type Mustang68 .
Mustang68 Manufacturer Ford .
Car8231 LicensePlate "Lic42" .
Ford Founder Henry_Ford
Example (cont’d):

```xml
<Car id="8231">
  <Type name="Mustang68">
    <Manufacturer name="Ford">
      <Founder>Henry Ford</Founder>
    </Manufacturer>
  </Type>
  <LicensePlate>Lic42</LicensePlate>
</Car>
```

**Figure:** XML representation

**Figure:** Graph representation
Notable RDF Extensions/Vocabulary

- FOAF: Friend of a friend, describing persons, their activities and relations to people and objects
- DC: Dublin core, vocabulary terms that can be used to describe documents by metadata
- DBO: DBpedia ontology, vocabulary to describe structured data of Wikipedia articles
Overview

1. Introduction
2. Linked Open Data
3. RDF
4. SPARQL
5. Usage and Examples
6. Conclusion
Reminder

We needed:

1. Language for connecting data and resources ✓

2. Language for retrieving information from Linked Data

To solve 2, we need a query language querying one representation of RDF
Which Representation of RDF to Query?

- XML-representation ⇒ X-Path, other XML query languages
- Triple representation ⇒ Fulltext, triple query?
- Graph representation ⇒ Pattern matching?

**SPARQL** ⇒ matching patterns against the RDF graph

Other RDF query languages: RDQL, ICS-FORTH RQL, SeRQL (not covered here)
What is a Graph Pattern?

- Graph pattern is constructed out of *triple patterns*
- Triple pattern is a RDF triple where *variables* can be used in resource, property or value

**Example:**

<http://example.com/cars/mustang68>
<http://purl.org/goodrelations/v1#hasManufacturer>
?name
What is a Graph Pattern?

- A single triple pattern is already a graph pattern
- Create complex graph patterns by using *group patterns*
- Many creation possibilities
Basic Graph Patterns (BGP)

- BGP is a sequence of triple patterns
- Can be regarded as the conjunction of the single patterns
- Write a full stop after each single pattern
- Can use namespace prefix like in XML

Example:

```
cars:mustang68 gr:hasManufacturer ?name .
?name foaf:page ?url
```
A group graph pattern is a set of graph patterns

Group pattern is indicated by surrounding brackets {} and can be empty

Patterns within a group are implicitly conjuncted, there is no AND keyword in SPARQL

Example:

\{cars:mustang68 gr:hasManufacturer ?name .
\} ⇔

\{{cars:mustang68 gr:hasManufacturer ?name .}
{cars:mustang68 foaf:page ?url .} \}
Filters

- Give filter expression to restrict values of a variable
- Filter expression is evaluated, when returns \texttt{true} ⇒ Value is accepted
- Use arithmetic filters or string filters (with regular expression evaluation based on XPath)
- Can use multiple filters in group graph pattern

Examples:

\texttt{FILTER (?model_year = 1968)}
\texttt{FILTER regex(?name, "Ford")}
Optional

- Allows to include values in the result when available, but doesn’t deny result when value is not available
- When value is not available, the query variable is not set
- Useful, when it is not sure whether RDF information is complete

Example:

```
OPTIONAL {cars:mustang68 cars:license_plate ?license_plate }
```
Returns result when at least one alternative matches
Returns all matching result
Similar to SQL UNION operation, thus keyword UNION was chosen

Example:
{
{car? gr:hasManufacturer "Ford" . } UNION
{car? gr:hasManufacturer "GM" . } }

Queries

- Queries are constructed by using the patterns above and some surrounding syntax
- Basic queries are using SELECT and WHERE, similar to SQL
- Reply depends on content of SELECT, e.g. SELECT ?x
- SELECT * is also possible
- Results are returned as bindings to the variables in SELECT

Example:
SELECT ?name
WHERE {cars:mustang68 gr:hasManufacturer ?name .}
Above: Namespaces "assumed" due to simplicity
Include namespace prefix in query by using the PREFIX keyword

Example:

*Give the manufacturer(s) of certain car.*

```
PREFIX cars: <http://example.com/cars>
PREFIX gr: <http://purl.org/goodrelations/v1>
SELECT ?name
WHERE {cars:mustang68 gr:hasManufacturer ?name . }
```
Queries: Blank Nodes

- When a certain value is not of interest for the result: Use blank node
- Write either _:<label> or []
- Is defined in a BGP, refers the same value if used more than once
- May not use the same blank node in two different BGPs
- Can also express this by using a variable and not including it in SELECT
Blank nodes examples:

*Give all manufacturers of anything.*

```sparql
PREFIX gr: <http://purl.org/goodrelations/v1>
SELECT ?name
WHERE {[] gr:hasManufacturer ?name . }
```

*Give all types of goods and their manufacturers.*

```sparql
PREFIX gr: <http://purl.org/goodrelations/v1>
SELECT ?name ?type
WHERE {_:v gr:hasManufacturer ?name .
:_:v gr:typeOfGood ?type }
```
Queries: CONSTRUCT

- CONSTRUCT keyword returns a new RDF graph instead of a variable binding
- Useful for preparing information for further processing or transforming a data set

Example:

Create RDF graph describing portfolio of manufacturers.

```sparql
PREFIX gr: <http://purl.org/goodrelations/v1>
CONSTRUCT {?name gr:offers ?car}
WHERE {?car gr:hasManufacturer ?name . }
```
Queries: ASK

- Use keyword ASK to check, whether a graph pattern matches
- Returns only yes or no

Example:

*Is there a car manufactured by Ford?*

```sparql
PREFIX gr: <http://purl.org/goodrelations/v1>
ASK { ?car gr:hasManufacturer "Ford" }
```
Queries: ORDER BY

- Can sort results similar to SQL by using ORDER BY keyword
- Can sort using multiple columns and is descending DESC() or ascending ASC()

Example:

Give all manufacturers, ordered by manufacturers name.

PREFIX gr: <http://purl.org/goodrelations/v1>
SELECT ?name
WHERE {?car gr:hasManufacturer ?name }
ORDER BY DESC(?name)
Queries: DISTINCT and REDUCED

- To exclude duplicates from result, DISTINCT keyword can be added to query
- REDUCED allows to exclude any number of duplicates from the result, e.g. if the result contains the same value thrice, REDUCED may return it once, twice or thrice again (the standard does not specify how this is decided)

**Example:**

*Give all manufacturers without duplicates.*

```
PREFIX gr: <http://purl.org/goodrelations/v1>
SELECT DISTINCT(or REDUCED) ?name
WHERE {?car gr:hasManufacturer ?name }
```
Queries: LIMIT and OFFSET

- LIMIT creates an upper bound for the number of solutions in the result
- OFFSET will skip the first $n$ solutions of the result
- Only useful when kept predictable by using ORDER BY

**Example:**

*Give manufacturers 20 to 30, ordered by manufacturers name.*

```
PREFIX gr: <http://purl.org/goodrelations/v1>
SELECT ?name
WHERE {?car gr:hasManufacturer ?name }
ORDER BY DESC(?name)
LIMIT 10
OFFSET 20
```
Overview

1. Introduction
2. Linked Open Data
3. RDF
4. SPARQL
5. Usage and Examples
6. Conclusion
Reminder

We needed:

1. Language for connecting data and resources ✓
2. Language for retrieving information from Linked Data ✓

Thus, we have now a representation for our data and a language to ask questions. What to do with that now?
Usage examples

- Use RDF to store any data schema-less and SPARQL to query
  ⇒ Similar usecase like SQL, e.g. AllegroGraph
- Use knowledge which is already collected and stored in RDF
  ⇒ Query SPARQL-Endpoints
- Machines can collect and present data with less human support
SPARQL Endpoint

- SPARQL accepting web service
- Allows clients (humans and machines) to query a knowledge base via SPARQL
- Returns data in formats explained above ⇒ To be used by experts or machines (or to be enhanced for end-users)
- Many large ontologies from Linked Open Data offer endpoints, e.g. BioGateway, data.gov or DBpedia
DBpedia

- Project to extract structured content from the information collected in Wikipedia
- Part of Linked Open Data
- Started by Free University of Berlin and University of Leipzig
- Data is stored using RDF
- As of September 2011, DBpedia contains 1 billion of RDF triples
- Commonly referred when choosing URI for real world resources
- Provides open SPARQL endpoint
Querying DBpedia

- Use SPARQL endpoint at http://dbpedia-live.openlinksw.com/sparql
- Use DBpedia ontologies <http://dbpedia.org/ontology/> and <http://dbpedia.org/resource/>

Example queries:

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
SELECT ?name
WHERE { res:Ford_Mustang dbo:manufacturer ?name}

CONSTRUCT {res:Ford_Motor_Company dbo:product ?car}
WHERE { ?car dbo:manufacturer res:Ford_Motor_Company}
```
RelFinder

Question: How are any two things connected with each other?
- Dynamic graphical visualization
- Bases on DBpedia dataset
- Uses SPARQL to find connections

Example:
Let’s try to find out how Harry S. Truman and Dwight D. Eisenhower are connected!
RelFinder

Image created with http://www.visualdataweb.org/relfinder.php
RelFinder: How it works

- Accesses database, e.g. DBpedia, via SPARQL endpoint
- Poses queries with growing complexity, patterns have the two subjects to be compared as leftmost/rightmost object

Sample query:
```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
SELECT ?conn ?obj ?conn2
  res:Dwight_D._Eisenhower ?conn2 ?obj }
```
# Overview

1. **Introduction**
2. **Linked Open Data**
3. **RDF**
4. **SPARQL**
5. **Usage and Examples**
6. **Conclusion**
Problems

- Very complex language, semantics not fully defined (ambiguities, gaps)
- Knowledge bases tend to become very large (remember the Semantic Web map) $\Rightarrow$ Runtime problems
- Problem of graph pattern matching is NP-complete
- Triple model is very different to classic models like relational databases
- Rather small (research) community
Conclusion

- Versatile framework for knowledge and data store (RDF)
- Query language for navigating in data store
- Several real world applications, e.g. Semantic Web, triple store (AllegroGraph), DBpedia etc.
- Yet many goals to approach, e.g. semantic voice/natural language processing
- Not well known or commonly used


