

IRIs like <https://dbpedia.org/resource/Stanley\_Kubrick> . can be abbreviated with a prefix like:

@prefix dbr: <https://dbpedia.org/resource/> .

Then you can use dbr:Stanley\_Kubrick instead of <https://dbpedia.org/resource/Stanley\_Kubrick>

## SPARQL

Can formulate queries over RDF data

?person foaf:knows ex:alice .

?book dbo:author ?author .

?s ?p ?o .

Example of a very simple SPARQL query:

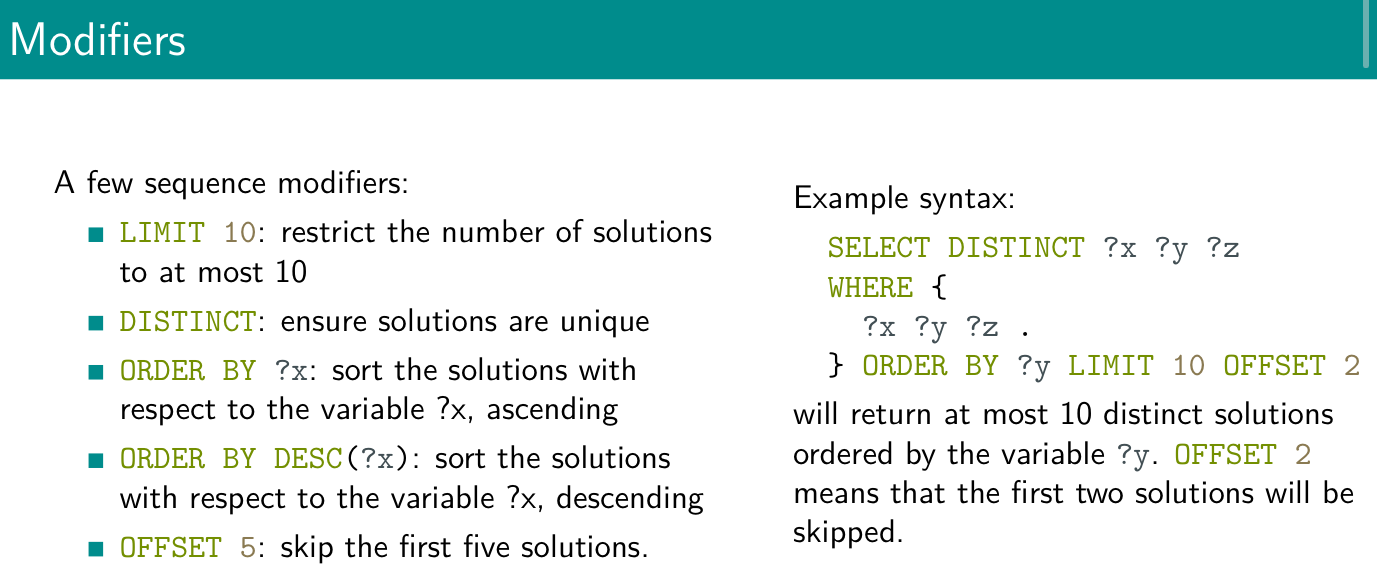
SELECT ?s ?o WHERE { ?s ?p ?o . }

This query on a RDF graph will

1. Match every triple in the graph
2. Retrieve the bindings for the variables ?s and ?o

A SELECT query will return a table with one column for each variable appearing after SELECT. The solutions are unordered

LIMIT *n* (with *n* a number) limit to only one solution



## Ex.1

# 

1. select ?x ?y where { ?x foaf:knows ?y . } ⟹ ex:bob foaf:knows ex:alice . ex:alice foaf:knows ex:carol.
2. select ?x where { ?x a ex:Person . } ⟹ ex:bob a ex:Person. ~~ex:carol a ex:Person.~~
3. select ?x where { ?x rdf:type ex:Person . } ⟹ ex:bob a ex:Person. ~~ex:carol a ex:Person~~.
4. select ?x ?y where { ?x foaf:knows ?y . ?y a ex:Student .} ⟹ ex:alice foaf:knows ex:carol. ex:carol a ex:Student.
5. select ?x ?z where { ?x foaf:knows ?y . ?y a ex:Student ; foaf:name ?z } ⟹ ex:alice foaf:knows ex:carol. ex:carol a ex:Student. ex:carol foaf:name “Carol”

## Ex.2

Retrieve 100 triples.

SELECT ?s ?p ?o

WHERE {?s ?p ?o.}

LIMIT 100

Retrieve 100 distinct triples.

SELECT DISTINCT \*

WHERE {?s ?p ?o}

LIMIT 100

Retrieve 100 distinct predicates.

SELECT DISTINCT ?p

WHERE {?s ?p ?o.}

LIMIT 100

Retrieve 100 people (i.e., resources of type person).

SELECT ?person

WHERE {?person rdf:type dbo:Person.}

LIMIT 100

Retrieve predicates that have a resource of type person as subject (some of these predicates are used for later queries).

SELECT ?p

WHERE {?person rdf:type dbo:Person;

?p ?object}

LIMIT 100

Retrieve the birth names and years of 100 people

SELECT ?birthName

WHERE {?person rdf:type dbo:Person;

dbo:birthName ?birthName.}

LIMIT 100

Retrieve the birth names of 100 people born in 1984, order by name

SELECT DISTINCT ?name

WHERE {?person a dbo:Person;

dbo:birthName ?name;

dbo:birthYear "1984"^^<https://www.w3.org/2001/XMLSchema#gYear>}

ORDER BY ?name

LIMIT 100

Retrieve the length of the Nile river

SELECT ?len

WHERE {dbr:Nile dbo:length ?len.}

Retrieve trumpet players that were also bandleaders.

SELECT ?artist

WHERE {?artist dbo:instrument dbr:Trumpet;

dbo:occupation dbr:Bandleader.}

LIMIT 100

Retrieve movies that were written, directed and produced by the same person

SELECT ?subject ?film

WHERE {

?film dbo:writer ?subject;

dbo:director ?subject;

dbo:producer ?subject.

?film rdf:type dbo:Film.

}

LIMIT 100

## Ex.3

Retrieve the name of writers and, if available, their birth year

SELECT ?s ?year

WHERE {?s rdf:type dbo:Writer.

OPTIONAL {?s dbo:birthYear ?year.}

}

LIMIT 100

The same as the previous query, but, if available, retrieve their birth place

SELECT ?s ?year ?place

WHERE {?s rdf:type dbo:Writer.

OPTIONAL {?s dbo:birthYear ?year.}

OPTIONAL {?s dbo:birthPlace ?place.}

}

LIMIT 100

Retrieve movies that were written, directed and produced by the same person, if available retrieve their box-office gross

SELECT ?s ?movie ?boxoffice

WHERE {?s dbo:writer ?movie;

dbo:producer ?movie;

dbo:director ?movie.

OPTIONAL { ?movie dbo:gross ?boxoffice.}

}

LIMIT 100

# Union

SELECT ---

WHERE {

{ --- }

UNION

{ --- }

}

## Ex.4

Retrieve movies that were written, directed or produced by charlie Kaufman

SELECT ?movie

WHERE {

{?movie dbo:director dbr:Charlie\_Kaufman}

UNION

{?movie dbo:producer dbr:Charlie\_Kaufman}

UNION

{?movie dbr:writer dbr:Charlie\_Kaufman}

}

LIMIT 100

Retrieve people that were born in Milan or in Rome

SELECT ?s

WHERE {

{?s dbo:birthPlace dbr:Milan}

UNION

{?s dbo:birthPlace dbr:Rome}

}

LIMIT 100

Same as the previous query, but also retrieve the occupation for people born in Milan

SELECT ?s ?occupation

WHERE {

{?s dbo:birthPlace dbr:Milan;

dbo:occupation ?occupation.}

UNION

{?s dbo:birthPlace dbr:Rome}

}

LIMIT 100

Same as the previous query but retrieve the occupation for the people born in Milan only if available

SELECT ?s ?occupation

WHERE {

{ ?s dbo:birthPlace dbr:Milan.

OPTIONAL { ?s dbo:occupation ?occupation.} }

UNION

{?s dbo:birthPlace dbr:Rome}

}

LIMIT 100

# Property paths

| The name of people known by bob | ex:bob foaf:knows**/**foaf:name ?name | Avoid to declare one variable |
| --- | --- | --- |
| The name of people known by someone known by bob | ex:bob foaf:knows/foaf:knows/foaf:name ?name |  |
| People that bob knows **or** that bob works with | ex:bob foaf:knows**|**ex:worksWith ?x | Same subject and object, only change the predicate |
| Ancestor of alice | ex:alice ex:hasParent**+** ?x | Like looping with the same predicate |
| Ancestor of alice, including alice | ex:alice ex:hasParent**\*** ?x |  |
| People known by bob, or that names of people known by bob | ex:bob foaf:knows**/**foaf:name? **?y** |  |

## Ex.5

Retrieve movies that were written, directed or produced by Charlie Kaufman

SELECT ?movie

WHERE {?movie dbo:director|dbo:written|dbo:produced dbr:Charlie\_Kaufman}

Retrieve people that were born or that live in Milan

SELECT ?s

WHERE {?s dbo:birthPlace|dbo:lives dbr:Milan}

LIMIT 100

Retrieve people that have written or directed a movie

SELECT ?s

WHERE {?movie rdf:type dbo:Film;

dbo:written|dbo:director ?s}

LIMIT 100

Retrieve the names of notable works of Elena Ferrante

SELECT ?s

WHERE {dbr:Elena\_Ferrante dbo:notableWork ?s}

Retrieve the grandparents of Elizabeth II

SELECT ?s

WHERE {dbr:Elizabeth\_II dbo:parent/dbo:parent ?s}

LIMIT 100

Retrieve all ancestor of Elizabeth II

SELECT ?s

WHERE {dbr:Elizabeth\_II dbo:parent+ ?s}

LIMIT 100

Retrieve the parents, the grandparents, the great-grandparents and the great-great-grandparents of Elizabeth II

SELECT ?s

WHERE {dbr:Elizabeth\_II dbo:parent/dbo:parent?/dbo:parent?/dbo:parent ?s}

LIMIT 100

# Construct, ask, describe

SPARQL have four query forms

1. SELECT: returns all, or a subset of, the variables bound in a query pattern match
2. CONSTRUCT: returns an RDF graph constructed by substituting variables in a set of triple templates
3. ASK: returns a boolean indicating whether a query pattern matches or not
4. DESCRIBE: returns an RDF graph that describes the resources found

## Ex.6

Was Natalie Portman born in (some city) the US?

ASK {dbr:Natalie\_Portman dbo:birthPlace ?x.

?x dbo:country dbr:United\_States.}

Is Rome the capital of Italy?

ASK {dbr:Italy dbo:capital dbr:Rome}

Describe Rome

DESCRIBE dbr:Rome

Describe 20 people born in Italy

DESCRIBE ?s

WHERE {?s dbo:bornIn dbr:Italy}

LIMIT 20

Construct the graph of the ancestor of dbo:Elizabeth\_II

CONSTRUCT {?s dbo:parent ?o}

WHERE { ?s dbo:parent ?o. dbr:Elizabeth\_II dbo:parent\* ?o}

# Filter

Filter: restrict solutions to those for which the filter expression evaluates to TRUE

Syntax:

SELECT ---

WHERE { --- FILTER { (---) (---) }

Can specify a conjunction or disjunction of constraints with && or ||

Filter can also involve functions such as:

FILTER(lang(?cityName) = “en”)

FILTER (year(?date) > 1990)

NOT EXISTS: tests whether a graph pattern does not match the dataset

MINUS: operation that takes two arguments. It evaluates both its arguments and then calculates solutions in the left-hand side that are not compatible with the solutions on the right hand side

## Ex.7

Retrieve the names of 100 people born after 1984

select distinct ?n ?b

where { ?s a dbo:Person ; dbo:birthName ?n ; dbo:birthYear ?b . filter (?b > "1984"^^) } order by ?n

limit 100