RDF 데이터에 대한 경로 질의 처리

김성완
삼육대학교 컴퓨터학부
swkim@syu.ac.kr

Query Processing for Path Query on RDF Data

Sung Wan Kim
Division of Computer, Sahmyook University

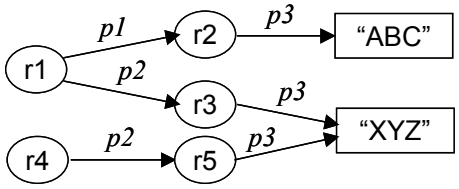
요 약

웹 리소스에 대한 메타 데이타 표현을 위해 RDF가 표준안으로 제정됨에 따라 RDF 데이타 저장 및 질의 처리 등의 연구가 많이 진행되어 왔다. 전통적인 저장 시스템을 기반으로 하는 저장 구조는 단순한 트리플 패턴 기반의 질의 처리에는 효율적이다. 그러나 여러 개의 트리플 패턴들이 결합된 질의 즉, 경로 기반의 질의 처리는 많은 조인 연산이 요구된다. 한편, 경로 질의의 효율적인 처리를 위해 접미사 배열을 응용한 인덱싱 기법이 제안되었다. 그러나 이 연구에서는 부분 경로식을 포함한 단순 경로 기반 질의 처리의 성능이 향상됨을 보여주었으나 다양한 경로 질의 유형에 대한 처리는 고려하지 않았다. 본 논문에서는 접미사 배열을 응용한 인덱싱 기법을 기반으로 한 경로 기반의 RDF 질의 처리 방안을 기술 한다. 특히, 단순 경로 질의처리 이외에 다양한 질의 유형의 처리를 위한 방안들을 예제를 통해 설명한다.

1. 서 론

시맨틱 웹은 현재의 웹 데이타에 대해 의미적 정보를 명세하는 메타 데이타를 부가하여 보다 지능적인 웹 기반 정보 시스템을 구축하게 해준다. 일반적인 정보 시스템에서와 마찬가지로 시맨틱 웹 기반의 정보 시스템에서의 질의 처리는 매우 중요한 기능이다.

한편, W3C에서 RDF [6]를 웹 리소스에 대한 메타 데이타 표현을 위한 표준안으로 제정함에 따라 RDF 질의어 정의, RDF 데이타 저장 및 질의 처리 등의 연구가 활발히 진행되어 왔다. RDF 데이타는 <Subject, Property, Object> 형태의 트리플(또는 문장)의 집합이다. 여기서, Subject와 Property는 리소스 URI를 의미하며, Object는 리소스 URI 또는 리터럴 값을 갖는다. 또한, RDF 데이터는 <그림 1>과 같이 Subject와 Object를 노드로 하고 Property를 간선으로 갖는 방향성 그래프 형태로 표현할 수 있다. 그림에서 타원은 리소스를, 사각형은 리터럴을 각각 의미한다. 간선은 Subject와 Object의 관계를 의미한다.



<그림 1> RDF 그래프 예제

RDB 또는 ORDB 등의 전통적 저장 시스템을 기반으로 하는 기존의 대부분의 연구들에서는 RDF 데이타를 트리플 집합으로 분해하여 여러 테이블에 분할 저장 및 관리를 하고 있다. 이러한 시스템은 단순한 트리플 패턴 기반의 질의 처리에는 효율적이다. 여러 개의 트리플 패턴들이 결합된 질의는 일반적으로 경로식을 이용하여 표현될 수 있다. Matono 등은 [1]에서 경로 기반 질의 처리를 위해 접미사 배열을 응용한 기본적인 인덱싱 기법을 제안하고 실험을 통해 부분(partial) 경로식을 포함한 고정된 길이를 가지는 단순 경로 기반의 질의 처리 성능이 향상됨을 보여주었으나 다양한 경로 질의 유형에 대한 처리는 고려하지 않았다. [2]에서는 경로를 이용한 RDB 기반의 저장 구조를 제안하였으나 경로 기반 질의 유형에 따라 여전히 트리플 기반의 질의 기법과 동일한 처리가 요구되는 경우가 발생한다.

본 논문에서는 [1]에서 제안된 인덱스 구조를 기반으로 다양한 경로 기반의 질의 처리 방안들을 기술한다. 2장에서는 연구 동기를 서술한다. 3장에서는 제안 인덱싱 기법과 4장에서는 이를 이용한 질의 처리 예제를 각각 설명하고 5장에서는 탐색 공간 축소를 위한 인덱싱 구성을 설명한다. 6장에서는 결론을 서술한다.

2. 연구배경 및 관련연구

현재까지 제안된 대부분의 RDF 질의어들은 경로식 표현을 지원한다[4]. 경로 기반 질의는 원하는 자료를 접근하기 위해서 RDF 그래프를 순회하며 처리되어야 하며 전통적인 저장 시스템 기반의 저장 구조에서는 다수의 조인 연산을 통해 처리될 수 있다.

Matono 등은 [1]에서 경로 기반 질의 처리를 위해 접미사 배열(suffix array)를 활용한 인덱싱 기법을 제안하였다. 접미사 배열은 텍스트 검색 응용에서 널리 활용되는 자료구조로 이진 탐색을 이용하여 텍스트에서 임의의 스트링 패턴을 빠르게 찾아낼 수 있다. 이 인덱싱 기법에서는 RDF 데이타를 DAG(Directed Acyclic Graphs)로 간주하고 루트 노드(집입 차수가 0)로 부터 단말 노

드(진출 차수가 0)에 이르는 노드 레이블과 간선 레이블이 교대로 구성된 모든 경로들을 추출하였다. 예를 들어 <그림 1> 에서 r1.*p1*.r2.*p3*."ABC" 형태의 경로를 추출할 수 있다. 각 경로로 부터 추출된 접미사들에게는 경로 식별자(PID)와 인덱싱 포인트 쌍으로 구성된 *접미사 레이블*이 할당되며, 이는 접미사 배열의 기본 자료 단위로 활용된다. <그림 2>는 <그림 1>로 부터 추출된 3개의 경로와 각 접미사들에 대해 경로 식별자(PID)와 인덱싱 포인트의 할당을 테이블 형태로 표현한 것이다.

<그림 3>은 접미사 배열 인덱스를 구축하는 과정을 나타낸 것으로 모든 접미사 패턴들을 사전 순으로 정렬하고 중복되는 접미사들은 제거한다.

PID \ point	1	2	3	4	5
1	r1	*p1*	r2	*p3*	"ABC"
2	r1	*p2*	r3	*p3*	"XYZ"
3	r4	*p2*	r5	*p3*	"XYZ"

<그림 2> 경로 정보 테이블

r1.*p1*.r2.*p3*."ABC" (1, 1)	>	(1, 5) "ABC"
p1.r2.*p3*."ABC" (1, 2)		(2, 5) "XYZ"
r2.*p3*."ABC" (1, 3)		~~(3, 5) "XYZ"~~
p3."ABC" (1, 4)		(1, 2) *p1*.r2.*p3*."ABC"
"ABC" (1, 5)		(2, 2) *p2*.r3.*p3*."XYZ"
r1.*p2*.r3.*p3*."XYZ" (2, 1)		(3, 2) *p2*.r5.*p3*."XYZ"
p2.r3.*p3*."XYZ" (2, 2)		(1, 4) *p3*."ABC"
r3.*p3*."XYZ" (2, 3)		(2, 4) *p3*."XYZ"
p3."XYZ" (2, 4)		~~(3, 4) *p3*."XYZ"~~
"XYZ" (2, 5)		(1, 1) r1.*p1*.r2.*p3*."ABC"
r4.*p2*.r5.*p3*."XYZ" (3, 1)		(2, 1) r1.*p2*.r3.*p3*."XYZ"
p2.r5.*p3*."XYZ" (3, 2)		(1, 3) r2.*p3*."ABC"
r5.*p3*."XYZ" (3, 3)		(2, 3) r3.*p3*."XYZ"
p3."XYZ" (3, 4)		(3, 1) r4.*p2*.r5.*p3*."XYZ"
"XYZ" (3, 5)		(3, 3) r5.*p3*."XYZ"

<그림 3> 접미사 배열 인덱스의 구축

여기서 최종적으로 접미사 배열은 접미사 레이블 (1, 5) (2, 5) (1, 2) (2, 2) (3, 2) (1, 4) (2, 4) (1, 1) (2, 1) (1, 3) (2, 3) (3, 1) (3, 3) 으로 구성 되며, 질의 처리는 <그림 2>의 경로 정보 테이블을 함께 이용하여 처리된다. 간단한 예제에 대한 질의 처리는 다음과 같다. 질의문은 RDQL 형태로 표현하였다.

```
예) select ?x
    where (r1 p2 r3) (r3 p3 ?x)
```

위 예제는 주어진 경로 패턴으로부터 도달 가능한 리소스를 검색하는 예이다. where 절의 조건을 경로식을 이용해 표현하면 'r1.*p2*.r3.*p3*.?x'와 같이 나타낼 수 있다.

경로 정보 테이블과 접미사 배열에 대한 이진 탐색을 이용하여 주어진 경로 패턴 'r1.*p2*.r3.*p3*'에 일치하는 접미사 레이블 (2, 1)을 찾은 후 쉽게 답을 구해낼 수 있다. 그러나, [1]의 논문에서는 고정된 길이를 갖으며 경로식을 구성하는 요소들이 빠짐없이 순차적으로 구성된 단순 경로식만을 고려하였으며 경로 구성 요소가 빠진 경우나 임의 길이를 갖는 경로식의 처리는 언급하지 않고 있다. 또한, 다음과 같은 역방향 경로 탐색 질의 경우 올바른 결과를 구할 수 없다.

```
예) select ?x
    where (?x p3 "XYZ")
```

우선 접미사 배열에서 경로 패턴 'p3."XYZ"'에 일치되는 접미사 레이블을 검색하면 (2, 4)를 찾게 되어 PID가 2인 경로상의 리소스 r3을 결과로 포함할 수 있다. 그러나, 중복되는 접미사 패턴을 삭제했기 때문에 PID가 3인 경로상의 리소스 r5는 찾을 수 없게 된다.

한편, [2]에서는 경로 정보를 이용한 RDB 기반의 저장 구조를 제안하였다. 여기서는 각 리소스에 대해 루트 노드로부터 해당 리소스에 이르는 경로를 추출하였다. 경로 표현식에는 프로퍼티만을 포함하도록 하였으며 특히, 와일드 카드로 시작되는 경로 질의를 처리하기 위해 역방향 경로 표현을 사용하였다. 이렇게 추출된 경로 정보는 경로 테이블에 경로 식별자와 함께 저장하고, 리소스 테이블에서 경로 식별자를 참조하도록 저장 구조를 구성하였다. 그러나, 경로 기반의 질의 유형에 따라 경로 정보를 이용하지 못하고 여전히 트리플 기반의 저장 기법과 동일하게 조인 연산들이 요구되는 경우가 발생한다.

3. RDF 데이터의 인덱싱 및 저장

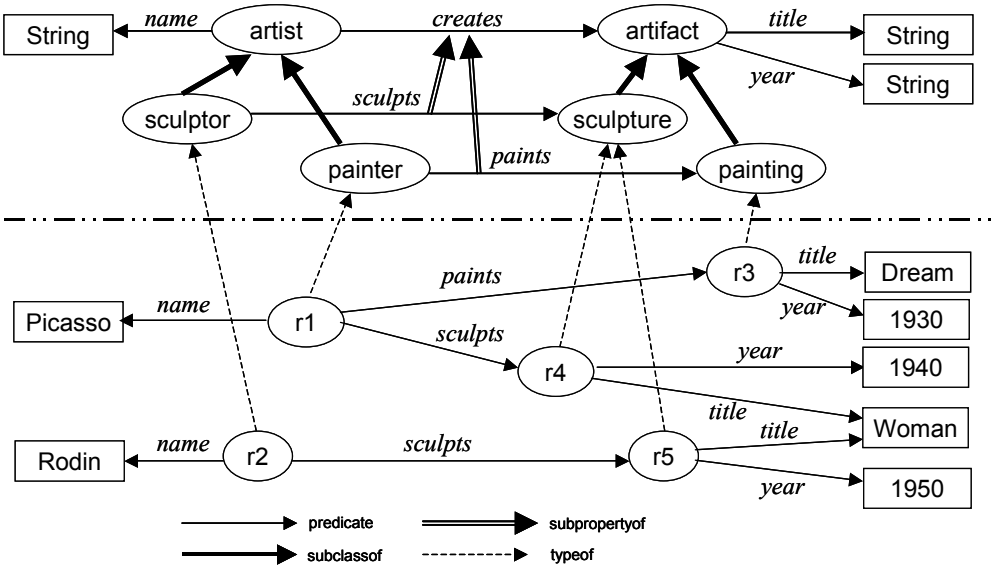
<그림 4>는 본 논문에서 사용될 예제이다. 상단 부분은 RDF Schema를 통해 정의된 내용을, 하단 부분은 RDF를 통해 명세된 리소스 서술 내용(resource description)을 각각 나타내며, 네임 스페이스는 포함하지 않았다. 본 논문에서는 주로 하단의 리소스 서술 내용만을 질의 처리 대상으로 고려하도록 한다. 또한, RDF 데이터는 DAG 형태로 간주한다.

본 논문에서 경로는 리소스 서술 내용에 대한 RDF 그래프상의 경로 구성 요소들 즉, 노드 레이블과 간선 레이블이 교대로 구성된 것으로 정의한다.

```
Path    ::=  (rscLabel '.' propLabel '.')*(rscLabel | literalValue)
rscLabel ::= URI Reference
propLabel ::= propName
literalValue ::= '"' literal '"'
propName ::= URI Reference
literal ::=Constant Values
```

3. 1 인덱스 구성 및 저장

기본적인 인덱싱 구성은 접미사 배열을 이용한 [1]의 기법과 동일하다. 다만, 2장에서 설명한 바와 같이 역방



<그림 4> 예제 RDF 그래프

향 경로 탐색 질의를 지원하기 위해 다음의 3가지 방법을 고려할 수 있다.

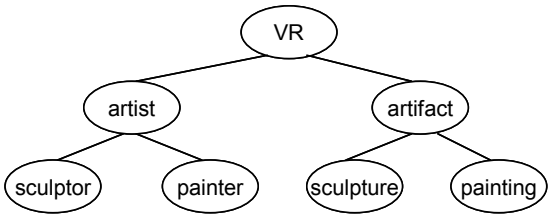
첫째, 가장 단순한 방법으로 중복되는 접미사 패턴을 삭제하지 않는 방법이다. 둘째, RDF 그래프로 부터 추출된 경로 패턴들에 대해 역방향 경로들로 각각 변환한 후 이를 이용하여 역방향 경로 기반의 접미사 배열 인덱스를 별도로 구축하는 방법이다. 역방향 경로 탐색 질의 처리는 사용자 질의에 포함된 경로 패턴을 역방향 경로로 변환하여 처리하면 된다. 셋째, 2장의 방법과 같이 접미사 배열을 구성하되 중복되는 접미사 패턴들에 대한 접미사 레이블들을 각각 독립적인 리스트에 유지하여 접미사 배열의 해당 요소에 연결하는 방법이다.

본 논문에서는 일차적으로 첫 번째 방법만을 고려하기로 한다. 인덱싱 구축 절차는 2장에서 설명한 것과 거의 동일하므로 본 장에서는 생략하기로 한다. <그림 4> 하단의 RDF 데이터에 대해 추출된 모든 경로 정보는 <그림 5>와 같다.

	1	2	3	4	5
1	r1	*paints*	r3	*title*	Dream
2	r1	*paints*	r3	*year*	1930
3	r1	*sculpts*	r4	*title*	Woman
4	r1	*sculpts*	r4	*year*	1940
5	r1	*name*	Picasso		
6	r2	*sculpts*	r5	*title*	Woman
7	r2	*sculpts*	r5	*year*	1950
8	r2	*name*	Rodin		

<그림 5> 경로 정보 테이블

추출된 각 경로의 접미사에는 경로 식별자(PID)와 인덱싱 포인트 쌍으로 구성된 레이블이 할당된다. 여기서는 이를 *접미사 레이블*이라고 하자. 이와 같이 구성된 경로들을 이용하여 생성된 접미사 배열은 (2, 5)(4, 5)(7, 5)(1, 5)(5, 3)(8, 3)(3, 5)(6, 5)(5, 2)(8, 2)(1, 2)(2, 2)(5, 1)(1, 1)(2, 1)(3, 1)(4, 1)(8, 1)(6, 1)(7, 1)(1, 3)(2, 3)(3, 3)(4, 3)(6, 3)(7, 3)(3, 2)(4, 2)(6, 2)(7, 2)(1, 4)(3, 4)(6, 4)(2, 4)(4, 4)(7, 4) 와 같다.



<그림 6> 클래스 트리

ClassName	Start	End	Level
artist	1	3	1
sculptor	1	1	2
painter	2	2	2
artifact	4	6	1
sculpture	4	4	2
painting	5	5	2

<그림 7> 클래스 계층 테이블

한편 [3][5]에서 활용한 구간(interval) 기반의 레이블링 기법을 이용하여 클래스 및 프로퍼티 계층 구조 정보를 따로 관리한다. <그림 4>의 RDF Schema 부분에 정의된 내용 중 클래스들에 대해 <그림 6>과 같이 트리 구조를 구성하고 각 클래스에 (시작번호, 종료번호, 레벨)로 구성된 레이블을 할당한다. 여기서, 루트 노드는 트리 구조를 구성하기 위해 삽입한 가상 노드이다. 종료번호는 클래스 계층을 후위순회 방식으로 순회할 때 해당 클래스를 방문한 순서를 의미한다. 시작번호는 해당 클래스의 가장 좌측의 후손 클래스의 종료번호를 의미한다. 레벨은 클래스 계층에서 해당 클래스의 레벨이다.

이러한 방식으로 각 클래스에 레이블을 할당 할 경우 특정 클래스의 하위 클래스를 단순한 수식 계산을 통해 구해낼 수 있다. 또한, 이렇게 할당된 클래스 레이블은 아래와 같은 구조의 클래스 테이블에 저장 관리된다. 프로퍼티 계층 정보도 이와 마찬가지로 관리한다. <그림 7>과 <그림 8>은 클래스 계층 정보 테이블과 프로퍼티 계층 정보 테이블을 각각 나타낸다.

PropertyName	Start	End	Level	domain	range
creates	1	3	1	artist	range
sculpts	1	1	2	sculptor	range
paints	2	2	2	painter	range

<그림 8> 프로퍼티 계층 테이블

클래스와 typeOf 관계에 있는 리소스 인스턴스의 관계는 <그림 9>과 같이 인스턴스 테이블에 따로 저장한다.

Resource	ClassName
r1	painter
r2	sculptor
r3	painting
r4	sculpture
r5	sculpture

<그림 9> 인스턴스 테이블

4. 질의 처리

4. 1 경로식의 질의 표현

다음은 본 논문에서 경로식을 이용한 질의 표현을 위해 사용한 표기 기호들을 나타낸 것이다.

_ : 경로를 구성하는 임의의 한 개 구성 요소
. : 경로상의 각 구성 요소에 대한 구분자
?n : 검색 대상 요소 (단, n은 변수명)

예를 들어 조각품을 조각한 작가 리소스와 조각품의 이름을 쌍으로 검색하는 질의에 대해 위의 기호를 이용하여 경로식으로 표현하면 '?x.sculpts._.title.?y'와 같이 나타낼 수 있다.

4. 2 경로식 분할

*단순 경로식*은 고정된 길이를 갖으며 경로식을 구성하는 요소들이 빠짐없이 순차적으로 나타난 것을 말한다. 이러한 단순 경로식은 2장에서 보인 예와 같이 접미사 배열 인덱스을 이용한 경로 패턴 매칭을 통해 쉽게 처리될 수 있다. 한편, *복합 경로식*은 경로식을 구성하는 요소가 빠진 경우나 경로식의 길이를 알 수 없는 경우를 의미하며 접미사 배열상의 이진 탐색을 수행하여 주어진 경로 패턴을 올바르게 찾아낼 수 없다. 이러한 복합 경로식은 여러 개의 단순 경로식들로 분할 될 수 있으며, 각 단순 경로식에 대한 처리 결과들을 결합하여 원래의 질의에 대한 최종 결과를 구성할 수 있다. 예를 들어, ?x.pred1._.pred2.rsc2의 경로식은 단순 경로식 ?x.pred1과 단순 경로식 pred2.rsc2로 분해되어 각각 처리된 후 두 결과를 결합하여 최종 결과를 구해낸다.

4. 3 질의 처리 방안

본 절에서는 3장 에서 구축된 인덱스 및 저장 구조를 이용한 복합 경로식으로 표현된 질의 처리 방안 및 그 과정을 질의 유형별로 예를 들어 설명한다. 질의문은 RDQL 형태로 표현하였다.

① 순방향 경로 탐색 질의

예1) 리소스 r1이 조각한 작품들의 제목을 검색하라.
```
select ?y
where (r1 sculpts ?x) (?x title ?y)
```

이 질의는 주어진 경로로 부터 도달 가능한 리소스를 검색하는 예제이다. where절의 조건을 경로식으로 표현하면 "r1.sculpts._.title?y"과 같으며 그 처리 과정은 다음과 같다. 우선 주어진 복합 경로식을 2개의 단순 경로식으로 분할하고 동시에 첫 번째 단순 경로식 'r1.sculpts'의 시작 요소로부터 두 번째 단순 경로식 'title.?y'의 시작 요소까지 인덱싱 포인트 값의 차이를 구해둔다(여기서는 3). 접미사 배열로부터 경로 길이가 2인 단순 경로식 'r1.sculpts' 패턴과 일치되는 접미사 레이블 집합 {(3, 1)(4 ,1)}을 구한다. 또한, 경로 길이가 1인 'title' 패턴을 처리하여 {(1, 4)(3, 4)(6, 4)} 집합을 얻는다.

양쪽 집합의 접미사 레이블들에 대해 데이터베이스의 정렬-합병(sort-merge) 조인 알고리즘을 적용하여 동일한 PID 값을 갖는 접미사 레이블들의 쌍들만을 구해낸 후, 두 접미사 레이블의 인덱싱 포인트 차이를 확인하여 (3, 1)과 (3, 4) 쌍을 구한다. 최종 결과는 두 번째 단순 경로식의 경로 길이 1을 (3, 4)의 인덱싱 포인트에 더하여 (3, 5)의 내용인 “Woman"을 최종적으로 구한다.

② 역방향 경로 탐색 질의

예2) 작품 제목이 "Woman"인 작품을 조각한 조각가를 검색하라.
```
select ?x
where (?x sculpts ?y) (?y title “Woman”)
```

위 예제는 주어진 경로에 대한 역방향 탐색에 대한 질의이다. where절의 조건을 경로식으로 표현하면 '?x.sculpts._.title.Woman'와 같다. 첫 번째 단순 경로식 'sculpts'에 일치하는 접미사 레이블 집합 {(3, 2) (4, 2) (6, 2) (7, 2)}를 추출하고, 두 번째 단순 경로식 'title.Woman'로부터 {(3, 4) (6, 4)}을 추출한다. 두 결과 집합에 대해 정렬-합병 조인 알고리즘을 적용하여 (3, 2)와 (3, 4)쌍과 (6, 2)와 (6, 4)쌍을 추출한다. 최종적으로 각 쌍의 첫 번째 접미사 레이블인 (3, 2)와

(6, 2)의 인덱싱 포인트에서 1을 뺀 (3, 2-1)과 (6, 2-1) 즉, (3, 1)과 (6, 1)의 내용인 r1과 r2를 최종적으로 구한다.

③ 계층 구조를 고려한 질의

예3) 리소스 r1이 제작한 작품의 작품명을 모두 검색하라.
```
select ?z
where (r1 creates ?y) (?y title ?z)
```

위의 예제에서 where 절의 조건을 경로식으로 표현하면 'r1.creates._.title.?z'와 같다. 또한, RDF Schema에 명세된 프로퍼티 계층 구조를 고려하여 프로퍼티 creates의 하위 프로퍼티인 paints와 sculpts도 질의에 포함되어야 한다. 하위 프로퍼티의 추출은 3.1절에서 설명한 것과 같이 프로퍼티에 할당된 레이블 값에 대해 간단한 수식 계산을 통해 구해낼 수 있다. 결과적으로 위 복합 경로식은 사용자의 선택에 의해 아래와 같이 2개의 확장된 질의도 포함하여 총 3개의 질의가 실행되어져야 한다.

```
r1.creates._.title.?z
r1.paints._.title.?z
r1.sculpts._.title.?z
```

예1과 같은 방식으로 3개의 복합 경로식들을 각각 처리하면 첫 번째 경로 패턴의 경우 검색 결과가 없으며, 두 번째 경로 패턴을 처리하여 접미사 레이블 (1, 5)의 내용 'Dream'과, 세 번째 경로 패턴을 처리하여 접미사 레이블 (3, 5)의 내용 'Woman'을 구하게 된다.

④ 그래프 패턴에 기반의 질의

지금까지는 RDF 그래프상의 한 개의 경로를 대상으로 한 질의들이었다. 아래에서는 두 개 이상의 경로를 대상으로 즉, 상이한 경로 식별자를 가지는 경로식들이 포함된 질의 처리에 대해 설명한다.

예4) Picasso가 조각한 작품의 이름을 모두 검색하라.
```
select ?z
where (?x name Piccaso) (?x sculpts ?y)
      (?y title ?z)
```

위 질의는 2개의 경로식이 질의에 포함된 경우이다. 첫째, 단순 경로식 '?x.name.Piccaso' 패턴을 처리하여 접미사 레이블 (5, 2)을 추출하고 인덱싱 포인트에서 1을 뺀 (5, 1) 즉, {r1}을 구한다. 둘째, 복합 경로식 '?x.sculpts._.title.?z' 패턴을 예제 3과 같은 방식으로 처리하면 {<r1 Woman> <r2 Thinker>}를 구하게 된다. 두 중간 결과에 대해 정렬-합병 조인 알고리즘을 적용하여 같은 리소스를 갖는 결과인 <r1 Woman>을 추출하여 최종적으로 “Woman"을 구한다.

한편, 질의 최적화 수행을 통해 위와 같이 처리하지 않고 첫 번째 경로식의 처리결과 수가 1개인 것을 이용하여 두 번째 경로식을 'r1.sculpts._.title.?z'과 같이 재구성하여 처리할 수도 있다.

⑤ 클래스 타입을 고려한 질의

예5) 조각품을 조각한 painter와 그 조각품 리소스 쌍을 검색하라.
```
select ?x ?y
where (?x sculpts ?y) (?x type painter)
```

위 질의도 예5와 같은 유형의 질의이나 클래스 타입에 대한 경로가 포함된 경우이다. 우선, 첫 번째 단순 경로식 '?x.sculpts.?y'를 처리하여 접미사 레이블 집합 {(3, 2) (4, 2) (6, 2) (7, 2)}을 추출하고, 각 접미사 레이블의 인덱싱 포인트에 대한 계산을 통해 변수 ?x와 ?y 쌍 {<r1 r4> <r2 r5>}를 구한다. 둘째, 인스턴스 테이블을 이용하여 painter 클래스 타입의 리소스 인스턴스들을 추출한다(여기서는 {r1}). 두 개의 중간결과에 대해 변수 ?x의 값을 기준으로 정렬-합병 알고리즘을 수행하면 <r1, r4>쌍을 최종 결과로 구할 수 있다.

⑥ 경로 길이를 알 수 없는 질의

예6) 조각을 한 조각가 리소스와 작품명을 쌍으로 검색하라.
```
select ?x, ?y
where ?x.sculpts._*.title.?y
```

이 예제는 경로의 길이가 정해지지 않은 정규 경로식 형태로 질의가 주어진 경우이다. '*'는 0개 이상의 구성요소 반복됨을 의미한다.

질의 처리는 예제 1과 같이 하나의 복합 경로식을 단순 경로식들로 분할하여 처리한다. 첫 번째 단순 경로식 '?x.sculpts'을 처리하여 접미사 레이블 집합 {(3, 2) (4, 2) (6, 2) (7, 2)}을 추출하고, 두 번째 단순 경로식 'title.?y'을 처리하여 접미사 레이블 집합 {(1, 4) (6, 4) (3, 4)}를 추출한다. 두 집합에 대해 정렬-합병 조인 알고리즘을 적용하여 같은 경로에서 구해진 요소들의 쌍을 추출하여 {(3, 2) (3, 4)}와 {(6, 2) (6, 4)}와 같은 접미사 레이블 쌍을 구한다. 여기서, 첫 번째 경로식과 두 번째 경로식 간의 인덱싱 포인트 차이를 구할 수 없으므로 접미사 레이블 쌍을 구성할 때 첫 번째 인덱싱 포인트 값이 두 번째 인덱싱 포인트 값보다 작은 경우만 고려하면 된다. 마지막으로 구성된 각 쌍에 대하여 첫 번째 접미사 레이블의 인덱싱 포인트에는 1을 빼고, 두 번째 접미사 레이블의 인덱싱 포인트에는 두 번째 단순 경로식의 경로 길이인 1을 더하여 최종 결과인 {(3, 1) (3, 5)}와 {(6, 1) (6, 5)}의 내용인 <r1 Woman>과 <r2 Woman>을 구한다.

5. 탐색 공간 축소를 위한 접미사 배열 인덱스의 구성

접미사 배열상의 이진 탐색을 이용한 질의 처리는 일반적으로 우수한 성능을 보인다. 그러나, RDF 그래프가

커지거나 본 논문에서와 같이 중복 접미사가 제거되지 않는 경우 접미사 배열의 크기가 커지게 되어 탐색 시간이 계속적으로 길어지게 된다.

접미사 배열을 이용한 인덱싱 구성을 살펴보면 접미사 패턴을 구성하는 각 요소들을 기준으로 추출된 모든 접미사 패턴들이 정렬되게 된다. 예를 들어, 첫째 요소를 중심으로 접미사 패턴들이 1차 정렬되고, 이 결과에 대해 둘째 요소를 중심으로 2차 정렬되어 클러스터링 된다. 또한, 질의 처리과정을 살펴보면 우선적으로 주어진 질의에 포함된 경로 패턴의 첫 번째 요소와 동일한 내용으로 시작하는 접미사 패턴들만을 접근하여 패턴 매칭을 수행한다. 이는 결국 그렇지 않는 접미사 패턴들은 탐색 공간에서 포함될 필요가 없는 것이다.

따라서, 단일 접미사 배열이 아니라 접미사들의 첫 번째 요소를 기준으로 물리적으로 분할된 다수의 접미사 배열들로 구성된 인덱스를 생성할 수 있다. 이 경우 'title'로 시작하는 경로 패턴을 가지는 질의는 'title'로 시작하는 접미사 패턴들의 접미사 레이블만 모아둔 접미사 배열에 접근하여 처리하게 된다. 이는 단일 접미사 배열 방법에 비해 접미사 배열에 대한 탐색 공간 축소를 통해 성능 향상이 가능하다. 다만, 인덱스 구성 시 상이한 각 프로퍼티들에 대해서만 다수의 독립적인 접미사 배열들을 생성할 것인가 아니면 상이한 각 리소스들 또는/그리고 리터럴 값들에 대해서도 다수의 접미사 배열을 생성할 것인가는 사용자 질의 유형의 분석 및 사용자의 선택에 의해 결정되어야 할 것이다. 예를 들어, 리소스가 경로 패턴의 첫 번째 요소로 주어지는 질의가 빈번히 발생하는 응용 환경의 경우 상이한 리소스의 개수 만큼 접미사 배열들을 유지하는 것이 성능 향상에 영향을 미치게 된다.

또한, 일정 개수 이상의 독립적인 접미사 배열이 생성될 경우 B-트리 등을 이용해 특정 접미사 배열을 빠르게 접근하도록 구성한다.

한편, 일반적으로 접미사 배열은 데이터의 갱신이 없는 정적인 환경에 적합한 자료 구조이다. 즉, 데이터의 갱신 시 접미사 배열에 대한 전체적인 재구성이 요구된다. 다수의 접미사 배열을 생성 시 갱신 영역을 축소할 수 있으므로 단일 접미사 배열의 경우 보다 우수한 성능을 기대할 수 있을 것이다.

6. 결론 및 향후 연구

본 논문에서는 접미사 배열을 이용한 인덱싱 기법을 기반으로 경로 기반의 RDF 질의 처리 방안을 기술하였다. 특히, 고정된 단순 경로 질의 처리 이외에 다양한 질의 유형의 처리를 위한 질의 처리 방안들을 예제를 통해 기술 하였다. 또한, 여기서는 지면 관계상 주로 리소스 서술 내용만을 인덱싱 및 질의 처리 대상으로 고려하도록 하였으나, RDF Schema에 대한 질의 처리는 3장에서 이용한 레이블링 기법을 응용하여 효율적으로 처리할 수 있을 것이다.

향후 연구로는 인덱싱 구성 및 질의 처리 방안에 대한 형식적인 정의 및 알고리즘 구성과 질의 처리 최적화를 고려한 몇 가지 제안 기법들에 대한 구현을 통해 실험적인 성능 평가를 하려 한다. 또한, RDF 그래프 상에 갱신이 발생 할 경우 처리 방법에 대한 연구도 요구된다.

한편, 지금까지 제안된 RDF 질의어에서 경로식 표현을 제공하고 있지만 대부분 고정된 길이를 갖는 기본적인 경로식만이 표현 가능하며 임의 길이에 대한 경로식 또는 정규 경로식 표현은 거의 지원되지 않고 있다[4]. 따라서 우선 RDF 질의어를 정규 경로식 표현이 가능하도록 확장 및 정의하고 이를 처리할 수 있는 적절한 인덱싱 및 질의 처리 방법도 계속 연구되어야 할 것이다.

Acknowledgement

본 논문은 삼육대학교 교원해외연수(2006년 7월 - 2007년 6월)사업의 지원에 의해 작성되었습니다.

참고문헌

[1] A. Matono, et al., "*An Indexing Scheme for RDF and RDF Schema based on Suffix Arrays*", First International Workshop on Semantic Web and Databases (SWDB), Sept. 2003 pp.151-168

[2] A. Matono, et al., "*A Path-based Relational RDF Database*", Proceedings of the 16th Australasian Database (ADC), 2005, pp.95-1034

[3] V. Christophides, et al., "*Optimizing Taxonomic Semantic Web Queries Using Labeling Schemes*", Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 1(2), 2004, pp.207-228

[4] P. Haase, et al., "A Comparison of RDF Query Languages", In Proceedings of the Third International Semantic Web Conference, 2004. pp. 502-517

[5] Y. Theoharis, et al, "*Benchmarking Database Representations of RDF/S Stores*", Proc. of the 4th ISWC 2005, pp.685-701

[6] W3C, RDF Primer (W3C Recommendation), Feb. 2004 at http://www.w3.org/TR/rdf-primer