NEW! The **Semantic Web Primer e-Book (First Edition)** includes **all** our primer tutorials

Tutorial 5: Querying Semantic Data

Now you know how to add semantics to your RDF data, how is it published and queried? Like the tables of a relational database are queried using SQL, the triples of RDF data are queried using SPARQL. We investigate some basic queries and compare SPARQL with the more familiar SQL to make it easier.

After this tutorial, you should be able to:

- Build your own basic SPARQL queries, step-by-step
- Understand that SPARQL is not just a query language, it is also a protocol
- Understand the XML format in which SPARQL protocol returns the results of queries
- Try executing some SPARQL queries yourself on some UK government RDF data
- Look ahead to SPARQL+ which has add, modify and delete capabilities

Estimated time: 5 minutes

You should have already understood the following lesson (and pre-requisites) before you begin:

- Tutorial 4: Introducing RDFS & OWL

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Plus two **exclusive** new tutorials on **RDF syntaxes**, and **NoSQL databases** found **only** in the e-Book.

Get Your Copy >>

If you come from a traditional IT background, you may well already be familiar with SQL (Structured Query Language, pronounced "sequel") which is used to retrieve data from a relational database such as MS SQL or MySQL.

Similarly, RDF data stores can also be queried using their own query language - SPARQL (SPARQL Protocol and RDF Query Language, pronounced "sparkle"). SPARQL is, however, a little more sophisticated.

SPARQL is a W3C standard and is currently at version 1.1.

5.1 A Starting Example

Just to show you what SPARQL looks like, here is a quick example:

```
PREFIX sch-ont:   <http://education.data.gov.uk/def/school/>
SELECT ?name WHERE {
  ?school a sch-ont:School.
  ?school sch-ont:establishmentName ?name.
  ?school sch-ont:districtAdministrative <http://statistics.data.gov.uk/id/local-authority-district/00AA>.
}
ORDER BY ?name
```

Do not worry if this all looks unfamiliar for now.

This query, if executed on the UK government's Open Semantic Database, will return the names of all the schools in the UK in administrative district 00AA, and orders the results in alphabetical order. Take a brief look at the query and see if you can see why this might be before we examine it more closely.

Try It Yourself Copy and paste the above query into the UK government's SPARQL query endpoint at http://education.data.gov.uk/sparql/education/query.html and the see the results. Such an URL is called a **SPARQL endpoint** in the semantic web world - and is the way in which the data on the semantic web is published to the outside world in a query enabled form.

SPARQL Is Similar To SQL

Like SQL, SPARQL selects data from the query data set by using a **SELECT** statement to determine which subset of the selected data is returned. Also, SPARQL uses a **WHERE** clause to define **graph patterns** to find a match for in the query data set.

A **graph pattern** in a SPARQL **WHERE** clause consists of the subject, predicate and object triple to find a match for in the data. Let's explore this further by taking a closer look at the example above.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

The **SELECT** statement requests the variable **?name** to be returned.

Note In SPARQL, variable names are prefixed with the question mark ("?") symbol. In a query **graph pattern**, they match any node - whether resource or literal.

Notice that this variable is also given in the **WHERE** clause search pattern - on the object of the second query search pattern. But also note the **?school** variable too. Because a specific URI has not been stated for a match but a variable, *any* matching subject URI will be returned for this part of the query pattern and the result will be *mapped* onto that variable name.

Hence, in the above SPARQL query, **?name** returns *all* the names of the schools which match the three search patterns given in the query. If we wanted, we could make the query more specific by adding additional match criteria. Or, we could make it more broad for example by removing the last search pattern requiring the school to match the district administrative value "00AA".

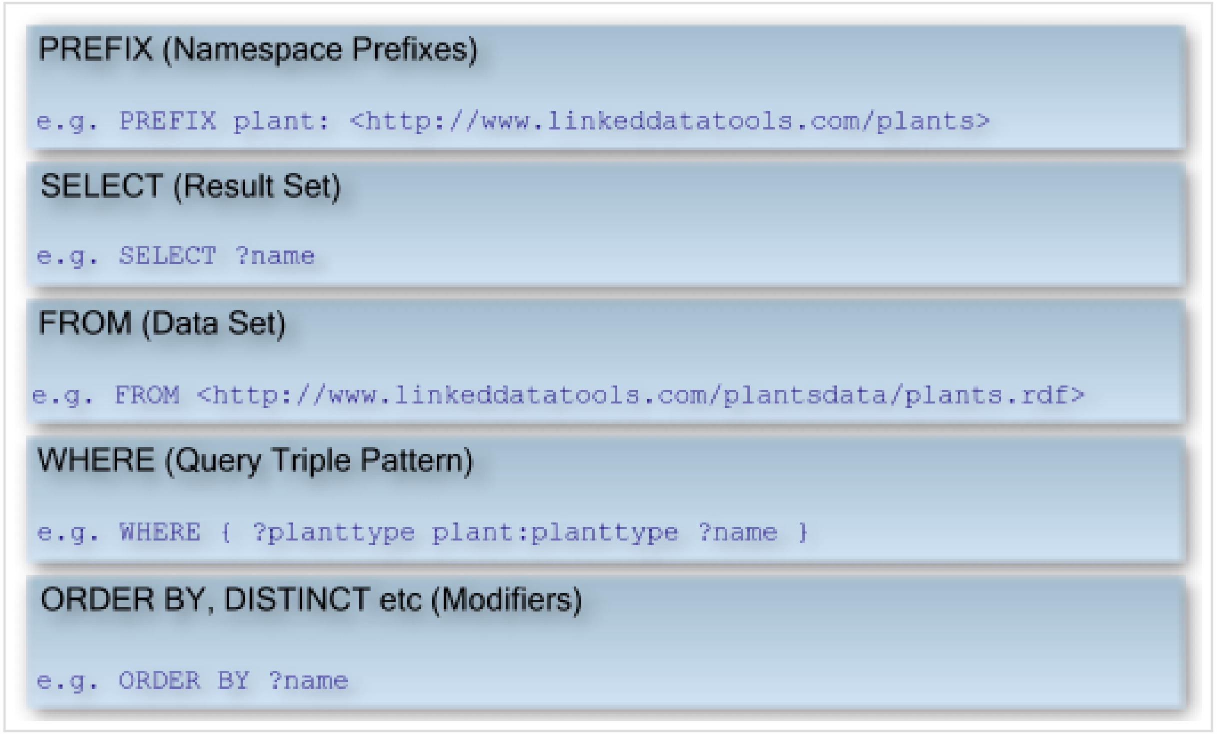
Lastly, note that the **?school** variable means that for all three search patterns, *any* subject will match the search pattern and will be returned to this variable. But, since it is not stated in the **SELECT** statement for this query, **?school** is mapped, but not returned in the result set.

You can see this for yourself by running this query on the government database as we suggest above.

5.2 SPARQL General Form

SPARQL queries take the following general form, showing the sections into which a query may be broken down and the clause or keyword which defines that section.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com



PREFIX (Namespace Prefixes)

e.g. `PREFIX plant: <http://www.linkeddatatools.com/plants>`

SELECT (Result Set)

e.g. `SELECT ?name`

FROM (Data Set)

e.g. `FROM <http://www.linkeddatatools.com/plantsdata/plants.rdf>`

WHERE (Query Triple Pattern)

e.g. `WHERE { ?planttype plant:planttype ?name }`

ORDER BY, DISTINCT etc (Modifiers)

e.g. `ORDER BY ?name`

Or, taking the examples for each section given above we have the following query:

```
PREFIX plant:   <http://www.linkeddatatools.com/plants>
FROM <http://www.linkeddatatools.com/plantsdata/plants.rdf>
SELECT ?name WHERE {
  ?planttype plant:planttype ?name.
}
ORDER BY ?name
```

Let's make it easy to understand how a SPARQL query is formed by building one, step-by-step.

Important Point The whole point of SPARQL is to provide a formal language with which to ask meaning-driven questions (such as the question asked above). As you learn SPARQL and practice writing queries more, it will become easier to translate those questions into queries. And to be able to turn basic, meaningful questions into queries that return meaningful results is, in fact, the primary unique benefit of semantic web techologies.

5.3 Building A SPARQL Query

To learn how to construct a SPARQL query on some triple data, let's start off with an example data set: one based upon the ontology for plants & shrubs that we defined in our previous tutorial.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Example triple data containing a variety of shrubs and plants, and their family names

Subject	Predicate	Object
http://www.linkeddatatools.com/plants#magnolia	http://www.linkeddatatools.com/plants#family	Magnoliaceae
http://www.linkeddatatools.com/plants#african_lilly	http://www.linkeddatatools.com/plants#family	Liliaceae
http://www.linkeddatatools.com/plants#silvertop	http://www.linkeddatatools.com/plants#family	Aralianae
http://www.linkeddatatools.com/plants#velvetleaf	http://www.linkeddatatools.com/plants#family	Malvaceae
http://www.linkeddatatools.com/plants#manglietia	http://www.linkeddatatools.com/plants#family	Magnoliaceae

Our example data set contains 5 plants & shrubs. The subject is a URI representing the plant (made readable by the fact the URI uses the common name of the plant or shrub). The predicate is the family name, taken from the ontology we defined previously. Then, we have literal objects, comprising a string value of the scientific family name of the plant or shrub.

Let's construct some simple queries on this data and look at the results.

NEW! Semantic Web Primer e-Book (First Edition)

Includes **all** our primer tutorials. Plus two **exclusive** new tutorials on **RDF syntaxes**, and **NoSQL databases** found **only** in the e-Book.

Get Your Copy >>

Select All Data

First, let's construct a query that will select all the plant URIs (subjects) and plant family names (literal-type objects) from the data above.

```
PREFIX plants: <http://www.linkeddatatools.com/plants>
SELECT * WHERE
{
   ?name plants:family ?family
}
```

The wildcard '*' in SPARQL, similar to SQL, will return all the mapped data in the result set. What does this mean? As we have stated two variables, **?name** and **?family**, the query will return both these declared

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

query variables in our result set, according to the subjects, predicates or objects they're mapped to.

So we've defined the query; now let's execute it. SPARQL is not just a query language, it's also a protocol, and it returns results in a specific schema that your software can read. One of the most widely used forms of result sets is the XML result format. Here's the XML format result set we get back, below:

```
<?xml version="1.0" ?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <head>
      <variable name="name"/>
      <variable name="family"/>
   </head>
   <results>
      <result>
         <binding name="name">
            <uri>http://www.linkeddatatools.com/plants#magnolia</uri>
         </binding>
         <binding name="family">
            <literal>Magnoliaceae</literal>
         </binding>
      </result>
      <result>
         <binding name="name">
            <uri>http://www.linkeddatatools.com/plants#african_lilly</uri>
         </binding>
         <binding name="family">
            <literal>Liliaceae</literal>
         </binding>
      </result>
      <result>
         <binding name="name">
            <uri>http://www.linkeddatatools.com/plants#silvertop</uri>
         </binding>
         <binding name="family">
            <literal>Aralianae</literal>
         </binding>
      </result>
      <result>
         <binding name="name">
            <uri>http://www.linkeddatatools.com/plants#velvetleaf</uri>
         </binding>
         <binding name="family">
            <literal>Malvaceae</literal>
         </binding>
      </result>
      <result>
         <binding name="name">
            <uri>http://www.linkeddatatools.com/plants#manglietia</uri>
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
            </binding>
            <binding name="family">
               <literal>Magnoliaceae</literal>
            </binding>
         </result>
      </results>
   </sparql>
```

It's fairly readable and verbose as it is. We can see we have our root <sparql> node, which can be used to define any XML namespaces that the result set uses. Then, because we've used a wildcard (*) in our **SELECT** statement, we have retrived both the variables we declared in our query **graph patterns** in our **WHERE** clause - **?name** and **?family**. These, as you can see, are defined in the result set's <head> section, to say these are the variables returned by the query.

Then we get our results, which are bound to these query variables. Each pattern match is returned in its own <result> node. Note if there were no matches resulting from our query, we would get no results returned in our XML.

You can see the way in which our result XML distinguishes between URI type values and literal values, using the <uri> and <literal> tags to enclose the data. It would be up to the software you have written to parse this data to use these distinctions as you require. But the fact that they are typed this way is extremely useful to whatever system is using the returned results.

Select Only "Magnoliaceae" Family

Let's be more specific - let's return only those plants from the Magnoliaceae family from our data set. How could this be done?

```
PREFIX plants: <http://www.linkeddatatools.com/plants>
SELECT * WHERE
{
   ?name plants:family "Magnoliaceae"
}
```

We've simply removed the **?family** variable from the previous query and replaced it with the literal "Magnoliaceae" instead - which means our query will now seek for an exact match on this literal rather than any object match.

Executing the query and returning the result, we get:

```
<?xml version="1.0" ?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
    <head>
        <variable name="name"/>
    </head>
    <results>
        <result>
            <binding name="name">
                <uri>http://www.linkeddatatools.com/plants#magnolia</uri>
            </binding>
        </result>
        <result>
            <binding name="name">
                <uri>http://www.linkeddatatools.com/plants#manglietia</uri>
            </binding>
        </result>
    </results>
</sparql>
```

Because we have removed the **?family** variable from our **WHERE** clause, it is of course no longer present in the results and our <head> section contains the only return variable, **?name**, and our two matching triple patterns - both members of the Magnoliaceae family with the literal value "Magnoliaceae" as their object value.

5.4 Further Exploration

This has given you a starting flavor of what querying triple data is like. Building SPARQL queries is something that takes practice, but well worth the effort to get to know. Now you have a good idea of what form a SPARQL query takes, try practicing your own queries or by reading further.

Fortunately, W3C have produced a good introduction to the topic that goes beyond the scope of this primer that we would recommend as a next avenue of learning. You can find this at on their SPARQL Query Language For RDF page.

Note You may like to try SPARQL queries of your own using our linked data tool suite **RDF Studio**. RDF Studio has full SPARQL help (including autocomplete of your queries) to help you write queries as you type.

5.5 What About CREATE, INSERT, UPDATE?

You may have noticed that whilst we have covered the basics of the **SELECT-WHERE** form of SPARQL to obtain subsets of data from a triple data store, we haven't yet covered the **CREATE**, **INSERT** or **UPDATE** methods. There's a good reason for this; at the moment, they are not implemented.

If you think about it, at some level it makes sense. Organisations publishing data in a queryable, public

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

SPARQL endpoint will probably in most circumstances not just want anyone writing and making changes to their (valuable) data. But also, at some level, it does seem like it's a missing requirement, particularly if you're using triple data stores locally rather than publicly across the web and want to make changes using a query language (like you would with a SQL database). How could this be done?

At the moment, new specifications such as SPARUL (SPARQL/Update) and SPARQL+ are being developed to address this problem, however a solid contender to fill this gap in functionality is yet to arise, at time of writing.

As RDF data is more widely adopted however, expect a winning contender to emerge and be implemented by Semantic Web frameworks. We will of course be taking notice too!

NEW! Semantic Web Primer e-Book (First Edition)

Includes **all** our primer tutorials. Plus two **exclusive** new tutorials on **RDF syntaxes**, and **NoSQL databases** found **only** in the e-Book.

Get Your Copy >>

You have completed this tutorial. You should now understand the following:

- That SPARQL can query RDF datasets, rather like SQL can query a relational database.
- That SPARQL endpoints are used to query and return data from the semantic web.
- How you can build and execute simple SPARQL queries yourself.
- Have a starting knowledge of the form in which SPARQL queries return results.

Well Done You have now completed our short course on the semantic web. We will be publishing more advanced topics over the coming months that will explore these themes further, and would invite you to check back for updates. We hope you have found these articles easy to read and useful as you explore this technology further. If you have found these tutorials useful, we would appreciate a link so that others can find us in future - simply click on the "share" button below for a variety of options.

SHARE

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Community

Register to download software from our site and interact with other users as you learn semantic web.

A+ A- Reset

Copyright © 2009-2015 LinkedDataTools.com. LinkedDataTools.com is for training only. We do not warrant the correctness of its content. The risk from using it, or any software downloaded from the site, lies entirely with the user. While using our site, you agree to have read and accepted our terms of use and privacy policy. Contact Us

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com