Presentation Outline

- Introducing SPARQL
- Looking back at RDF
- An Introduction to Turtle Notation
- SPARQL Syntax in Depth
- Simple Exercise
- Open Issues and Problems
- Some Interesting Projects
What is SPARQL?

- **SPARQL** – A recursive acronym for SPARQL Protocol and RDF Query Language

- **Protocol**
  - A means for (remote) clients to invoke a service which can perform SPARQL queries.

- **Query Language**
  - A query language for accessing RDF data.
SPARQL – Protocol

Designed in two ways:

• **An Abstract Interface**
  – Described with WSDL 2.0 as a Web Service

• **HTTP and SOAP Bindings**
  – Use HTTP/SOAP to query a SPARQL Web Service
  – Query a SPARQL endpoint, specifying the graphs and the query string
  – Example -->
SPARQL – Protocol Example

**Request:** /sparql/service.aspx?
query=<query_string>&default-graph-uri=<uri1>&named-graph-uri=<uri2>

**Response:**

```xml
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="x"/>
    <variable name="y"/>
  </head>
  <result>
    <binding name="x">
      <uri>http://www.example/123#</uri>
    </binding>
    <binding name="y">
      <bnode>xyz</bnode>
    </binding>
  </result>
</results>
</sparql>
```
Why SPARQL?

SPARQL is the query language of the Semantic Web. It lets us:

- Pull values from structured and semi-structured data
- Explore data by querying unknown relationships
- Perform complex joins of disparate databases in a single, simple query
- Transform RDF data from one vocabulary to another
Review: RDF

• From the introductory slides:

• **RDF** is a data model for objects and relationships between them.

• Essentially, data is represented as triples
  - Subject, Predicate, Object
  - Can also be represented graphically as:

```
Subject ─ Predicate ─ Object
```
Review: RDF

• RDF triples describe resources and their properties.

• A Resource is an object pointed to by an IRI.

• Essentially, an RDF dataset forms a directed graph, denoting the relationship between various resources.
Review: RDF Graph

• Data:
  – A resource with properties name=Alice and mbox=alice@example.net
  – A resource with property name=Bob
Turtle Notation

- Turtle (Turtle RDF Triple Language)
- Subset of N3
- Concise notation for representing RDF data
- Triples are stated as:
  - `<Subject> <predicate> <object> .`
  - The dot (.) signifies the end of the triple.
Turtle Notation: Shorthand

- If the subject is the same:
  ```turtle
  :a :b :c .
  :a :d :e .
  ```
  Can be written as:
  ```turtle
  :a :b :c ;
  :d :e .
  ```

- If the subject and predicate are the same:
  ```turtle
  :a :b :c .
  :a :b :e .
  ```
  Can be written as:
  ```turtle
  :a :b :c, :e .
  ```
Turtle Notation: More Tidbits

• **URIs** are enclosed in `<>

• **Prefixes:**
  - Instead of `<http://example.com/#resource>` a Xyz
  - `@prefix ex <http://example.com/`
  - .. then `ex:resource` a Xyz

• **Blank Nodes**
  - Format `_:nodeName`
  - .. or `[ ]` if the blank node is only being used once.
Turtle Notation: Literals

• Literals
  – "Literal"
  – "Literal"@language
  – """"Long literal with newlines""

• Datatyped Literals
  – "lexical form"^^datatype URI
    • e.g. "10"^^xsd:integer
**Turtle Notation**

- **Data in RDF/XML:**
  ```xml
  <rdf:Description>
    <foaf:mbox rdf:resource="mailto:alice@example.net"/>
    <foaf:name>Alice</foaf:name>
  </rdf:Description>
  <rdf:Description>
    <foaf:name>Bob</foaf:name>
  </rdf:Description>
  ```

- **Data in Turtle:**
  ```turtle
  _:a foaf:name "Alice" ;
  foaf:mbox <mailto:alice@example.net> .
  _:b foaf:name "Bob" .
  ```
Turtle Notation: Blank Nodes

Example

• Blank nodes which are used more than once look like:
  - _:a knows _:b
  - _:a age “17”

• Blank nodes used just once have the following shorthand
  - _:a event [ type “Birth”; date “1984-03-05” ] .

• This is equivalent to:
  - _:a event _:temp .
  - _temp type “Birth”.
  - _temp date “1984-03-05”
SPARQL Syntax - Introduction

• Basic steps to an RDF query:
  1. An RDF graph “pattern”, specified in Turtle, is matched against the RDF data.
  2. All nodes matching a pattern yield a solution
  3. Matched nodes are filtered by keywords such as DISTINCT, ORDER, LIMIT etc.
  4. The resulting nodes are returned in one of the following result forms: SELECT, CONSTRUCT, DESCRIBE or ASK.
SPARQL Syntax – Sample Query

• Data:

```sparql
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Alice" ;
   foaf:mbox <mailto:alice@example.net> .

_:b foaf:name "Bob" .
```

• Query (get all nodes with a name arc):

```sparql
PREFIX foaf: <http://xmlns.com/foaf/0.1/>;

SELECT ?name
WHERE {
  ?node foaf:name ?name .
}
```

Result Format

<table>
<thead>
<tr>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
</tr>
<tr>
<td>?name</td>
</tr>
<tr>
<td>???</td>
</tr>
<tr>
<td>WHERE</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>?node foaf:name ?name .</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>
SPARQL Syntax – Matching against a graph

SELECT ?name WHERE {
  ?node foaf:name ?name .
}

“Alice”

“Bob”

alice@example.net

foaf:mbox

foaf:name
SELECT ?name WHERE {
  ?node foaf:name ?name .
}

“Alice”

foaf:name

alice@example.net

foaf:mbox

“Bob”

foaf:name
SPARQL Syntax – Variable binding

• Variables are specified with ? Or $  
  – For e.g. $name, ?age

• Variable names do not correspond with actual resources or properties.

• Variables can be used in any or all of subject, predicate, and object positions

• Matching the graph means finding a set of bindings such that substituting variables for values creates a triple that is in the set of triples making up the graph.
SPARQL Syntax – Basic Patterns

• A set of triple patterns, all of which must be matched by a solution.

• A Pattern Solution of Graph Pattern GP on graph G is any substitution S such that S(GP) is a subgraph of G.
SPARQL Syntax - Example

- **Data:**
  ```sparql
  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
  _:a foaf:name "Alice" .
  foaf:mbox <mailto:alice@example.net> .

  _:b foaf:name "Bob" .
  ```

- **Query (get all nodes with a name and mbox arc):**
  ```sparql
  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
  SELECT ?name WHERE {
    ?node foaf:name ?name .
  }
  ```

- **Result:**
  ```
  Name
  -----
  Alice
  ```
SPARQL Syntax – OPTIONAL Patterns

• What if we want the previous query to return mbox only if it exists? Remember, RDF is semi-structured.

• Specified with the OPTIONAL Keyword
  – Example: OPTIONAL { :a :b ?x . }

• The specified tuple may or may not exist in the pattern.

• Same as saying: { PATTERN } OR { NOT (PATTERN) }
**SPARQL Syntax - OPTIONALs**

- **Data:**
  ```sparql
  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
  _:a foaf:name  "Alice" ;
  foaf:mbox  <mailto:alice@example.net> .
  _:b foaf:name  "Bob" .
  ```

- **Query (get all nodes with a name and mbox arc):**
  ```sparql
  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
  SELECT ?name, ?mbox WHERE {
    ?node foaf:name ?name .
    OPTIONAL { ?node foaf:mbox ?mbox . }
  }
  ```

- **Result:**
<table>
<thead>
<tr>
<th>Name</th>
<th>Mbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td><a href="mailto:alice@example.net">mailto:alice@example.net</a></td>
</tr>
<tr>
<td>Bob</td>
<td></td>
</tr>
</tbody>
</table>
SPARQL Syntax - Constraints

- FILTER keyword allows us to restrict the solution (For e.g., people older than 20)

- Query:
  ```sparql
  SELECT ?name WHERE {
    ?x foaf:name ?name .
    FILTER (?age >= 20) .
  }
  ```

- Can perform powerful pattern matching using:
  - FILTER regex(..) .
SPARQL Syntax - Constraints

- FILTER value tests are based on XQuery 1.0 and XPath 2.0.
- Can perform value comparisons (=, >, <, != etc) on all data types (XSD boolean, string, integer etc)
- Other functions also available such as bound, isURI, isBLANK, isLITERAL
- Can even be extended to included custom functions.
SPARQL Syntax – Group Patterns

- Group patterns match if all subpatterns match and all constraints are satisfied.
- In SPARQL Syntax, groups are specified with {}.
- Useful if we want to perform one query over multiple graphs, or if we want to select parts of data from one graph and parts from another.
SPARQL Syntax – Group Patterns (Union Example)

• Data:

```sparql
@prefix dc10: <http://purl.org/dc/elements/1.0/> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
_:a dc10:title "SPARQL Query Language Tutorial" .
_:b dc11:title "SPARQL Query Language (2nd ed)" .
_:c dc10:title "SPARQL" .
_:c dc11:title "SPARQL" .
```

• Query (get all nodes with a name and mbox arc):

```sparql
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
SELECT DISTINCT ?title
```

• Result:

```text
<table>
<thead>
<tr>
<th>title</th>
</tr>
</thead>
</table>
| "SPARQL Query Language Tutorial"
| "SPARQL"
| "SPARQL Query Language (2nd ed)"
```
SPARQL Syntax – Group Patterns using the GRAPH keyword

• Query:

```
SELECT ?rev1rating ?rev2rating
WHERE {
  GRAPH <http://example.org/reviews/author1> {
  }
  GRAPH <http://example.org/reviews/author2> {
  }
}
```
SPARQL Syntax – Solution Modifiers

- Data returned by a SPARQL query is unordered. We can modify the solution using the following modifiers:
  - `DISTINCT`: ensure that the results are unique. Similar to SQL.
  - `ORDER BY`: Sort the result by one of the variable bindings.
    - `ORDER BY ?x ?y`
    - `ORDER BY DESC(?x)`
SPARQL Syntax – Solution
Modifiers

- LIMIT / OFFSET: Similar to SQL. LIMIT is used to restrict the number of results (LIMIT 5), and OFFSET is used to show results after skipping a specified number of results (OFFSET 5).
SPARQL Syntax – Result Forms

- SELECT is just one of the 4 ways a SPARQL query can return results
- ASK returns a boolean result (were any triples found? i.e. Was the pattern matched or not?)
- CONSTRUCT allows construction of an RDF graph from the results of the SPARQL query
- DESCRIBE also creates a graph, however the form of the graph is controlled by the server or query processor, not the application making the query.
**SPARQL Syntax - CONSTRUCT**

- **Query:**

```
CONSTRUCT { ?friend pim:fullName ?name .
            ?friend foaf:mbox ?mbox }
        FILTER regex(?person, "Alice") .}
```

- **What does this query do?**
- It constructs an rdf graph with a node for all people who Alice has an foaf:knows relationship with, with arcs specifying the person's name and mbox.
SPARQL Syntax - DESCRIBE

• Query:

```sparql
DESCRIBE ?friend
  FILTER regex(?person, "Alice") .}
```

• Result (controlled by server):

```xml
<rdf:Description>
  <foaf:givenName>Johnny</foaf:givenName>
  <foaf:knows>
    ...
  </foaf:knows>
</rdf:Description>
```
SPARQL Syntax – Simple Exercise

• Data:
  @prefix ... <snipped>

  _:a a foaf:Person ;
  foaf:name "Alice" ;
  foaf:mbox <mailto:alice@work.example.com> ;
  foaf:mbox <mailto:alice@home.example.org> ;
  foaf:knows _:b ;

  _:b a foaf:Person ;
  foaf:name "Bob" ;
  foaf:mbox <mailto:bob@work.example.com> ;
  bio:event [ a bio:Birth; bio:date "1973-01-02"^^xsd:date ] ;
  rdf:type foaf:Person .

  _:a foaf:knows _:c .
  _:c foaf:mbox <mailto:eve@work.example.com> .

• Write a query which lists all of Alice's email addresses
SPARQL Syntax – Simple Exercise

• Data:

```sparql
@prefix ... <snipped>
_:a a foaf:Person ;
  foaf:name "Alice" ;
  foaf:mbox <mailto:alice@work.example.com> ;
  foaf:mbox <mailto:alice@home.example.org> ;
  foaf:knows _:b ;
_:b a foaf:Person ;
  foaf:name "Bob" ;
  foaf:mbox <mailto:bob@work.example.com> ;
  bio:event [ a bio:Birth; bio:date "1973-01-02"^^xsd:date ] ;
  rdf:type foaf:Person .
_:a foaf:knows _:c .
_:c foaf:mbox <mailto:eve@work.example.com> .
```

• Write a query which lists email addresses of all people known to Alice.
SPARQL Syntax – Simple Exercise

• Data:
  @prefix ... <snipped>

    _:a a foaf:Person ;
    foaf:name   "Alice" ;
    foaf:mbox   <mailto:alice@work.example.com> ;
    foaf:mbox   <mailto:alice@home.example.org> ;
    foaf:knows  _:b ;
    bio:event   [ a bio:Birth; bio:date "1974-02-28"^^xsd:date ] ;

    _:b a foaf:Person ;
    foaf:name   "Bob" ;
    foaf:mbox   <mailto:bob@work.example.com> ;
    bio:event   [ a bio:Birth; bio:date "1973-01-02"^^xsd:date ] ;
    rdf:type    foaf:Person .

    _:a foaf:knows  _:c .
    _:c foaf:mbox  <mailto:eve@work.example.com> .

• Write a query to figure out Alice's Birth date.
SPARQL Syntax – Simple Exercise

• Data:
  @prefix ... <snipped>

    _:a a foaf:Person ;
    foaf:name "Alice" ;
    foaf:mbox <mailto:alice@work.example.com> ;
    foaf:mbox <mailto:alice@home.example.org> ;
    foaf:knows _:b ;

    _:b a foaf:Person ;
    foaf:name "Bob" ;
    foaf:mbox <mailto:bob@work.example.com> ;
    bio:event [ a bio:Birth; bio:date "1973-01-02"^^xsd:date ] ;
    rdf:type foaf:Person .

    _:a foaf:knows _:c .
    _:c foaf:mbox <mailto:eve@work.example.com> .

• Write a query which lists name and/or email addresses of all people known to Alice.
Open Issues and Problems

- Even though SPARQL has gained a lot of traction recently, there are many open issues. For example:
  - No Updates. SPARQL can only read data.
  - No computed results.
    - Select (?x + ?y) .... (Lets say to sum up product + shipping cost)
  - No aggregate functions.
    - Select count(?x) ... (For inventorying purposes maybe)
  - No support for RDF collections of the (rdf:first, rdf:rest form)
Interesting Projects

• DBPedia (http://dbpedia.org)

  “DBpedia is a community effort to extract structured information from Wikipedia and to make this information available on the Web. DBpedia allows you to ask sophisticated queries against Wikipedia and to link other datasets on the Web to Wikipedia data.”
<table>
<thead>
<tr>
<th>Nr.</th>
<th>film</th>
<th>actor</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Godfather Saga</td>
<td>Al Pacino</td>
<td>434 min.</td>
</tr>
<tr>
<td>2</td>
<td>Fanny and Alexander</td>
<td>Börje Ahlstedt</td>
<td>188 min.312 min. (TV version)</td>
</tr>
<tr>
<td>3</td>
<td>Berlin Alexanderplatz</td>
<td>Barbara Sukowa</td>
<td>USA: 931 min West Germany: 894 min</td>
</tr>
<tr>
<td>4</td>
<td>Out 1</td>
<td>Bernadette Lafont</td>
<td>773 minutes</td>
</tr>
<tr>
<td>5</td>
<td>Fanny and Alexander</td>
<td>Bertil Guve</td>
<td>188 min.312 min. (TV version)</td>
</tr>
<tr>
<td>6</td>
<td>Out 1</td>
<td>Bulle Ogier</td>
<td>773 minutes</td>
</tr>
<tr>
<td>7</td>
<td>Island at War</td>
<td>Clare Holman</td>
<td>approx. 398 min</td>
</tr>
<tr>
<td>8</td>
<td>The 10th Kingdom</td>
<td>Daniel Lapaine</td>
<td>Approx. Running Feature Time 417 minutes</td>
</tr>
<tr>
<td>9</td>
<td>The Deluge (film)</td>
<td>Daniel Olbrychski</td>
<td>315 min.</td>
</tr>
<tr>
<td>10</td>
<td>The Godfather Saga</td>
<td>Diane Keaton</td>
<td>434 min.</td>
</tr>
</tbody>
</table>
RDFa

- Allows content publishers to indicate the semantics of the content “inline” on a web page.

- Example:
  ```html
  <h1>Books</h1>
  
  <table>
    <tr>
      <td about='http://example.com/book1' property='title'>Harry Potter and the Philosopher's Stone</td>
    </tr>
  </table>
  ```

- This is equivalent to the following RDF triple:
  ```xml
  ```
Summary

- SPARQL is a W3C Recommendation (as of January 2008) for querying RDF data.
- Essentially, it allows matching a pattern over an RDF graph, or even multiple graphs, using the Turtle Notation.
- There are still open areas such as updates in SPARQL.
- Apart from its usability as a query language for the Semantic Web, SPARQL is playing an important transitional role as well.
Questions, Comments, Criticism?

Hasan Ammar
ammarh@gmail.com