**LECTURE 8**

**RESOURCE DESCRIPTION FRAMEWORK**

RDF is a model to represent data about physical objects and abstract concepts. It’s a model to express relations between entities using a graph format.

**RDF allows describing anything: persons, animals, objects, and concepts of any kind**. They are considered resources. RDF represents meaningful information for software applications. Though, humans could read and use RDFs.

We represent information by statements in the following format:

<subject> <predicate> <object>

Those statements express a relation between the subject and the object. Both, the subject, and the object are resources.

Let’s see some RDF examples in pseudocode:

<John> <is a> <person>

<John> <is a friend of> <Jane>

<John> <is born on> <the 10th of May 2000>

<John> <is interested in> <the Rosetta Stone>

<the sculpture of David> <is in> <British Museum>

RDF Resource, Property, and Property Value

RDF identifies things using Web identifiers (URIs), and describes resources with properties and property values.

Explanation of Resource, Property, and Property value:

* A **Resource** is anything that can have a URI, such as "https://www.w3schools.com/rdf"
* A **Property** is a Resource that has a name, such as "author" or "homepage"
* A **Property value** is the value of a Property, such as "Jan Egil Refsnes" or "https://www.w3schools.com" (note that a property value can be another resource)



<rdf:RDF> is the root element of an RDF document. It defines the XML document to be an RDF document. It also contains a reference to the RDF namespace:

The <rdf:Description> element identifies a resource with the about attribute.

The <rdf:Description> element contains elements that describe the resource:







**RDF STATEMENTS**

The combination of a Resource, a Property, and a Property value forms a **Statement** (known as the **subject, predicate and object** of a Statement).

Let's look at some example statements to get a better understanding:

Statement: "The author of https://www.w3schools.com/rdf is Jan Egil Refsnes".

* The subject of the statement above is: https://www.w3schools.com/rdf
* The predicate is: author
* The object is: Jan Egil Refsnes

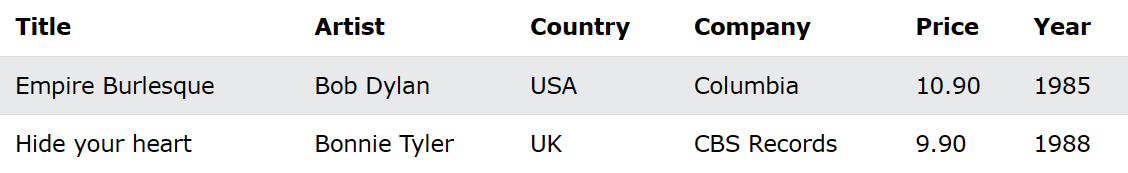
Statement: "The homepage of https://www.w3schools.com/rdf is https://www.w3schools.com".

* The subject of the statement above is: https://www.w3schools.com/rdf
* The predicate is: homepage
* The object is: https://www.w3schools.com

<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:si="https://www.w3schools.com/rdf/">  
  
<rdf:Description rdf:about="https://www.w3schools.com">  
  <si:title>W3Schools</si:title>  
  <si:author>Jan Egil Refsnes</si:author>  
</rdf:Description>

## RDF Example

Here are two records from a CD-list:



<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://www.recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">  
  <cd:artist>Bob Dylan</cd:artist>  
  <cd:country>USA</cd:country>  
  <cd:company>Columbia</cd:company>  
  <cd:price>10.90</cd:price>  
  <cd:year>1985</cd:year>  
</rdf:Description>  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Hide your heart">  
  <cd:artist>Bonnie Tyler</cd:artist>  
  <cd:country>UK</cd:country>  
  <cd:company>CBS Records</cd:company>  
  <cd:price>9.90</cd:price>  
  <cd:year>1988</cd:year>  
</rdf:Description>  
.  
.  
.  
</rdf:RDF>

Properties as Attributes

The property elements can also be defined as attributes (instead of elements):

<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://www.recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Empire Burlesque"  
cd:artist="Bob Dylan" cd:country="USA"  
cd:company="Columbia" cd:price="10.90"  
cd:year="1985" />  
  
</rdf:RDF>

Properties as Resources

The property elements can also be defined as resources:

<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://www.recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">  
  <cd:artist rdf:resource="http://www.recshop.fake/cd/dylan" />  
  ...  
  ...  
</rdf:Description>  
  
</rdf:RDF>

## The <rdf:Bag> Element

The <rdf:Bag> element is used to describe a list of values that do not have to be in a specific order.

The <rdf:Bag> element may contain duplicate values.

### **Example**

<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://www.recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Beatles">  
  <cd:artist>  
    <rdf:Bag>  
      <rdf:li>John</rdf:li>  
      <rdf:li>Paul</rdf:li>  
      <rdf:li>George</rdf:li>  
      <rdf:li>Ringo</rdf:li>  
    </rdf:Bag>  
  </cd:artist>  
</rdf:Description>  
  
</rdf:RDF>

The <rdf:Seq> Element

The <rdf:Seq> element is used to describe an ordered list of values (For example, in alphabetical order).

The <rdf:Seq> element may contain duplicate values.

Example

<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://www.recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Beatles">  
  <cd:artist>  
    <rdf:Seq>  
      <rdf:li>George</rdf:li>  
      <rdf:li>John</rdf:li>  
      <rdf:li>Paul</rdf:li>  
      <rdf:li>Ringo</rdf:li>  
    </rdf:Seq>  
  </cd:artist>  
</rdf:Description>  
  
</rdf:RDF>

## The <rdf:Alt> Element

The <rdf:Alt> element is used to describe a list of alternative values (the user can select only one of the values).

### **Example**

<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://www.recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Beatles">  
  <cd:format>  
    <rdf:Alt>  
      <rdf:li>CD</rdf:li>  
      <rdf:li>Record</rdf:li>  
      <rdf:li>Tape</rdf:li>  
    </rdf:Alt>  
  </cd:format>  
</rdf:Description>  
  
</rdf:RDF>

## The rdf:parseType="Collection" Attribute

As seen in the previous chapter, a container says that the containing resources are members - it does not say that other members are not allowed.

RDF collections are used to describe groups that can ONLY contain the specified members.

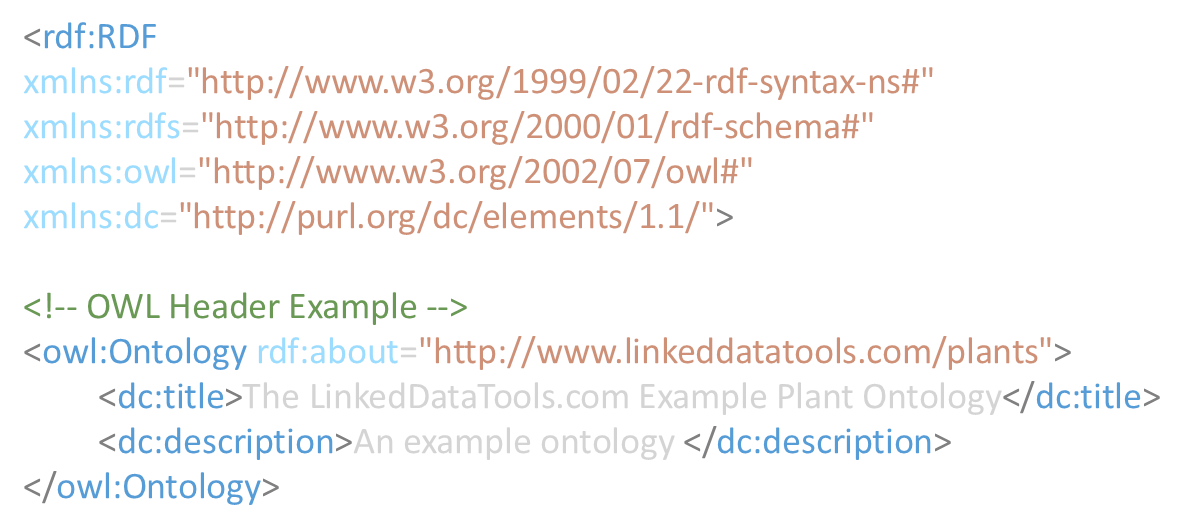
A collection is described by the attribute rdf:parseType="Collection".

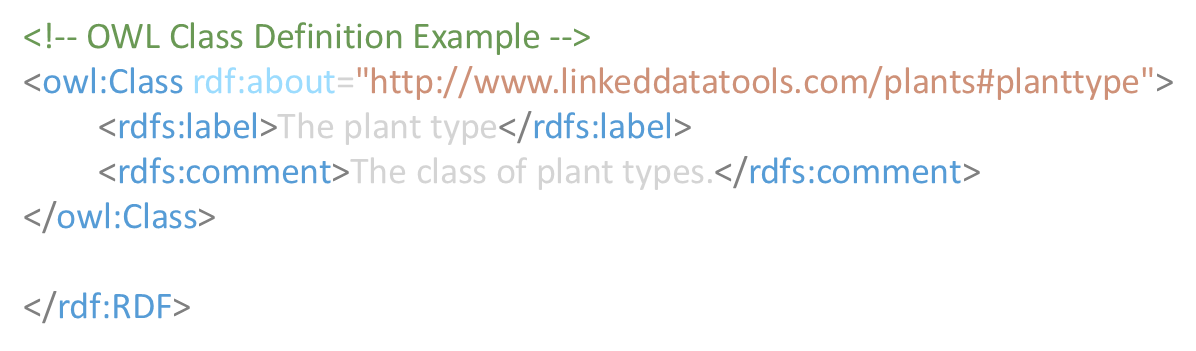
### **Example**

<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://recshop.fake/cd/Beatles">  
  <cd:artist rdf:parseType="Collection">  
    <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/George"/>  
    <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/John"/>  
    <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/Paul"/>  
    <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/Ringo"/>  
  </cd:artist>  
</rdf:Description>  
  
</rdf:RDF>

**RDFS AND OWL**

**DATA ANNOTATION WITH SEMANTIC META DATA USING RDFS AND OWL**

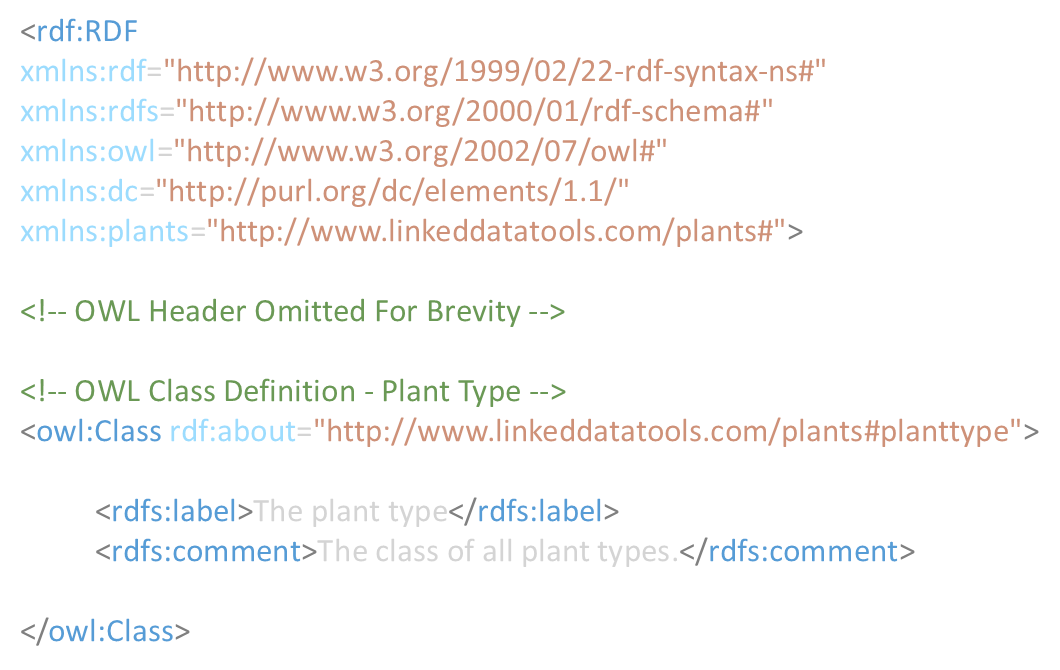


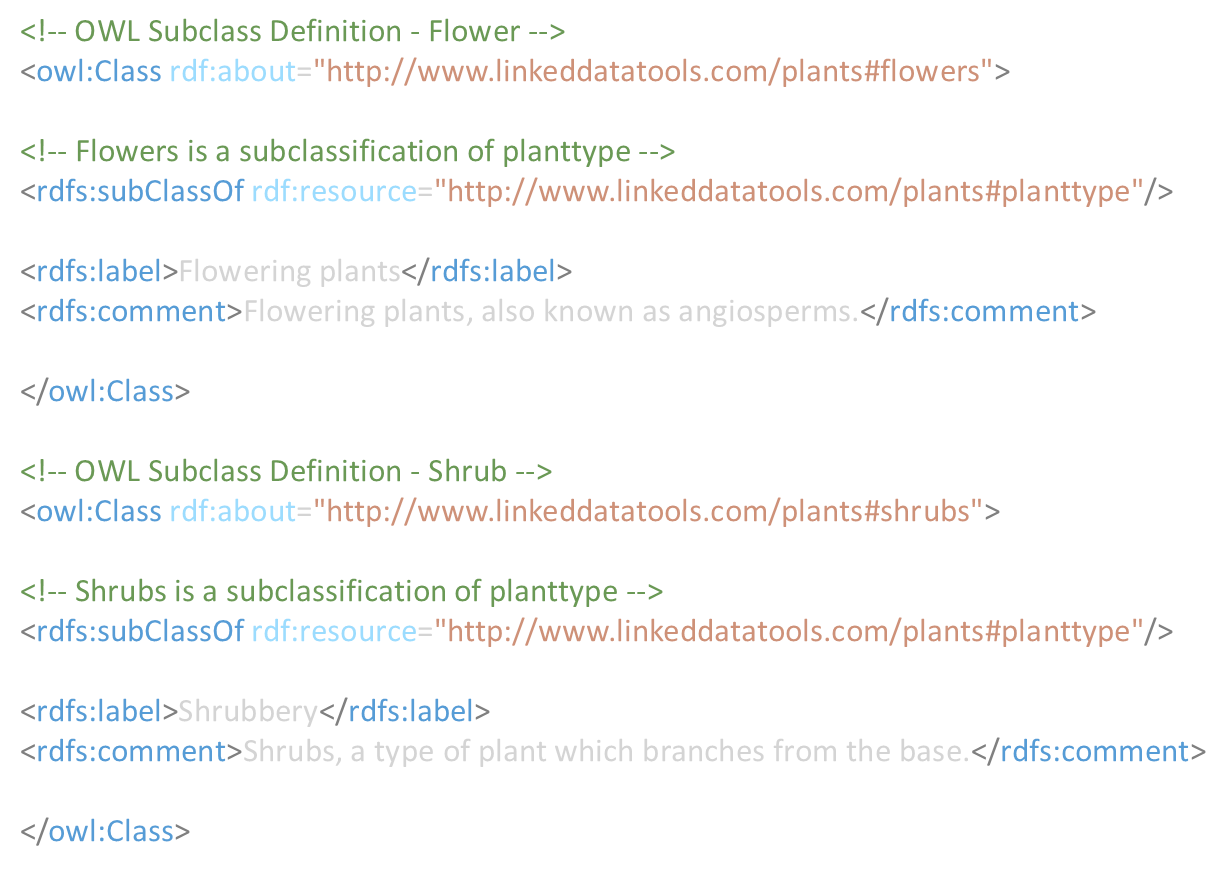


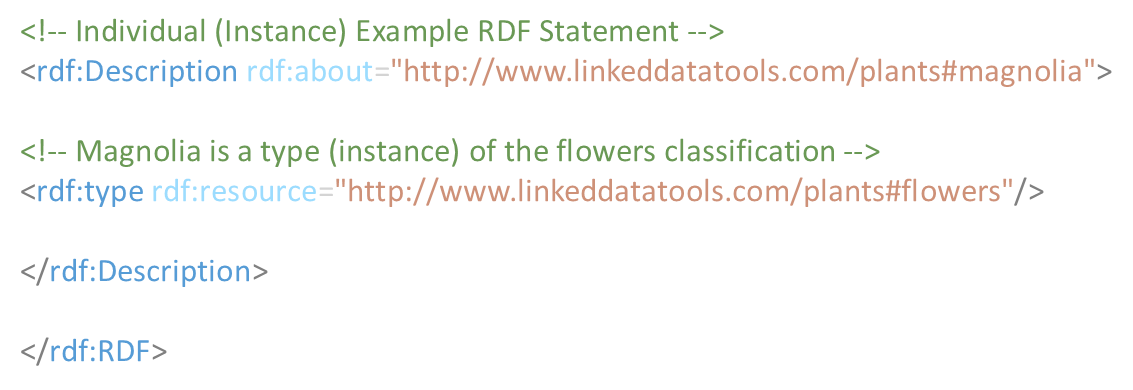
**OWL CLASSESS, SUBCLASSES AND INDIVIDUALS**

The primary purpose of your ontology is to classify things in terms of semantics, or meaning. In OWL, this is achieved through the use of *classes* and *subclasses*, instances of which in OWL are called *individuals*. The individuals that are members of a given OWL class are called its *class extension*.

A *class* in OWL is a classification of individuals into groups which share common characteristics. If an individual is a member of a class, it tells a machine reader that it falls under the semantic classification given by the OWL class.

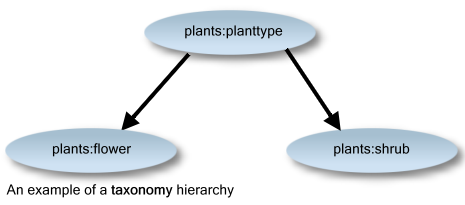






**Taxonomy – A Hierarchy Of Terms**

What we’ve done is define our semantic terms, or classes, in a hierarchy. In the semantic web world, this hierarchy of terms is called a *taxonomy*. Here’s a graphical illustration the taxonomy hierarchy we’ve defined:



**Note** We haven’t created another *subclass* of the **flower** class called **magnolia**. Rather, **magnolia** is an *individual* (instance) of class **flower**. Why is this? Magnolia is a member of the **flower** classification, but it is **not** a further flower **subclassification**. It makes sense from a semantic perspective for magnolia – and indeed other flowers – to be *individuals* (instances) of the class **flower** and not *subclassifications*.

**OWL PROPERTIES**

Individuals in OWL are related by *properties*. There are two types of property in OWL:

* **Object properties** (owl:ObjectProperty) relates individuals (instances) of two OWL classes.
* **Datatype properties** (owl:DatatypeProperty) relates individuals (instances) of OWL classes to literal values.

<rdf:RDF

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

xmlns:owl="http://www.w3.org/2002/07/owl#"

xmlns:dc="http://purl.org/dc/elements/1.1/"

xmlns:plants="http://www.linkeddatatools.com/plants#">

<!-- OWL Header Omitted For Brevity -->

<!-- OWL Classes Omitted For Brevity -->

<!-- Define the family property -->

<owl:DatatypeProperty rdf:about="http://www.linkeddatatools.com/plants#family"/>

<rdf:Description rdf:about="http://www.linkeddatatools.com/plants#magnolia">

<!-- Magnolia is a type (instance) of the flowers class -->

<rdf:type rdf:resource="http://www.linkeddatatools.com/plants#flowers"/>

<!-- The magnolia is part of the 'Magnoliaceae' family -->

<plants:family>Magnoliaceae</plants:family>

</rdf:Description>

</rdf:RDF>

ADDING OBJECT PROPERTY

<rdf:RDF

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:owl="http://www.w3.org/2002/07/owl#"

xmlns:dc="http://purl.org/dc/elements/1.1/"

xmlns:plants="https://www.linkeddatatools.com/plants#">

<!-- OWL Header Omitted For Brevity -->

<!-- OWL Classes Omitted For Brevity -->

<!-- Define the family property -->

<owl:DatatypeProperty rdf:about="https://www.linkeddatatools.com/plants#family"/>

<!-- Define the similarlyPopularTo property -->

<owl:ObjectProperty rdf:about="https://www.linkeddatatools.com/plants#similarlyPopularTo"/>

<!-- Define the Orchid class instance -->

<rdf:Description rdf:about="https://www.linkeddatatools.com/plants#orchid">

<!-- Orchid is an individual (instance) of the flowers class -->

<rdf:type rdf:resource="https://www.linkeddatatools.com/plants#flowers"/>

<!-- The orchid is part of the 'Orchidaceae' family -->

<plants:family>Orchidaceae</plants:family>

<!-- The orchid is similarly popular to the magnolia -->

<plants:similarlyPopularTo rdf:resource="https://www.linkeddatatools.com/plants#magnolia"/>

</rdf:Description>

<!-- Define the Magnolia class instance -->

<rdf:Description rdf:about="https://www.linkeddatatools.com/plants#magnolia">

<!-- Magnolia is an individual (instance) of the flowers class -->

<rdf:type rdf:resource="https://www.linkeddatatools.com/plants#flowers"/>

<!-- The magnolia is part of the 'Magnoliaceae' family -->

<plants:family>Magnoliaceae</plants:family>

<!-- The magnolia is similarly popular to the orchid -->

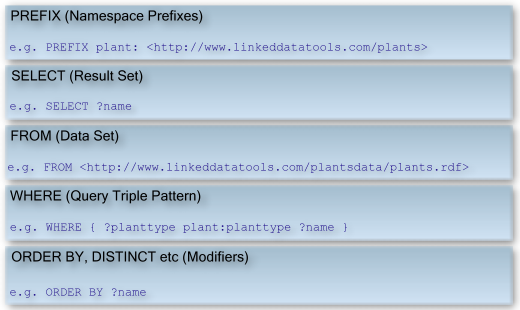
<plants:similarlyPopularTo rdf:resource="https://www.linkeddatatools.com/plants#orchid"/>

</rdf:Description>

</rdf:RDF>

**SPARQL(SPARQL PROTOCOL AND RDF QUERY LANGUAGE)**

**SYNTAX**



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

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

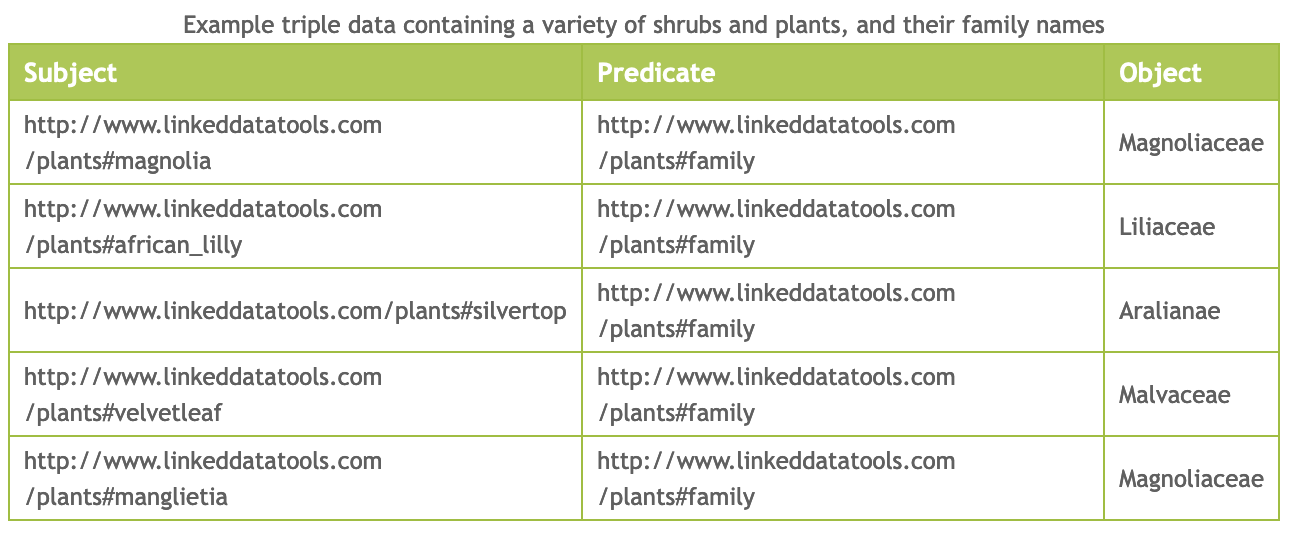
SELECT ?name WHERE {

?planttype plant:planttype ?name.

}

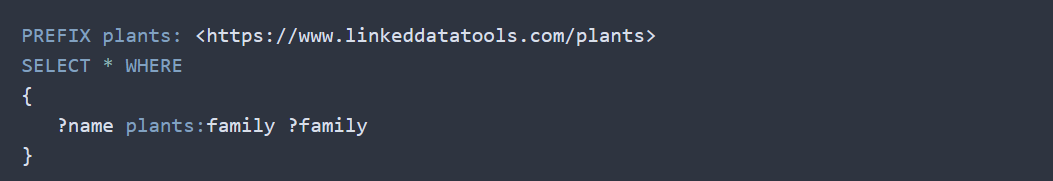
ORDER BY ?name

**TRIPPLE REPRESENTATION**



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

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.



 As we have stated two variables, **?name** and **?family**, the query will return both these declared query variables in our result set, according to the subjects, predicates or objects they’re mapped to.

**Select Only “Magnoliaceae” Family**

