Introduction to SPARQL query language

Rouzbeh Meymandpour and Fethi Rabhi

CAPSICUM Business Architects & UNSW
Creative Commons Attribution-ShareAlike 4.0 International
Review RDF Graphs – RDF Statements (Triples)

Subject ➔ Predicate ➔ Object

Vertebra ➔ has ➔ Cat ➔ has ➔ Fur

Animal ➔ is an ➔ Mammal ➔ is a ➔ Bear

Fish ➔ lives in ➔ Water ➔ lives in ➔ Whale

Cat ➔ has ➔ Fur

Whale ➔ lives in ➔ Water

Bear ➔ has ➔ Fur

Vertebra ➔ has ➔ Cat
**Nodes:** Subjects and Objects

- **Resource nodes:** A resource is a *thing* that can have *things* said about it.
  
  Represented by ovals.
  
  Identified by a Unique Resource Identifier (URI)

- **Literal nodes:** Literal means *value*.
  
  Represented by rectangles

**Edges:** Predicates (aka Properties)

- From a Resource to another Resource (aka Relations)
- From a Resource to a Literal (aka Attributes)
- Identified by a Unique Resource Identifier (URI)
• **RDF/XML**: RDF represented as XML. RDF/XML is verbose. Difficult to read and write as a human.

• **N-Triples**: One triple per line.

• **Turtle**: More compact than RDF/XML, more readable than N-Triples. Syntax of SPARQL queries.
Review Turtle Examples

• Turtle

@prefix dbp: <http://dbpedia.org/property/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
dbr:Leonard_Cohen
    dbo:birthPlace dbr:Montreal ;
    dbp:dateOfBirth "1934-09-21"^^xsd:date .

**SPARQL: Protocol + Query Language**

- **SPARQL: Protocol:**
  - Interactions between a SPARQL engine (endpoint) and a client via HTTP.

- **SPARQL: RDF Query Language:**
  - Based on RDF Graph matching
  - Six forms:
    - **SELECT:** The most common query form that returns raw results,
    - **Update:** DELETE and INSERT
    - **CONSTRUCT:** Returns the results as a new RDF graph
    - **ASK:** Returns a Boolean (True/False) result based on the query
    - **DESCRIBE:** Returns a valid RDF graph describing a resource (where the resource is a subject and/or object, varies based on the engine)
Querying RDF with SPARQL

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License from Introduction to the Semantic Web By Dean Allemang
“Find all electronic publications that are related to a book”
“Find all electronic publications that are related to a book”

Electronic Publication

Book

?ep

skos:related

?book

How many triples?
SELECT ?ep ?book
WHERE {
}

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Adapted from Introduction to the Semantic Web By Dean Allemang
SPARQL Syntax: Sections

- **PREFIX** *(optional)*
  - Prefix declarations for abbreviating URIs
- **SELECT** *(the most popular of the six forms)*
  - Returns the results
- **FROM** *(optional)*
  - Defines the RDF graph that is being queried
- **WHERE**
  - Specifies the query graph (conditions) to be matched
- **Solution Modifiers:** ORDER BY, LIMIT, OFFSET, GROUP BY, HAVING
@prefix dbp: <http://dbpedia.org/property/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

SELECT ?birthPlace
WHERE {
  ?birthPlace a dbo:Place .
  ?birthPlace a dbo:Country .
}

Note: a is the same as rdf:type
SPARQL Syntax:

```
@prefix dbp: <http://dbpedia.org/property/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

SELECT ?birthPlace
WHERE {
    ?birthPlace a dbo:Place ;
        a dbo:Country .
}
```
@prefix dbp: <http://dbpedia.org/property/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

SELECT ?birthPlace
WHERE {
  ?birthPlace a dbo:Place, dbo:Country .
}
All Canadian songwriters and their instrument
(including those without any instrument)

@prefix dbp: <http://dbpedia.org/property/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

SELECT ?songwriter ?instrument
WHERE {
    ?songwriter a dbo:Singer-Songwriter ;
        dbo:birthPlace dbr:Canada ;

    OPTIONAL {
    }
}

All instruments used by Canadian musicians

@prefix dbp: <http://dbpedia.org/property/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
SELECT DISTINCT ?instrument
WHERE {
    ?songwriter a dbo:Musician ;
    dbo:birthPlace dbr:Canada ;
    dbo:instrument ?instrument .
}
All Canadian songwriters who were born on or before 1950

@prefix dbp: <http://dbpedia.org/property/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
SELECT ?songwriter ?birthYear
WHERE {
  ?songwriter a dbo:Singer-Songwriter ;
    dbo:birthYear ?birthYear ;
  FILTER (?birthYear <= 1950)
}
SPARQL Syntax: Useful FILTER Functions and Operators

- Logical: &&, ||, !
- Mathematical: +, -, *, /
- Comparison: =, !=, <, >, <=, >=
- SPARQL tests: isURI, isBlank, isLiteral, bound, IN
- SPARQL accessors: str, lang, datatype

Full reference: https://www.w3.org/TR/sparql11-query/
All Canadian songwriters who were born on or before 1950 sorted by their year of birth

```sparql
@prefix dbp: <http://dbpedia.org/property/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
SELECT ?songwriter ?birthYear
WHERE {
  ?songwriter a dbo:Singer-Songwriter ;
  dbo:birthYear ?birthYear ;
  FILTER (?birthYear <= 1950)
}
ORDER BY ?birthYear
```
Top 10 oldest living Canadian songwriters

```sparql
@prefix dbp: <http://dbpedia.org/property/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
SELECT ?songwriter
WHERE {
  ?songwriter a dbo:Singer-Songwriter ;
    dbo:birthYear ?birthYear .
  OPTIONAL {
    ?songwriter dbo:deathYear ?deathYear .
  }
  FILTER (!bound(?deathYear))
}
ORDER BY ?birthYear
LIMIT 10
```
Top 10 oldest living Canadian songwriters

```
@prefix dbp: <http://dbpedia.org/property/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
SELECT ?songwriter
WHERE {
    ?songwriter a dbpedia:Songwriter ;
    dbo:birthYear ?birthYear .
    FILTER NOT EXISTS {
        ?songwriter dbpedia:deathYear ?deathYear .
    }
}
ORDER BY ?birthYear
LIMIT 10
```
SPARQL Syntax: Aggregating Results - COUNT

How many albums Leonard Cohen released?

@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

SELECT COUNT (?album) AS ?Number_of_Albums
WHERE {
    ?album dbo:artist dbr:Leonard_Cohen ;
    a dbo:Album .
}
SPARQL Syntax: Aggregating Results – GROUP BY

Which American country musicians released more albums?

```
SELECT ?artist COUNT (?album) AS ?number_of_albums
WHERE {
    ?album dbo:artist ?artist ;
        a dbo:Album .
}
GROUP BY ?artist
ORDER BY desc(?number_of_albums)
LIMIT 25
```
Database management systems

• RDF databases
  – NoSQL DBMS
  – Efficiently process RDF triples
  – Allow SPARQL queries to be executed
  – Offer API that allows such queries to be executed using REST calls (HTTP/SPARQL server)
  – Back-end database engine for storage

• Example
  – OpenLink Virtuoso
  – MarkLogic database
• Public SPARQL endpoint over the DBpedia data set
  – Available at http://dbpedia.org/sparql.
  – Provided using OpenLink Virtuoso.

• Queries against DBpedia using:
  – OpenLink Interactive SPARQL Query Builder (iSPARQL) at http://dbpedia.org/isparql;
  – SNORQL query explorer at http://dbpedia.org/snorql (does not work with Internet Explorer)
  – any other SPARQL-aware client(s).
- People who were born in Berlin before 1900
- Musicians who were born in Berlin
- Soccer players, who are born in a country with more than 10 million inhabitants, who played as goalkeeper for a club that has a stadium with more than 30,000 seats and the club country is different from the birth country
- Games
Exercise

• Try some SPARQL queries from the slides using Berlin SNORQL query explorer