Querying Data through Ontologies

* SPARQL:

=> SPARQL stands for SPARQL Protocol and RDF Query Language, which is Used to query RDF data.

-> Steps and Writing and Excuting a SPARQL Query

1 Define the Query Goal:

Identify what information you want to retrieve from the RDF data.

This can be anything from finding relation between entities.

2 Write the SPARQL Query:

- Select Clause: Specify the variable you want to reterive

- Where Clause: Define the pattern to match in the RDF data.

Includes specifying the triple

pattern that describe the relationship and properties.

3 Execute the Query :

Run the query against an RDF dataset using a SPARQL endpoint or a local SPARQL processor.

4 Process the Result :

Analyze the results returned by the query.

Result can be presented in Various formats such as tabular, JSON or XML.

=> Example : Library Management System.

RDF dataset :

<http://ex.org/book1><http://ex.org author><http://ex.org/ABC>.
<http://ex.org/book1> <http://ex.org/title> "Introduction".
<http://ex.org/ABC> <http:ex.org/name> "ABC".

Here, We want to Find all the Book which is writtenby ABC Author.

-> SPARQL Query:

```
PREFIX ex: <http://ex.org/>

SELECT ?book ?title
WHERE
 {
   ?book ex:author ex:ABC.
   ?book ex:title ?title.
 }
```

-> Result:

Book	title
<http://ex.org/book1>	Introduction

* Querying RDF Data For RDFS ontologies

=> RDF Data represent Fact about individual and their relationship ~~betw~~ using triple structure.

RDFS Onto: Provides schema

For RDF data for Defining classes, properties and their complex relanship.

For RDFS Ontologies we have to use Forward-Chaining Algorithm to get the query result.

=> Algorithm : Saturation Algorithm

1 Initialize:
    'F' <— Set of Initial Facts
    'Δ0' <— Set of Initial facts

2 Repeat Until Saturation

(i) Initialize 'Δ1' as an Empty set

(ii) For each RDFS rule
    (Condition => Conclusion):
    - If the condition satisfied by facts in 'Δ0' and the conclusion is not already in 'F', add the conclusion to 'Δ1'

(iii) Update 'F' with new facts from 'Δ1'.

(iv) Set 'Δ0' to 'Δ1'

Saturation means until all the facts in RDFS ontologies can be inferred.

Ex. Library Management.

1 Define Class and Properties:
RDFS:

```
ex:Book rdf:type owl:Class;
ex:Author rdf:type owl:Class.

ex:WrittenBy rdf:type owl:Object
    Property;
    rdfs:domain ex:Book;
    rdfs:range ex:Author.

ex:Author owl:disjointWith
    ex:Book.
```

2 RDF Data

```
ex:book1 rdf:type ex:Book;
   ex:title "Introduction";
   ex:writtenBy ex:author1.

ex:author1 rdf:type ex:Author;
   ex:name "ABC"
```

3 Querying with SPARQL

```
PREFIX ex: <http://ex.org/>
PREFIX ex:<http://www.w3.org/2001/
        XMLSchema#>

SELECT ?Book ?title ?authorName
WHERE
{
  ?book rdf:type ex:Book.
  ?book ex:writtenBy ?author.
  ?book ex:title ?title
  ?author rdf:type ex:author.
  ?author ex:name ?authorName.
}
```

4 Result:

book 1 | Introduction | ABC

* DL-Lite:

=> DL-Lite is a family of Description Logics designed to balance expressive power.

It is often used for ontology-based data access.

It is use when you want to performing reasoning while dealing with large data set.

=> Steps of DL-Lite

1 Define the Ontology: Specify Concepts, role and Axioms in DL-Life.

Classes: type of entities
Roles: Relationship between entities.
Axioms: Concept and Roles Statement.

2 Create the Knowledge Base: Create RDF data.

3 Perform Reasoning: Apply DL-Lite reasoning to infer new knowledge based on the ontology

4 Query the Data: Use SPARQL to extract information.

Ex. Library Management

1 Define Ontology:

Classes: Book, Author, Person
Role: WrittenBy
Axioms:
'Book ⊆ Resource' (Book are Resourc
'Author ⊆ Person'
'WrittenBy : Book → Author'

2 RDF data:

<http://ex.org/book1><http://ex.org/title> "Introduction".
<http://ex.org/author1><http://ex.org/name> "ABC".
<http://ex.org/book1><http://ex.org/writtenBy><http://ex.org/author1>.

3 Performing Reasoning:

- Book instances are also instances of Resource
- Author instances are also instances of Person.

4 SPARQL Query:

Date : / /

```
PREFIX ex:<http://ex.org/>

SELECT ?book ?author
WHERE
{
  ?book ex:WrittenBy ?author.
  ?author a ex:Author.
}
```