SPARQL

SPARQL END POINT → SERVER → CLIENT → User Interface

SPARQL Variables
e.g ?title ?author ?date

① Query for variables via Select

SELECT ?title ?author ?date

SPARQL is based on
1) RDF Turtle Serialization
2) basic graph pattern matching

Graph pattern (Triple Pattern) is a RDF Triple that contains variables at any arbitrary place (S, P, O).

Graph Pattern = Turtle + Variables

→ look for books and their authors (via property dbo:author

?book   dbo:author ?author .

Conjuctive queries

⟹ Find books, their authors and their literary genre.

```
[?book]   dbo:author      ?author .
[?book]   dbo:literaryGenre   ?genre .
```

the same book

- Given a specific book URI, find its author(s), the birthplace(s) of the author(s), including the number of population of the birthplace(s).

dbr: Brave_New_World   dbo: author   ? author.

? author   dbo: birthplace   ? birthplace.

? birthplace   dbo: populationTotal   ? population

SPARQL QUERY

search for all the authors and the titles of their notable works

specific namespace —

```
PREFIX    rdf : <http: // www.w3.org /1999/02/22-rdf-syntax-ns
PREFIX    rdfs : <http: //ww.w3.org /2001 /01 /rdf-schema>
PREFIX    dbo: <http: // dbpedia.org /ontology/ >
```

SELECT ? author-name ? title — specifies output variable

FROM <http://dbpedia.org/> —specifies graphs to be queried.

WHERE {

  ? author   rdf:type   dbo: Writer .   → specifies graph patterns to be matched

  ? author   rdfs: label   ? author-name .

  ? author   dbo: notableWork   ?work .

}   ? work   rdfs : label ?title .

```
PREFIX    ab: <http://learnings sparql.com/ns/addressbooks>
SELECT    ?first    ?last    ?email
WHERE
    {
        ?person  ab: homeTel  "(229) 276-5135";
        ?person  ab: firstName   ?first .
        ?person  ab: lastName    ?last .
        ?person  ab: email      ?email .

        }
```

shortcut

```
    WHERE
        {  ?person   ab: homeTel  "(229) 276-5135";
                     ab: firstname    ?first;
                     ab: lastname    ?last;
                     ab: email ..

            }
```

```
PREFIX   ab : <http://learningsparql.com/ns/
                                        addressbook:

SELECT   ?first  ?last  ?workTel

    WHERE
    {
        ?s ab:FirstName    ?First ;
any          ab:lastName   ?last ;
variable     ab:workTel    ?workTel .


    }
```

```
WHERE

    {   ?s      ab:firstName    ?first ;
            ab: last Name      ?last ;
        OPTIONAL
        {  ?s    ab:workTel    ?workTel .
        }
    }
```

```
        SELECT ?first    ?last  ?workTel ?email
    WHERE
    {   ?s    ab: firstName     ?first ;
          .    ab :lastName      ?last ;
        OPTIONAL
separate    { ?s    ab:workTel    ?workTel.
←combine[     ?s  ab:email       ?email .
            }
    }
```

```
SELECT ? . .....

  WHERE
  { ?s   ab: firstName  ?first ;
         ab: lastName   ?last ;

      OPTIONAL

      {   ?s   ab: workTel ?worktel
      }

  OPTIONAL
         { ?s ab: email ?email.
         }
  }
```

@prefix ab: <http://learningsparql.com/ns/addressbook>
@prefix d: <http://learning sparql.com/ns/data#>.

PREFIX ab: <http://learnings parql.com/ns/addressbook#>

SELECT ?last ?first ?coursename

WHERE

 { ?s   ab:firstName   ?first ;
    ab:lastName   ?last ;
    ab: takingCourse   ?course .

  ?course   ab:CourseTitle   ?courseName .

 }

Unique

SELECT DISTINCT ?first ?last
WHERE

 { ?s   ab:takingCourse   ?class ;
    ab: firstName   ?first ;
    ab: lastName   ?last .

 }

for selecting all

SELECT *
WHERE
{
 ?s ?p ?o .

}

Ans:

NOTEXIST / **FILTER**

Another solution

SELECT   ?first ? last

WHERE

{ ?s ab: firstName ?first ;
      ab: lastName ? last .
   NOTEXIST { ?s ab: workTel ? workNum }
                    ↓
}   no workTel property

WHERE

{ ?s ab: firstName ?first ;
       ab: lastName ?last.

   MINUS { ?s ab: workTel
                        ?workNum }

Another solution

WHERE

{ ?s ab : firstName ?first ;
       ab : lastName   ?last .

has
workTel ⟵ OPTIONAL { ?s ab : workTel ?workNum .}
then   FILTER (! bound (?workNum)) |
store id   ↓
a workNum }   ↓   if workNum
                       not   has a value
                                  bound will be
                                     True

Ans.

Combine results — UNION

SELECT *

WHERE

{

   { ?person ab:firstName ?first; ab:lastName ?last .}

    UNION

   { ?course ab:courseTitle ?courseName .}

}

SELECT ?first ?last

WHERE

{

  { ?s ab:firstName ?first;
       ab:lastName ?last;
       ab:takingCourse ?course.

  ?course ab:courseTitle "Modeling Data with OWL".

 }

UNION

{ ?s ab:firstName ?first;
     ab:lastName ?last;
     ab:takingCourse ?course.

 ?course ab:courseTitle "Using SPARQL with non-RDF Data".
}

LIMIT   OFFSET

SELECT *
  WHERE
{ ?s ?p ?o }
  LIMIT 2

PREFIX rdfs : < http://ww'w.w8.org/2000/01/rdf-schema#>
  SELECT ?label
    WHERE
    { ?s rdfs: label ?label . }
  LIMIT 2

  OFFSET → skip a certain no of results

    SELECT *
      WHERE
      { ?s ?p ?o}
      OFFSET 3
      LIMIT 2