RDF/S 스토리지 상에서 키워드 검색을 위한 SPARQL 변환기법*

이동훈○, 김학수 , 김보경 , 이동호†
한양대학교 컴퓨터공학과
dhlee401@hanyang.ac.kr, hsookim@hanyang.ac.kr, bkhacker@hanyang.ac.kr, dhlee72@hanyang.ac.kr

SPARQL Translation Technique for Keyword Search on RDF/S

Donghoon Lee○ , Hak Soo Kim , Bo-Kyeong Kim , Dong-Ho Lee
Department of Computer Science and Engineering, Hanyang University

1. 서론

일반적으로 키워드 검색은 정보 검색 분야에서 사용되는 용어로서 사용자가 키워드의 나열을 통해서 나열된 키워드를 포함하고 있는 문서를 검색하는 질의 처리기법이다.[1,2,3,4] 이와 같은 검색 기법은 사용자가 복잡한 질의 언어 또는 데이터의 구조에 대한 사전 지식을 요구하지 않는 장점을 가지고 있다.

구조화된 데이터에 대한 키워드 검색을 지원하면 다음과 같은 장점이 있다. 첫째, 질의 키워드 사이의 관계를 검색하는 방법으로 데이터의 구조에 대한 정보 없이도 검색이 가능하다. 둘째, 사용자는 특별한 질의 언어가 없어도 검색이 가능하다.

RDF데이터에 대한 질의 언어로는 원하는 데이터를 그래프 패턴 형태로 기술하는 SPARQL이 있다.

본 논문에서는 RDF데이터의 특성을 고려한 RDF데이터에 대한 SPARQL을 자동으로 생성하여 효율적인 키워드 검색 기법을 제안한다. 제안하는 기법에서는 OWL문서에 대한 RDF그래프에 대한 인덱스를 구성하고, 그래프 상에서 노드에 기존 Dewey넘버링 방식을 그래프 구조에 맞게 확장하여 RDF그래프 노드 사이의 넘버링을 하여 노드 사이의 관계 및 최소 경로를 빠르게 결정할 수 있는 인덱스 구조를 제안하고, 각 노드간의 패스(path)거리를 구하여 키워드 검색 결과에 대한 최소 경로를 찾아서 SPARQL질의 처리한다.

2. RDF/S 상에서 키워드 검색을 위한 인덱스 구조 생성 및 검색 알고리즘

본 논문에서는 RDF/S 및 OWL스토리지 상에서의 키워드 검색을 위하여 인덱스 구조 및 키워드 검색 알고리즘을 제안한다. 제안하는 알고리즘은 RDF 트리플구조 기반의 온톨로지 스토리지에 쉽게 적용할 수 있는 장점을 가지고 있다.

RDF/S 및 OWL 스토리지 에서의 키워드 검색의 결과는 모든 질의 키워드를 단 한번씩만 포함하는 최소 그래프이며, 사용자가 질의 키워드를 k_n개를 입력하면 구축되어 있는 마스터 인덱스로부터 질의 키워드를 포함하는 리소스 리스트를 가지고 온다.

2.1 인덱스 구조

마스터 인덱스는 관계형 데이터베이스 기반의 키워드 검색에서 제안한 인덱스로서 정보 검색의 역인덱스(Inverted Index)와 같은 구조이다. 즉, 키워드를 주 키(Primary Key)로 하여 키워드를 포함하는 트리플을 저장하는 구조로 되어 있다. 결과적으로 키워드를 포함하는 트리플을 빠른 시간 안에 검색할 수 있는 장점이 있기 때문에 이러한 구조가 많이 사용되고 있다.

2.2 최소 경로 계산 알고리즘

마스터 인덱스에서 각 키워드에 대한 노드리스트를 추출하여 각 노드들 사이의 최소 경로 검색을 하기 위해 각각의 노드에 LDN(Loopness Dewey Number)을 부여한다. LDN안에는 드웨이 넘버와 MPLN(Minimal Path Length Number)의 쌍으로 구성된다.

```
Algorithm 1 SearchMinimalPathLength(x,y)
Input: Node x and y with DN and MPLN
Output: return the minimal path length
1         return min[PLF(x,y),MPLNF(x,y)]

Function PLF(x,y)
Input: Node x and y
Output: return the path length by DN
1         return
(x.ln-DNofSCN(x,y).ln)+(y.ln-DNofSCN(x,y).ln)

Function DNofSCN(x,y,isNew)
Input: Node x and y,
       isNew(If true; Finding SCN with MPLN > 0)
Output: return the shortest common node between x and y

1    sncAt = -1;
2    sncWithMPLN = -1;
3    For( i = 0; i < x.DN.ln and i < y.DN.ln; i++)
4      If( x.DN[i] == y.DN[i] )
5        sncAt = i;
6        If( x.MPLN != '?' or y.MPLN != '?' )
7          sncWithMPLN = i;
8        End If
9      End If
10   End For;
11  return x.DN.sub(0,(isNew)? sncWithMPLN: sncAt);
```

정의 1 (Loopness Dewey Numbering) : *각 Node는 (Dewey Number, Minimal Path Length Number)의 쌍으로 구성되고 각 노드의 넘버링은 깊이 우선탐색(Depth First Search)에 의해 결정된다.*

* 이 논문은 2007년도 정부(과학기술부)의 재원으로 한국 과학재단의 지원을 받아 수행된 연구임 (No. R01-2007-000-20135-0)
* 본 연구는 지식경제부 및 정보통신 연구 진흥원의 대학 IT연구센터 육성·지원사업 (IITA-2009-c1090-0902-0031)의 연구결과로 수행되었음
† 교신저자

```
Function MPLNF(x,y)
Input: Node x and y
Output: return Path Length by MPLN

1  Scn = DNofSCN(x,y,false);
2  valueX = valueOfMPLN(x,scn,ln);
3  valueY = valueOfMPLN(y,scn.ln);
4  If( valueX > 0 and valueY > 0 )
5   pathLength = valueX + valueY;
6  Else
7   Scn = DNofSCN(x,y,true);
8   newValueX = getPathLength(m.mpln,x.ln,scn.ln);
9   newValueY = getPathLength(y.mpln,y.ln,scn.ln);
10  newPathLength = getMPLN(x,y);
11  pathLength = min((newValue+newValueY),newPathLength);
12 End If;
13 Return pathLength;
```

정의 2(Minimal Path Length Number : MPLN): *MPLN은 드웨이 넘버와 동일한 길이를 가지며 드웨이 넘버 내의 노드들의 최소 경로길이를 가지며 결정되지 않은 노드들 사이의 최소 경로길이는 '?'로 표현된다.*

2.3 노드 병합 알고리즘

입력되는 질의 키워드 수가 적거나 각 키워드당 검색되는 노드의 수가 적을 경우 최적의 각 키워드에 속하는 노드와 다른 키워드에 속하는 노드 사이의 최소 거리를 가지는 최적노드 집합을 계산하는 비용이 적다. 하지만 질의 키워드와 검색되는 노드의 수가 증가하게 된다면 계산비용이 기하 급수적으로 증가한다. 따라서 이러한 문제를 해결하기 위해 최적의 노드 조합의 근사치에 접근하는 노드 병합 알고리즘을 제안한다.

```
Algorithm 2 MergingNodeGroup(K)
Input: K = Sorted Array per each Group as shown in 그림 1
Output: A list of nodes with including all the input keywords

1         Array startGroup = selectMinimalGroup(K);
2         Array selectedNode;
3         Array resultSet;
4         remove startGroup from K;

5         For( i = 0; i < startGroup.length; i++ )
6           selectedNode.add(startGroup[i]);
7           For( next = 0; next < K.length; next++ )
8             currentSelectedNode                    =
binarySearch(selectedNode, K[next]);
9             selectedNode.add(currentSelectedNode);
10          End For;
11          resultSet.add(selectedNode);
12          initialize selectedNode;
13        End For;

14        Return resultSet;
```

그림 1은 노드 병합 알고리즘의 예이다. 사용자가 입력 질의 키워드가 "C,B,D"라고 가정하고 각 키워드를 포함하는 노드가 그림처럼 검색되었을 때, 노드의 수가 가장 적은 D가 앞에 오게된다. D이외의 다른 그룹들 사이에는 정렬이 필요 없다. 그리고 각 그룹의 노드 들은 LDN 값에 따라 정렬된다. 먼저 D키워드를 포함하는 첫 번째 노드를 선택하고 그 다음 B 그룹에서 최소 길이를 가지는 노드를 선택하기 위하여 이진 탐색을 진행한다. 이진 탐색의 비교 값은 계산된 길이 값으로 진행한다. 따라서, B 그룹에서 처음에는 "길이=4" 그룹의 노드들과 MPLN값을 계산하고 그 다음 "길이=2" 그룹의 노드들과 MPLN값을 계산하여 더 크면 검색을 중지하고 "길이=4"에서 계산된 값이 최소 길이를 가지는 노드로 선택된다. 하지만 본 예제에서는 더 작기 때문에 계속 이진 탐색을 진행한다. 결과적으로 "길이=1" 그룹의 노드 "0.0"이 최소 길이를 가지는 노드로 선택되었다. 마찬가지 방식으로 선택된 노드 {"0.0.1", "0.0"} 과 C 그룹에서 최소 길이를 가지는 노드를 이진 탐색[MergingNodeGRoup line 2,8]한다. 최소 길이를 계산하는 데 있어서 지금까지 선택된 두 노드 사이의 거리가 최소가 되는 길이를 기준으로 이진 탐색을 진행한다.

2.4 제안하는 알고리즘의 복잡도

본 논문에서 제안한 알고리즘으로 시간 복잡도를 분석을 하게 되면, 알고리즘의 주가 되는 이진 탐색의 시간 복잡도는 $O(n \log n)$ 이다. 그리고 입력 데이터의 최대수를 모두 n 으로 계산한다. 첫 번째 그룹의 노드 개수를 n 개라 하고 전체 그룹의 개수가 n 개라고 하면 이중 포문에 대한 시간 복잡도는 이고 이만큼 이진 탐색이 진행되기 때문에 시간 복잡도는 $O(n^3 \log n)$이 된다. 따라서, 이진탐색의 경우 정력이 보증되기 때문에 어떠한 값도 $O(n \log n)$ 번만 검색하면 찾을 수 있기 때문에 매우 효율적인 검색 방법이라고 할 수 있다

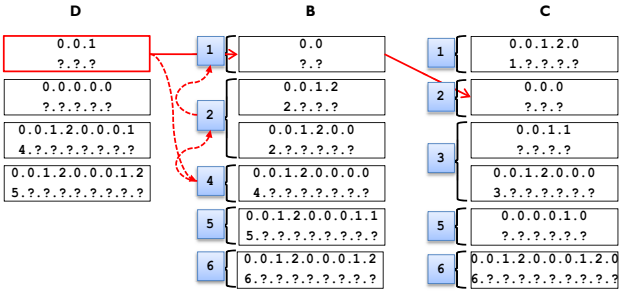

그림 1 노드 병합 알고리즘의 예

3. 결론

본 논문에서는 RDF/S 스토리지 상의 데이터에 대한 효율적인 키워드 검색 기법을 제안 하였다. 제안된 기법은 사용자가 입력한 키워드와 직접적으로 관련된 의미 있는 정보를 제공하고, RDF 그래프의 문제점이었던 전체적인 그래프 탐색 없이 인덱스만을 가지고 검색이 가능하도록 하였다. 즉, RDF 그래프의 각 노드에 루프니스 드웨이 넘버를 부여하여 각 노드들 사이의 거리를 빠르게 계산할 수 있게 하였고, 이진 탐색을 이용하여 검색하고자 하는 키워드에 대해 각 노드들의 최소 거리를 가지는 정보를 추출하여 자동으로 SPARQL질의로 변환 후 처리하는 방식이다.

이는 RDF 그래프의 문제점이었던 전체적인 그래프 탐색을 하여 생기는 비용을 탐색 없이 검색 함으로서 발생되는 비용을 감소시키는 장점이 있고, 보다 효율적으로 키워드 검색을 할 수 있다.

4. 참고문헌

[1] H. He, H.Wang, J. Yang, and P. S. Yu, "Blinks: ranked keyword searches on graphs," in SIGMOD Conference, pp. 305–316, 2007.

[2] Vagelis Hristidis et al. : DISCOVER: Keyword Search in Relational Databases. In VLDB, 2002.

[3] Y. Xu and Y. Papakonstantinou: Efficient Keyword Search for Smallest LCAS in XML Databases. In SIGMOD: pp. 527-538, 2005 .

[4] Guo, L., et al., "XRANK: Ranked Keyword Search over XML Documents," In Proc. of ACM SIGMOD Conference, pp. 16-27, 2003.