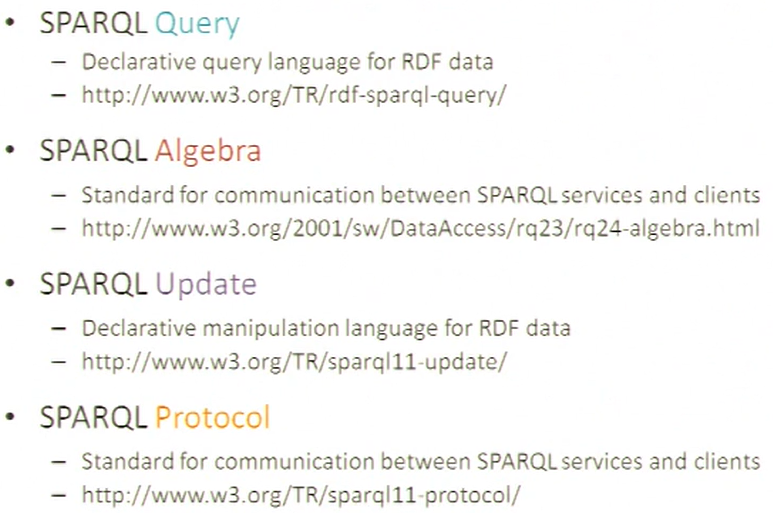
<https://euclid-project.eu/modules/chapter2.html>

CHAPTER 2: Querying Linked Data

SPARQL (Protocol And RDF Query Language): is a language for formulating *queries* through which information can be retrieved from dataset

# Part I: Introduction to SPARQL

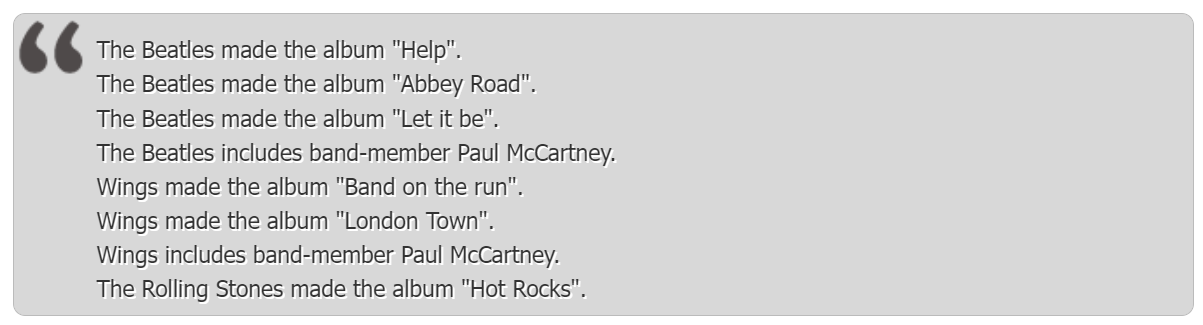
<https://www.w3.org/TR/sparql11-query/#introduction>



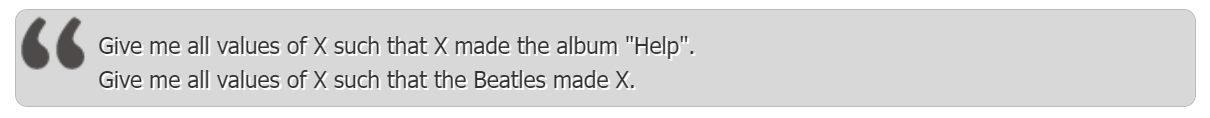
SPARQL is a query language, declare what you want to search

SPARQL algebra: related to the relation algebra

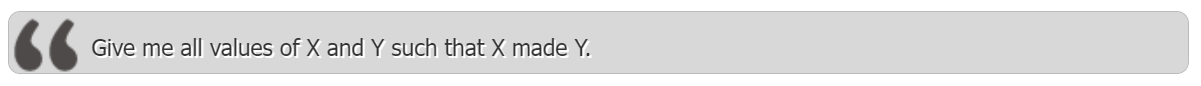
RDF dataset with statements containing the following infomration:



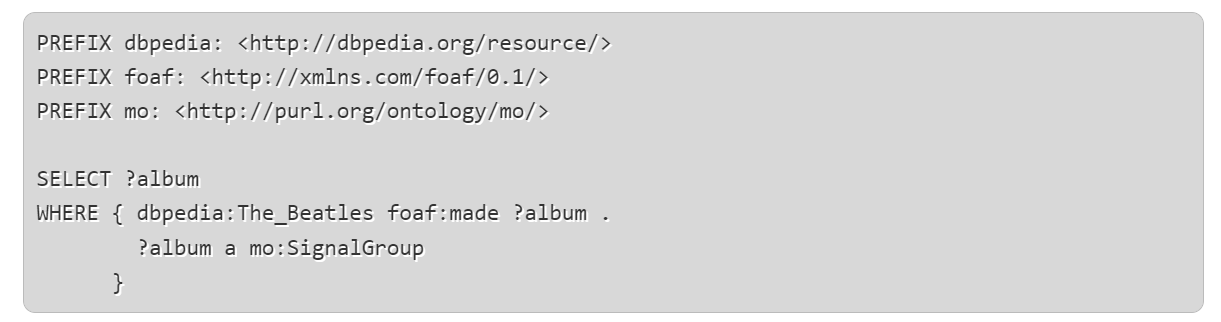
If the user asks *who made the album “Help”?* or *Which albums did the Beatles make?* Is a WH-question (who, what, where…). The SPARQL counterparts to these RDF triples that contain variables are



Or, by using more than one variable



SPARQL for *Which albums did the Beatles make?*



PREFIX statement: define abbreviations for namespaces

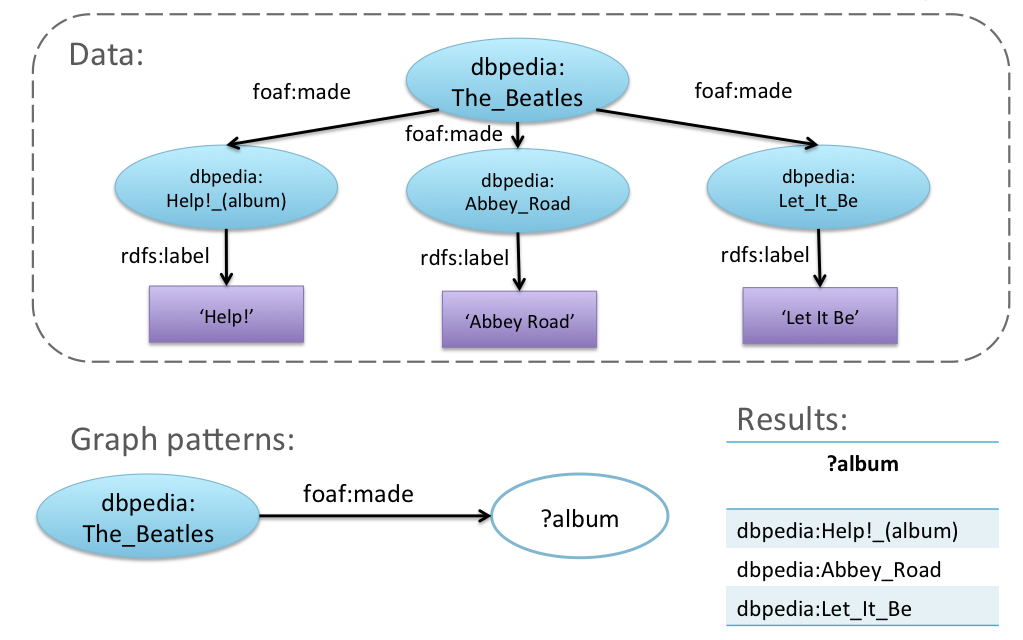
SELECT: where actually the query starts. It contains a variable (mark with ? before)

WHERE: is the remainder of the query, contains a list of RDF *triple patterns*, like RDF triples but include variables

In the example the WHERE

* first line: match resources made by the Beatles
* second line: their resources belong to a class mo:SignalGroup. Distinguish albums (that are a “signal groups” from their constituent tracks, that are also encoded as resources made by the Beatles)

The response to a query is computed by a process known as ***graph matching***



## SPARQL terminology:

**RDF triple**: statement of the form subject--predicate--object expressed in one of the RDF formalism

**RDF triple pattern**: same as RDF triple except that any or all of its three constituents may be replaced by a variable

**RDF graph**: set of RDF triples. Set of vertices (nodes) linked by edges. The subject and object are vertices, the predicate is an edge that links them by pointing from subject to object. A RDF graph can be described as a *directed labelled multigraph*.

* The edges are direction (can not switch subject and object without changing the statement)
* The edges are named (by the predicate identifier)
* There can be multiple edges linking two vertices

**RDF dataset**: set of RDF triples comprising a *default* RDF graph (unmaed) and zero or more *named* RDF graph

**Graph pattern**: conjunction of RDF triple patterns. Same as an RDF graph

## Types of query

**ASK**:

Query to test whether there are any resources in the dataset matching the search pattern, the response is either true or false

**SELECT**:

Return a table in which columns represent variables and rows represent variable bindings matching the search pattern

**CONSTRUCT**:

Return a RDF graph

**DESCRIBE**:

Returns an RDF graph, extracted from the dataset which provided all available information about a resource

## Queries using ASK

*Is Paul McCartney a member of the Beatles*?

PREFIX dbpedia: <http://dbpedia.org/resource/>

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

ASK

WHERE { dbpedia:The\_Beatles dbo:formerBandMember dbpedia:Paul\_McCartney }

* The PREFIX statements are not essential in the query, but useful for abbreviating RDF triples or patterns. !!! do not put a full stop after the PREFIX statement, or will give error
* ASK: where the query properly begins
* WHERE: introduce a graph pattern
* PREFIX, ASK, WHERE are not case sensitive

*Are there any X such that X is a member of the Beatles?*

PREFIX dbpedia: <http://dbpedia.org/resource/>

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

PREFIX mo: <http://purl.org/ontology/mo/>

ASK

WHERE { dbpedia:The\_Beatles dbo:formerBandMember ?person }

↑ this will return a boolean (it is true) because we used ASK

## Queries using SELECT

PREFIX dbr: <http://dbpedia.org/resource/>

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?person

WHERE { dbr:The\_Beatles dbo:formerBandMember ?person }

↑ this will return the names of the former band member

PREFIX dbpedia: <http://dbpedia.org/resource/>

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

PREFIX mo: <http://purl.org/ontology/mo/>

SELECT ?album

WHERE { dbpedia:The\_Beatles foaf:made ?album . ?album a mo:SignalGroup }

* PREFIX statements are optional
* SELECT: where the query proper begins, you list all the variables that you would like to see tabulated in the response. Or you can just put an asterisk \* meaning that all variables should be tabulated
* WHERE: introduces a graph pattern including one or more variables
* Layout is free provided that terms are separated by white space

## Ordering the rows in the query result

For some queries you want the results to be presented in a particular order. This can be done by using the keywords ORDER BY

PREFIX dbpedia: <http://dbpedia.org/resource/>

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

PREFIX mo: <http://purl.org/ontology/mo/>

PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT \*

WHERE { dbpedia:The\_Beatles foaf:made ?album .

?album a mo:SignalGroup .

?album dc:title ?title

}

ORDER BY ?title

Row of the results should be presented in alphabetical order of the values in the ?title colum

SELECT \* means that *all* the variables in the WHERE clause have to be tabulated

Can also use ORDER BY DESC (?title) which will present the rows in *descending order* rather than ascending

## Returning results page by page

Maybe there are too many results at one. Could be useful if the music portal included a paging facility, allowing users to view the information in manageable portions. Can be done by using LIMIT and OFFSET

PREFIX dbpedia: <http://dbpedia.org/resource/>

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

PREFIX mo: <http://purl.org/ontology/mo/>

PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT \*

WHERE { dbpedia:The\_Beatles foaf:made ?album .

?album a mo:SignalGroup .

?album dc:title ?title

}

ORDER BY ?title

LIMIT 10 OFFSET 0

You should get the same table, but cut off after the first ten rows

The result will be the same even if you put *LIMIT 10* without specifying the OFFSET

If OFFSET what 10 you would get the next ten row segment of the table covering rows 11-20

If LIMIT is L and OFFSET is S the query will return L rows starting at S+1 and continuing up to S+L

## Using tests to filter the results

<https://euclid-project.eu/modules/chapter2.html>