TẠP CHÍ KHOA HỌC, Đại học Huế, Số 65, 2011

CÁC NGÔN NGỮ TRUY VẤN RDF: ĐÁNH GIÁ TỔNG QUAN VÀ SO SÁNH CÁC ĐẶC TÍNH NGÔN NGỮ

Hoàng Nguyễn Tuấn Minh
Trường Đại học Khoa học, Đại học Huế
Hoàng Hữu Hạnh
Đại học Huế

TÓM TẮT

Khung ứng dụng RDF được xem là công cụ để mô tả thông tin về các tài nguyên cho Web ngữ nghĩa một cách linh động. RDF có thể được sử dụng để biểu diễn thông tin cá nhân, mạng xã hội, siêu dữ liệu về tài nguyên số cũng như để cung cấp một phương tiện tích hợp các nguồn thông tin hỗn tạp. Các ngôn ngữ truy vấn RDF như SPARQL có thể được sử dụng để tạo các truy vấn trên các nguồn dữ liệu đa dạng. Bài báo trình bày các điểm mạnh của ngôn ngữ SPARQL, trong ngữ cảnh phân tích và đánh giá với các ngôn ngữ truy vấn RDF khác. Kết quả đánh giá được tổng hợp để làm rõ hơn các ưu điểm của ngôn ngữ SPARQL trong truy vấn siêu dữ liệu trên Web ngữ nghĩa.

1. Giới thiệu

Các ngôn ngữ truy vấn RDF - Resource Description Framework- có thể được phân nhóm thành ba "dòng" khác biệt theo các khía cạnh như mô hình dữ liệu, tính biểu trưng, hỗ trợ thông tin lượt đồ, và các kiểu truy vấn. Cơ bản trong 3 dòng này là SPARQL [3]. Dòng này có nguồn gốc từ nguôn ngữ SquishQL, sau đó phát triển thành RDQL [11] và cuối cùng được mở rộng thành ngôn ngữ SPARQL. Những ngôn ngữ này xem RDF như là dữ liệu bộ ba mà không quan tâm đến lượt đồ hay thông tin về ontology trừ khi điều đó được nêu rõ trong nguồn RDF. SPARQL hiện là khuyến nghị của tổ chức W3C (một dạng của "chuẩn") cho "ngôn ngữ truy vấn cho RDF". Đặc biệt, SPARQL cho phép: 1) Trích rút các đồ thị RDF con, 2) Kiến tạo một đồ thị RDF mới sử dụng dữ liệu đầu vào là đồ thị RDF truy vấn, 3) Trả về "các mô tả" của các tài nguyên mà phù hợp với dạng truy vấn, 4) Chỉ định các bộ ba hay dạng độ thị truy vấn tuỳ chọn, và 5) kiểm tra sự tồn tại của các bộ.

Một dòng các ngôn ngữ RDF, gọi là "dòng RQL", gồm ngôn ngữ RQL [9] và mở rộng của nó như SeRQL [1]. Điểm chung của dòng này là hỗ trợ kết hợp truy vấn dữ liệu và lượt đồ. Mô hình dữ liệu RDF được sử dụng hơi chệch với mô hình dữ liệu chuẩn của RDF và RDFS (RDF Schema), do đó làm mất đi các chu trình trong phân cấp bao hàm và các yêu cầu về cả miền xác định và miền giá trị định nghĩa cho mỗi thuộc tính. Bản thân RQL có khá nhiều đặc tính và tuỳ chọn trong các cấu trúc ngữ nghĩa.

Điều này tạo ra một ngôn ngữ phức tạp dù là mạnh và điều đó là không có tính biểu diễn hơn các ngôn ngữ truy vấn RDF khác, đặc biệt là các ngôn ngữ dòng SPARQL.

Bên cạnh đó, còn có một số loại ngôn ngữ truy vấn RDF khác sử dụng các mô hình khác, như là sử dụng các luật, hoặc như các ngôn ngữ suy diễn như TRIPLE [12] và Xcerpt. Ngôn ngữ Xcerpt đáng được lưu ý vì nó kết hợp truy vấn trên Web chuẩn (HTML/XML), với truy vấn trên Web ngữ nghĩa (chẳng hạn như RDF và TopicMaps) và cũng cho phép đặc tả truy vấn không đầy đủ hoặc dựa trên dạng (pattern-based).

2. Ngôn ngữ truy vấn SPARQL

2.1. Giới thiệu

SPARQL được phát triển bởi nhóm RDF Data Access Working Group - một phần trong hoạt động của Semantic Web và đã được W3C - tổ chức chịu trách nhiệm xây dựng, quản lý đưa ra các chuẩn liên quan đến World Wide Web - khuyến nghị vào năm 2008. Định dạng thông thường của một truy vấn SPARQL là:

`PREFIX`	Chỉ định tên cho một URI
`SELECT`	Trả về tất cả hoặc vài giá trị biến theo mệnh đề WHERE
`CONSTRUCT`	Trả về một đồ thị RDF với các biến liên quan
`DESCRIBE`	Trả về một "mô tả" của tài nguyên tìm được
`ASK`	Trả về kết quả tìm một mẫu đồ thị có hay không
`WHERE`	danh sách, tức là kết nối các mẫu (đồ thị) truy vấn
`OPTIONAL`	danh sách, tức là kết nối các mẫu (đồ thị) truy vấn tuỳ chọn
`AND`	biểu thức logic (để lọc các giá trị)

Một câu truy vấn chọn dữ liệu SPARQL-SELECT bao gồm 2 mệnh đề chính, mệnh đề *SELECT* và mệnh đề *WHERE* cùng các thành phần khác. Mệnh đề *SELECT* định danh các biến mà ứng dụng quan tâm và mệnh đề WHERE bao gồm các *mẫu bộ ba (triple pattern)*, các thành phần khác sẽ được đề cập đến trong các phần tiếp theo. Cú pháp tổng quát của SPARQL-SELECT được liệt kê như sau:

```
PREFIX ns: <namespaceURI>
PREFIX : <.>
SELECT variables
[FROM <dataURI>]
[FROM NAMED <dataURI>]
WHERE { constraints  [FILTER]  [OPTIONAL] }
[ORDER BY variables] [OFFSET/LIMIT n] [DISTINCT]
```

Dữ liệu trong RDF được mô tả theo dạng các bộ ba. Tập hợp các bộ ba RDF tạo ra một đồ thị, gọi là đồ thị RDF. Ngôn ngữ truy vấn SPARQL lấy thông tin từ các đồ thị RDF, nó cung cấp các tính năng sau:

- Chiết xuất thông tin dưới dạng các URI, các node trắng (*blank node*), các *plain* literal và *typed literal.*
- Chiết xuất các đồ thị con RDF.
- Xây dựng các đồ thị RDF mới dựa trên thông tin của các đồ thị truy vấn.

2.2. Các khái niệm cơ bản

Định nghĩa 2.1: Một *mẫu bộ ba* (*triple pattern)* là một *bộ ba RDF (RDF triple)* nhưng mỗi thành phần (subject, predicate hay object) đều có thể là một biến truy vấn.

Định nghĩa 2.2: Một *mẫu đồ thị cơ sở* (basic graph pattern) là một tập các mẫu bộ ba.

Ngôn ngữ SPARQL dựa trên cơ sở đối sánh các *mẫu đồ thị* (*graph pattern). Mẫu đồ thị* đơn giản nhất là các *mẫu bộ ba*. Một định danh URI được đặt trong 1 cặp dấu '<>'. Một giá trị *literal* được đặt trong cặp dấu (“”).

SPARQL cung cấp một cơ chế viết tắt. Tiếp đầu ngữ có thể được định nghĩa và một QName sẽ cung cấp một dạng viết làm cho URI có thể ngắn gọn. Một *node trắng* có thể xuất hiện trong một *mẫu truy vấn* (*query pattern)*. Nó giữ vai trò như một biến, mặc dù nó không được đề cập trong kết quả của câu truy vấn hay bất kỳ nơi nào ở trong mẫu đồ thị.

2.3. Các mẫu đồ thị (Graph Patterns)

2.3.1. Mẫu đồ thị cơ sở (Basic Graph Pattern)

Một mẫu đồ thị cơ sở (*basic graph pattern)* là một tập các mẫu bộ ba. Đối sánh mẫu đồ thị được định nghĩa dựa trên việc kết hợp các kết quả phù hợp từ việc đối sánh các *mẫu đồ thị cơ sở.*

Định nghĩa 2.3: Một mẫu lời giải (*pattern solution) S* của một phần đồ thị *GP* trên đồ thị *G* là bất kì phép thế *S* nào sao cho *S(GP)* là đồ thị con của *G*.

2.3.2. Mẫu đồ thị nhóm (Group Graph Pattern)

Trong chuỗi truy vấn SPARQL, một *mẫu đồ thị nhóm* được phân định bởi một cặp ngoặc {}. Ví dụ mẫu đồ thị của truy vấn này là một mẫu đồ thị nhóm chỉ có một mẫu đồ thị cơ sở. Mẫu đồ thị cơ sở này chứa hai mẫu bộ ba.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  {
          ?x foaf:name ?name .
          ?x foaf:mbox ?mbox .
       }
```

2.3.3. Ràng buộc giá trị

Với SPARQL ta có thêm vào truy vấn các ràng buộc giá trị để đạt được kết quả mong muốn với FILTER.

Ví dụ: Truy vấn:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x ns:price ?price .
        FILTER (?price < 30) .
        ?x dc:title ?title . }
```

Nhằm liệt kê các kết quả với giá bé hơn 30.

2.3.4. Mẫu đồ thị tùy chọn (Optional Graph Pattern)

Qua những ví dụ trên nhận thấy rằng mỗi lời giải của câu truy vấn phải hoàn toàn phù hợp với các thành phần của mẫu truy vấn. Nhưng với việc thêm vào từ khóa OPTIONAL ta có thể có nhiều hơn kết quả, mặc dù mỗi lời giải pháp có thể chỉ thoả một phần của truy vấn. Nhiệm vụ của OPTIONAL trong mẫu truy vấn là chọn tất cả các giá trị có thể có của mẫu truy vấn.

Ví dụ: Với dữ liệu:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .
:book1 dc:title "SPARQL Tutorial" .
:book1 ns:price 42 .
:book2 dc:title "The Semantic Web" .
:book2 ns:price 23 .
```

Câu truy vấn:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE
          { ?x dc:title ?title .
 OPTIONAL { ?x ns:price ?price .
            FILTER (?price < 30 )}}
```

Cho kết quả là:

title	price
"SPARQL Tutorial"	
"The Semantic Web”	23

2.3.5. Mẫu đồ thị kết hợp (Union Graph Pattern)

SPARQL cung cấp một cách thức kết hợp các mẫu đồ thị lại với nhau bằng cách dùng mệnh đề UNION.

Ví dụ: Truy vấn sử dụng UNION

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE  { { ?book dc10:title  ?title }
   UNION { ?book dc11:title  ?title } }
```

Trường hợp sử dụng truy vấn này là các thông tin được biểu diễn bởi các lượt đồ khác nhau theo phiên bản.

2.4. RDF Dataset

Mô hình dữ liệu RDF thể hiện thông tin về các đồ thị bao gồm các bộ ba với các subject, predicate và object. Các kho dữ liệu RDF chứa nhiều các đồ thị RDF và thông tin bản ghi về mỗi đồ thị và nó cho phép một ứng dụng để thực hiện các truy vấn có liên quan đến thông tin từ nhiều hơn một đồ thị.

Định nghĩa 2.4: Một RDF Datasset đại diện cho một tập hợp các đồ thị. RDF Dataset bao gồm một đồ thị *mặc định (defaut graph)* và hoặc không có hoặc có nhiều đồ thị được đặt tên *(named graph)*. Trong đó các đồ thị được đặt tên được định danh bởi một URI.

Đồ thị được sử dụng cho việc đối sánh một mẫu đồ thị cơ sở là *đồ thị đang hoạt động* (active graph). Ở trong các phần trước tất cả các truy vấn đều được thực thi đối với một đơn đồ thị, điều đó có nghĩa là đồ thị mặc định của RDF Dataset chính là đồ thị đang hoạt động. Từ khóa GRAPH dùng để chỉ định đồ thị đang hoạt động là một trong các đồ thị được đặt tên của Dataset nằm trong truy vấn.

Định nghĩa của RDF Dataset không hạn chế mối quan hệ của đồ thị được đặt tên và đồ thị mặc định. Thông tin có thể được lặp đi lặp lại trong các đồ thị khác nhau. Có hai chú ý ta cần xem xét:

- Để có thông tin trong đồ thị mặc định phải bao gồm các thông tin gốc trên các đồ thị được đặt tên;
- Nên bao gồm các thông tin trên đồ thị đặt tên trong đồ thị mặc định.

Một truy vấn SPARQL có thể đặc tả các dataset sẽ được sử dụng cho việc đối sánh bằng cách sử dụng mệnh đề FROM và mệnh đề FROM NAMED. Từ khóa FROM và FROM NAMED cho phép một truy vấn một RDF dataset bằng cách tham chiếu URI. Mỗi URI được sử dụng để cung cấp cho một đồ thị được đặt tên trong RDF dataset. Nếu không có mệnh đề FROM nhưng có một hay nhiều mệnh đề FROM NAMED thì có

nghĩa dataset đó chứa một mệnh đề đồ thị mặc định rỗng.

Khi truy vấn một tập hợp các đồ thị, từ khóa GRAPH được sử dụng để đối sánh các mẫu bộ ba trên các đồ thị được đặt tên. GRAPH có thể cung cấp một URI để lựa chọn một đồ thị hoặc hoặc sử dụng một biến chỉ các URI trong truy vấn RDF dataset.

Ví dụ: Xét Dataset dưới đây với hai đồ thị có tên:

```
# Named graph: http://example.org/foaf/aliceFoaf
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
_:a  foaf:name     "Alice" .
_:a  foaf:mbox     <mailto:alice@work.example> .
_:a  foaf:knows    _:b .
_:b  foaf:name     "Bob" .
_:b  foaf:mbox     <mailto:bob@work.example> .
_:b  foaf:nick     "Bobby" .
_:b  rdfs:seeAlso  <http://example.org/foaf/bobFoaf> .
<http://example.org/foaf/bobFoaf> rdf:type
foaf:PersonalProfileDocument.
```

```
# Named graph: http://example.org/foaf/bobFoaf
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
_:z  foaf:mbox     <mailto:bob@work.example> .
_:z  rdfs:seeAlso  <http://example.org/foaf/bobFoaf> .
_:z  foaf:nick     "Robert" .
<http://example.org/foaf/bobFoaf> rdf:type
foaf:PersonalProfileDocument .
```

Câu truy vấn:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?src ?bobNick
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE {
    GRAPH ?src
    { ?x foaf:mbox <mailto:bob@work.example> .
      ?x foaf:nick ?bobNick
    }
  }
```

Cho kết quả truy vấn:

src	bobNick
http://example.org/foaf/aliceFoaf	"Bobby"
http://example.org/foaf/bobFoaf	"Robert"

Bây giờ xét một truy vấn khác với mục đích là tìm hiểu các thông tin với các liên kết thông tin mô tả ở trong hai đồ thị khác nhau:

```
PREFIX data:  <http://example.org/foaf/>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?mbox ?nick ?ppd
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE {
  GRAPH data:aliceFoaf {
    ?alice foaf:mbox <mailto:alice@work.example> ;
           foaf:knows ?whom .
    ?whom  foaf:mbox ?mbox ;
           rdfs:seeAlso ?ppd .
    ?ppd  a foaf:PersonalProfileDocument .
  } .
  GRAPH ?ppd {
      ?w foaf:mbox ?mbox ;
         foaf:nick ?nick
  }
}
```

Trong ví dụ này là tìm thông tin của một người qua việc lưu trữ thông tin và mối quan hệ giữa họ với nhau, và kết quả truy vấn thu được là:

mbox	nick	ppd
<mailto:bob@work.example>	"Robert"	<http://example.org/foaf/bobFoaf>

3. So sánh SPARQL với các ngôn ngữ truy vấn RDF trên Web ngữ nghĩa

3.1. Một số ngôn ngữ truy vấn RDF khác

3.1.1. RQL

RQL là một ngôn ngữ truy vấn cho RDF và RDFS. Ý tưởng là chúng ta xem một hoặc nhiều mô hình và lượt đồ RDF là một tập đồ thị kết nối. RQL cung cấp những tính

năng để điều hướng qua đồ thị đó và lựa chọn những cạnh và những nút cụ thể cho sự truy hồi. RQL hỗ trợ tổng quan hóa các biểu thức đường dẫn theo các biến trên cả các nút và các cạnh của đồ thị RDF. Sự khác biệt của RQL là dựa và khả năng kết hợp mềm mại truy vấn dữ liệu và lượt đồ bằng cách khai thác sự phân biệt các nhãn và đa phân nhóm của tài nguyên.

Một truy vấn RQL được xây dựng đặc trưng bởi 3 mệnh đề mà chúng ta thường thấy trong SQL: select, from, where.

```
select X, @P
from  {X} @P {Y}
where  Y like "Pablo"
```

Mệnh đề *Select* cho phép chỉ định một chiếu trên các kết quả truy vấn của bạn. Trong truy vấn trên, chúng ta rất quan tâm đến các biến X và @P, nhưng không quan tâm đến biến Y. Mệnh đề *From* là nơi chứa biểu thức đường dẫn. Tại đây, các biến để chỉ định vị trí trong đồ thị RDF model bằng cách chỉ định biểu thức đường dẫn. Trong ví dụ này, X và Y được giới hạn là các nút trong đồ thị, trong khi @P là các cung (cạnh) kết nối (các @ là tiền tố của biến biểu thị rằng biến chỉ được giới hạn là các thuộc tính). Vì vậy, trong cấu trúc này tương ứng với một statement, nơi mà X là subject, @P các predicate, và Y là object. Mệnh đề *Where* là tùy chọn và có thể được sử dụng để ràng buộc các giá trị của các biến bị ràng buộc trong mệnh đề *From*. Trong ví dụ, chúng tôi chỉ muốn trả lại những giá trị mà có giá trị cho Y bằng chuỗi "Pablo". Điều này tương ứng với lựa chọn tất cả các statement nơi mà "Pablo" là object.

3.1.2. SeRQL

SeRQL là ngôn ngữ truy vấn RDF cho Sesame. Nó là ngôn ngữ truy vấn và chuyển đổi có dựa trên một vài ngôn ngữ đã có sẵn như RQL, RDQL,và N3. Những mục tiêu thiết kế đầu tiên của nó là hợp nhất các công việc tốt nhất từ các ngôn ngữ truy vấn và đưa ra một cách diễn đạt ngôn ngữ truy vấn tốt hơn mà nó tập trung vào các mối quan tâm thực hành.

SeRQL hỗ trợ các biểu thức đường dẫn phổ biến như các ràng buộc boolean hay các tùy chọn phù hợp trong cả hai cấu trúc cơ bản: *select - from – where* và *construct – from – where.* Cấu trúc đầu tiên trả về kết quả quen thuộc là các ràng buộc biến/ bảng. Cấu trúc thứ hai trả về đồ thị con phù hợp. Theo đúng nghĩa, các truy vấn *construct – from –where* trong SeRQl đáp ứng thuộc tính đóng và thuộc tính trực giao vì thế nó cho phép kết hợp các truy vấn. SeRQL không an toàn khi nó cung cấp các chức năng đệ quy được xây dựng sẵn khác nhau.

SeRQL được thực thi và có giá trị trong hệ thống Sesame, một số tính năng truy vấn vẫn bị mất đi trong khi thi hành truy vấn. Đáng chú ý nhất là các hàm kết hợp (*minimun, maximun, average, count*) và việc lồng các truy vấn [7].

3.1.3. TRIPLE

TRIPLE thể hiện tốt trong cả hai dạng ngôn ngữ truy vấn và ngôn ngũ luật trong hệ thống thời gian thực. Ngôn ngữ này có nguồn gốc từ F-Logic. Các bộ ba RDF *(S, P, O)* được biểu diễn như là các biểu thức F-Logic *S[P → O]* và chúng có thể được lồng vào nhau.

Ví dụ: biểu thức *S[P1 → O1, P2 → O2[P3 → O3]]* tương ứng với ba bộ ba RDF là: *(S,P1,O1), (S,P2,O2) và (S,P3,O3)*

TRIPLE không phân biệt giữa các truy vấn với các luật mà nó chỉ đơn giản là các luật không đầu nơi mà các kết quả là các ràng buộc biến tự do trong truy vấn.

Ví dụ: *FORALL X ← (X [rdfs:label → "foo"]) @default:ln* truy vấn này sẽ trả về tất cả các tài nguyên có nhãn (label) là "foo".

Vì đầu ra là một bảng các biến và các ràng buộc có thể thực hiện được, TRIPLE không phải là có thuộc tính đóng. Cũng như thế, TRIPLE cũng cho cảm giác không an toàn vì nó cho phép các luật không an toàn ví dụ như là:

FORALL X (X[rdfs:label → "foo"] ← (a[rdfs:label → "foo"])@default:ln

Trong khi TRIPLE có tính đầy đủ và đóng cho chính mô hình dữ liệu của nó thì việc ánh xạ từ RDF đến TRIPLE không phải là không bị mất mát. Ví dụ như các node RDF ẩn danh được làm rõ. TRIPLE có khả năng đáp ứng đồng thời với nhiều mô hình RDF mà chúng được định danh thông qua hậu tố *@model.*

TRIPLE không mã hóa một ngữ nghĩa RDF cố định. Ngữ nghĩa mong muốn phải được chỉ định như là một tập các luật theo truy vấn. TRIPLE không hỗ trợ các kiểu dữ liệu và cập nhật các thành phần cơ bản thực tế cũng không thể thực hiện được.

3.1.4. RDQL

RDQL là một ngôn ngữ truy vấn cho RDF trong các mô hình Jena. Mục đích là để cung cấp một mô hình truy vấn hướng dữ liệu, sự "hướng dữ liệu" nghĩa là nó truy vấn thông tin được tổ chức trong các mô hình, không có kết luận được thực hiện. RDQL là một mở rộng của ngôn ngữ truy vấn RDF SquishQL. Các lớp của ngôn ngữ truy vấn này xem RDF như bộ ba dữ liệu, mà không có thông tin lượt đồ hay ontology, trừ khi được kèm theo rõ ràng trong các mã nguồn RDF.

RDF cung cấp một đồ thị với các cạnh có hướng – các nút là các nguồn tài nguyên hoặc literal. RDQL cung cấp một cách để xác định một mẫu đồ thị mà nó được đối sánh với đồ thị để sinh ra một tập các kết quả đối sánh. Nó trả lại một danh sách các ràng buộc - mỗi ràng buộc là tập các cặp tên- giá trị của các giá trị của các biến.

Ví dụ:

```
SELECT ?x,?y
FROM <http://example.com/sample.rdf>
WHERE (?x,<dc:name>,?y)
USING dc for <http://www.dc.com#>
```

Điều này sẽ trả lại tất cả các bộ `?x ?y` cho biết tên tài nguyên và giá trị của thuộc tính `dc:name` cho mỗi tài nguyên.

3.1.5. N3 (Notation 3)

N3 cung cấp một cú pháp dựa vào văn bản cho RDF. Vì vậy mô hình dữ liệu của N3 hợp với mô hình dữ liệu RDF. Thêm vào đó N3 cho phép chúng ta định nghĩa các luật mà có thể được biểu diễn sử dụng trong một cú pháp riêng biệt.

Ví dụ: `?y rdfs:label "foo" => ?y a:QueryResult.`

Các luật cũng có thể được sử dụng cho mục đích truy vấn. Đối với các truy vấn có mục đích này phải được lưu trữ như là các luật trong một file được chỉ định sẵn, file này được sử dụng để liên kết với dữ liệu. Lệnh lọc CWM cho phép tự động lựa chọn mà nó được tạo ra bởi các luật. N3 thậm chí còn có các thuộc tính trực giao, thuộc tính đóng và an toàn, nhưng sử dụng N3 làm ngôn ngữ truy vần thì rất nặng nề.

N3 được hỗ trợ bởi hai hệ thống tự do có sẵn là Euler và CWM. Cả hai hệ thống này đều không làm việc gắn liền tự động với ngữ nghĩa RDF. Ngữ nghĩa này được cung cấp bởi các luật được tạo ra bởi người sử dụng [6].

3.2. So sánh và đánh giá

Trong phần này chúng tôi xem xét các ngôn ngữ trong ngữ cảnh so sánh tổng quát giữa chúng để thấy được ưu điểm và nhược điểm của từng ngôn ngữ. Sự so sánh giữa các ngôn ngữ truy vấn ngữ nghĩa này dựa theo một số tiêu chuẩn sau [5].

3.2.1. Hỗ trợ cho mô hình dữ liệu RDF

Vì mô hình dữ liệu RDF không phụ thuộc vào một cú pháp mã cụ thể, các ngôn ngữ truy vấn thường không cung cấp các tính năng cho các chức năng đặc tả đoạn mã truy vấn, chẳng hạn như thứ tự của đoạn mã. RQL là ngôn ngữ truy vấn duy nhất có cung cấp cho sự hỗ trợ này. RQL tương thích một phần với ngữ nghĩa học chính thức của RDF. N3, TRIPLE hỗ trợ ngữ nghĩa học RDF và kế thừa theo thứ tự thông qua các luật được tạo theo ý người dùng. Các hệ thống của SPARQL cũng hỗ trợ vấn đề này chẳng hạn như Jena hỗ trợ kế thừa theo thứ tự thông qua các suy luận các cơ chế các luật và trên cơ sở ontology.

3.2.2. Các thuộc tính ngôn ngữ truy vấn

Thuộc tính đóng (Closure): Thuộc tính đóng đòi hỏi rằng các kết quả của một

truy vấn cũng chính lại là các thành phần của mô hình dữ liệu. Điều đó có nghĩa là nếu một truy vấn hoạt động trên một mô hình dữ liệu đồ thị thì kết quả truy vấn đó một lần nữa phải là các đồ thị. N3 và SeRQL hỗ trợ cho thuộc tính đóng, trong khi SPARQL cung cấp câu lệnh CONSTRUCT cho mục đích này

Thuộc tính đầy đủ (Adequacy): Một truy vấn được gọi là đầy đủ nếu nó sử dụng tất cả các khái niệm của mô hình dữ liệu cơ sở. Do đó thuộc tính này bổ sung cho thuộc tính đóng. Đối với thuộc tính đóng một kết quả truy vấn không được ở bên ngoài các mô hình dữ liệu đối với thuộc tính đầy đủ toàn bộ dữ liệu cần được khai thác. SPARQL hỗ trợ thuộc đầy đủ. Ánh xạ từ mô hình TRIPLE đến các mô hình dữ liệu RDF có sự mất đi và vì thế có tính đầy đủ không hoàn chỉnh.

Tính trực giao (Orthogonality) đòi hỏi tất cả các hoạt động được sử dụng không phụ thuộc và ngữ cảnh sử dụng. Trong tất cả các ngôn ngữ, RDQL không hỗ trợ tính trực giao.

Tính an toàn (Safety) đòi hỏi một truy vấn đúng về ngữ pháp trả về một tập hữu hạn các kết quả và trên một tập dữ liệu hữu hạn. Các khái niệm tiêu biểu mà nó làm cho các ngôn ngữ trở nên không an toàn các hàm đệ quy, phủ định và các hàm được dựng sẵn (built – in). RDQLvà N3 là các ngôn ngữ truy vấn an toàn.

3.2.3. Các biểu thức đường dẫn

Các biểu thức đường dẫn được đưa ra dưới các dạng cú pháp khác nhau trong tất cả các ngôn ngữ truy vấn RDF. Một vài ngôn ngữ truy vấn RDF như là SeRQL hỗ trợ cách thức để giải quyết các thông tin không đầy đủ hoặc không đúng quy cách. SPARQL hỗ trợ cho sự không đầy đủ hay không đúng quy cách này bằng cách sử dụng cấu trúc OPTIONAL có RQL hỗ trợ một phần cho việc tùy chọn các ràng buộc biến.

3.2.4. Các phép toán đại số cơ bản

Trong mô hình dữ liệu quan hệ, các phép toán đại số cơ bản được quan tâm như là phép chọn, phép chiếu, phép tích Descarte, phép hợp và phép hiệu. Ba phép toán đại số cơ bản là phép chiếu, phép chọn và tích số đều được hỗ trợ trong tất cả các ngôn ngữ được đề cập. Phép hợp được hỗ trợ trong tất cả các ngôn ngũ ngoại trừ RDQL. Phép hiệu được hỗ trợ trong RQL, SeRQ và SPARQL.

3.2.5. Định lượng

Một predicate được gọi là xác định tồn tại trên một tập hợp các tài nguyên (an existential predicate) nếu có ít nhất một giá trị thoả mãn predicate. Tương tự, một predicate được gọi là phổ biến (universal predicate) nếu tất cả các giá trị thoả mãn predicate. Định lượng tồn tại (existential quantification) được hỗ trợ bởi tất cả các ngôn ngữ khi bất cứ predicate lựa chọn (selection predicate) có ràng buộc tồn tại. Định lượng phổ biến (universal quantification) được hỗ trợ bởi RQL, SeRQL, SPARQL và một phần trong TRIPLE.

3.2.6. Sự tổng hợp và sự gộp nhóm

Các hàm tổng hợp tính toán các giá trị vô hướng từ nhiều tập các giá trị. Một trường hợp đặc biệt của sự kết hợp là đếm số lượng các thành phần trong một tập hợp mà nó chỉ được hỗ trợ bởi RQL và N3. Ngoài ra sự gộp nhóm cho phép các tổng hợp được tính toán trên các nhóm giá trị. SPARQL hỗ trợ tính năng này thông qua mệnh đề ORDER BY.

3.2.7. Sự đệ quy

Các truy vấn đệ quy thường được xử lý trên các kịch bản nơi mà mối quan hệ cơ bản được ngầm hiểu trong tự nhiên. Biểu hiện của một thuộc tính được đưa ra trong truy vấn như các mối quan hệ đối xứng và các mối quan hệ ngầm định không phải là hỗ trợ có sẵn trong RDF. TRIPLE và N3 hỗ trợ cho đệ quy thông qua các luật. Đệ quy cũng là một trong các nguyên nhân làm cho các ngôn ngữ truy vấn trở nên không an toàn.

3.2.8. Reification

Reification là một độc đáo của RDF, cho phép chúng ta xem xét các phát biểu RDF các tài nguyên chính nó. Refication được hỗ trợ trong một số phạm vi trong tất cả các ngôn ngữ ngoại trừ N3.

3.2.9. Collections và Container

RDF cho phép chúng ta định nghĩa các nhóm của các thực thể bằng cách sử dụng các Collection (là nhóm đóng của các thực thể) và các Container. Tất cả các ngôn ngữ đang được nghiên cứu hỗ trợ một phần khả năng để truy hồi những tập hợp này và các thành phần của nó với thông tin có thứ tự.

3.2.10. Namespace

Đưa ra một tập các tài nguyên, nó có thể đang quan tâm đến truy vấn tất cả các giá trị của thuộc tính từ một namespace đã biết hay một namespace với một pattern đã biết.Pattern phù hợp trên các namespace hữu dụng rất nhiều cho dữ liệu RDF đã được phiên bản hóa, như nhiều lượt đồ phiên bản dựa trên namespace để mã hóa thông tin phiên bản. Tất cả các ngôn ngữ truy vấn ngoại trừ TRIPLE và Versa hỗ trợ tính năng này, RDQL cũng hỗ trợ một phần cho tính năng này.

3.2.11. Các Literal và các kiểu dữ liệu

RDF hỗ trợ các kiểu hệ thống của lượt đồ XML để tạo ra các typed literal (literal định kiểu). Một ngôn ngữ truy vấn RDF phải hỗ trợ các kiểu dữ liệu lượt đồ XML. Một kiểu dữ liệu bao gồm một không gian từ vựng, một khoảng không gian giá trị và một ánh xạ từ vựng đến giá trị. Tất cả các ngôn ngữ truy vấn đều có hỗ trợ cho không gian từ vựng, trong khi chỉ có RQL, SeRQL có hỗ trợ cho không gian giá trị. RDQL cũng hỗ trợ một phần cho không gian giá trị.

3.2.12. Sự kế thừa theo thứ tự (Entailment)

Từ vựng lượt đồ RDF hỗ trợ cho sự kế thừa theo thứ tự của cá thông tin ẩn dấu ví dụ như sử dụng các mối quan hệ subclass, domain và range. Tất cả các ngôn ngữ truy vấn hỗ trợ cho kế thừa có thứ tự với các mức độ khác nhau.

3.2.13. Các thuộc tính mong muốn (Desired Properties)

Nhiều ngôn ngữ truy vấn hỗ trợ rất ít chức năng gộp nhóm và tổng hợp. Đáng ngạc nhiên là, ngoại trừ SPARQL, không một ngôn ngữ nào có khả năng phân loại và sắp xếp trên thông tin đầu ra.Do bản chất bán cấu trúc của RDF, hỗ trợ cho các đối sánh tùy chọn là rất quan trọng trong bất kỳ ngôn ngữ truy vấn RDF nào vì vậy các ngôn ngữ khác nên được hỗ trợ với một cú pháp chuyên dụng như trong SPARQL. Nói chung, hỗ trợ ngôn ngữ cho các tính năng đặc tả RDF như các container, các collection, các kiểu dữ liệu lượt đồ XML và reification là khá nghèo nàn. Vì đây là các tính năng của mô hình dữ liệu, mức độ đầy đủ trong các ngôn ngữ là rất thấp.

Từ những so sánh trên ta có bảng tổng hợp các so sánh:

Bảng 3.1. *Bảng tổng hợp so sánh các ngôn ngữ truy vấn trong Web ngữ nghĩa*

Các ngôn ngữ / Tiêu chuẩn so sánh	**RDQL**	**Triple**	**SeRQL**	**N3**	**RQL**	**SPARQL**
Hỗ trợ mô hình dữ liệu RDF	N	Y	N	Y	Y	Y
Tính đóng	N	N	Y	Y	N	Y
Tính đầy đủ	N	R	N	N	N	Y
Tính trực giao	N	Y	Y	Y	Y	Y
Tính an toàn	Y	N	N	Y	Y	R
Biểu thức đường dẫn	Y	Y	Y	Y	Y	Y
OPTIONAL path	N	N	Y	N	R	Y
Phép hợp (UNION)	N	Y	Y	Y	Y	Y
Phép hiệu (DIFFERENCE)	N	N	Y	N	Y	Y
Định lượng (Quantification)	N	R	Y	N	Y	Y
Sự tổng hợp và gộp nhóm	N	N	N	Y	Y	Y
Đệ quy	N	Y	N	Y	N	N
Reification	R	R	Y	N	R	Y
Collections and Containers	R	R	R	R	R	Y

Namespace	R	N	Y	Y	Y	Y
Lexical Space	Y	Y	Y	Y	Y	Y
Value Space	R	N	Y	N	Y	Y
Kế thừa có thứ tự	R	R	Y	R	Y	Y
Thuộc tính mong muốn	N	N	N	N	N	Y

Trong đó: Y: Có hỗ trợ. N: Không hỗ trợ. R: Hỗ trợ nhưng hạn chế.

Bảng so sánh tổng hợp trên đây (Bảng 3.1) với cơ sở là sự phân tích các ngôn ngữ truy vấn siêu dữ liệu khác trên Web ngữ nghĩa, chúng ta có thể nhận thấy rằng ngôn ngữ SPARQL thừa hưởng các đặc tính tốt của tất cả các ngôn ngữ khác. Việc chọn và sử dụng SPARQL cho các truy vấn ngữ nghĩa là một sự lựa chọn tốt. Tuy nhiên SPARQL vẫn còn phải tiếp tục phát triển và mở rộng để đáp ứng các yêu cầu truy vấn nâng cao hơn.

4. Kết luận

Bài báo đã trình bày tóm tắt các đặc tính của ngôn ngữ truy vấn siêu dữ liệu cho Web ngữ nghĩa là SPARQL, đồng thời cũng tiến hành nghiên cứu, phân tích và đánh giá các ngôn ngữ truy vấn khác trên Web ngữ nghĩa để tìm ra điểm mạnh, yếu trong tương quan với ngôn ngữ SPARQL. Bài báo cũng đã tổng kết các phân tích và đánh giá các ngôn ngữ trong một bảng thông tin với các đặc tính của ngôn ngữ truy vấn Web ngữ nghĩa, làm cơ sở cho việc chọn lựa các ngôn ngữ truy vấn phù hợp trong quá trình phát triển các ứng dụng Web ngữ nghĩa trong tương lai. Hướng phát triển của bài báo là sẽ nghiên cứu và mở rộng ngôn ngữ SPARQL để có thể kết hợp với các ngôn ngữ luật cũng như tăng cường khả năng tìm kiếm toàn văn (fulltext) đối với dữ liệu web ngữ nghĩa.

TÀI LIỆU THAM KHẢO

[1]. Broekstra, J., and A. Kampman. *SeRQL - An RDF Query and Transformation Language*. Proceedings of the SWAD-Europe workshop on Semantic Web Storage and Retrieval, (2003), 1-20.

[2]. D. Beckett. *SPARQL query results XML format*. Technical report, W3C, 2005.

[3]. Eric Prud'hommeaux, Andy Seaborne. *SPARQL Query Language for RDF*. http://www.w3.org/TR/rdf-sparql-query/ (W3C Recommendation), 2008.

[4]. Grigoris Antoniou, Frank van Harmelen. *A Semantic Web Primer*. MIT Press, 2004.

[5]. Haase, P., J. Broekstra, A. Eberhart, and R. Volz. *A Comparison of RDF Languages*. Proceedings of the 2004 International Semantic Web Conference, LNCS, Vol 3298, (2004), 502-517.

[6]. Hoàng Hữu Hạnh. *Web ngữ nghĩa: những thách thức và hướng tiếp cận mới.* Tạp chí Khoa học Đại học Huế, Số 48, (2008), 31-40.

[7]. J. Perez, M. Arenas, C. Gutierrez. *Semantics and Complexity of SPARQL*. Proceedings of the 2004 International Semantic Web Conference, LNCS, Vol 4273, (2006), 30-43.

[8]. James Bailey, Fran¸cois Bry, Tim Furche and Sebastian Schaffert. *Web and Semantic Web Query Languages: A Survey*. 2005.

[9]. Karvounarakis, G., S. Alexaki, V. Christophides, D. Plexousakis, and M.Schol. *RQL: A Declarative Query Language for RDF*. Proceedings of the 11th International World Wide Web Conference, (2002), 592-603.

[10]. Marcelo Arenas, Claudio Gutierrez, Jorge P´erez. *SPARQL Formalization*. In Intẻnational Semantic Web Conference Tutorial, 2007.

[11]. Seaborne, A. *RDQL: A Query Language for RDF*. W3C Member Submission. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/ (September 10, 2007).

[12]. Sintek, S.Decker. *TRIPLE - an RDF query, inference and transformation language.* Deductive Databases and Knowledge Management, 2001.

AN SURVEY ON RDF QUERY LANGAUES: EVALUATION AND COMPARISION ON KEY FEATURES

Hoang Nguyen Tuan Minh
College of Sciences, Hue University
Hoang Huu Hanh
Hue University

SUMMARY

Resource Description Framework (RDF) is regarded as the primary facility to describe the information of Web resources semantically. RDF is used to represent a number of data sources ranging from personal information to digital documents and archives. RDF is also used as a foundation for the heterogeneous data integration. SPARQL language can be used to query different data sources in RDF triple format. This paper presents the strengths and weaknesses of SPARQL query language in the context of analysis and evaluation of different RDF query languages. The outcome of the evaluation is used to clarify the strong points of SPARQL in Semantic Web information retrieval.