XML 문서에서 효율적인 키워드 검색을 위한 원소의 의미 정보 이용 검색 기법

김종진*, 김재형*, 이승미**, 손진현*
*한양대학교 컴퓨터공학과
**한양대학교 컴퓨터공학과 BK21 AIS 사업팀
{jjkim,jhkim,smlee}@database.hanyang.ac.kr, jhson@hanyang.ac.kr

A Search Scheme using Semantic Information of the Element for the Efficient Keyword Search in XML Documents

*Jong-Jin Kim, *Jae-Hyung Kim, **Seung-Mi Lee, *Jin-Hyun Son
*Dept. of Computer Science and Engineering, Hanyang University
**BK21 AIS Team, Dept. of Computer Science and Engineering, Hanyang University

요 약

저장하고 표현해야 할 정보의 종류가 많아지고 그 양이 증가함에 따라 사용자는 자신이 원하는 정보를 찾기 위해 정보 검색의 과정을 수행한다. 하지만 전통적인 검색 방법은 해당 사용자가 이미 정보의 표현 방법, 즉 스키마를 모두 알고 있다는 가정 하에 진행되어 진다. 키워드 검색 기법은 사용자에게 내부적인 스키마 구조를 숨기고 단지 몇 개의 키워드로 사용자에게 원하는 정보를 검색해 준다. 본 논문에서는 XML 문서 안의 정보를 찾는 새로운 키워드 검색 기법을 제안한다. 제안하는 방법으로 각 XML 문서 원소가 표현하는 의미 정보를 최대한 활용하여 사용자가 원하는 정보를 검색해 줄 수 있는 인덱스를 구축하고 이를 통해 사용자가 알고자 하는 검색의 목표가 무엇인지 그 의미도 파악할 수 있도록 지원한다. 구축된 인덱스를 바탕으로 검색에 수행되는 XML 원소 비교의 횟수를 줄여서 전체 검색 성능을 향상 시킬 수 있도록 한다.

1. 서론

본래 DB(Database System)에서의 정보 검색은 정보를 표현하는 구조 정보, 즉 스키마를 알고 있는 상태에서 이뤄진다. 하지만 IR(Information Retrieval)의 정보 검색 개념이 추가되면서 내부 스키마 정보를 모르는 상태에서 데이터를 검색할 수 있도록 한다[1]. 이러한 기술이 요구되는 이유는 취급해야 하는 정보의 양이 점점 많아져서 그 정보를 표현하는 스키마를 모두 이해하는 것이 불가능하고 이러한 상황에서 사용자는 자신이 원하는 정보를 더욱 간단한 방법으로 찾고자 하기 때문이다. 키워드 검색은 관계형 데이터베이스(RDB), 관계형 스트리밍 데이터, XML 문서, Web Information, P2P 등 다양한 분야에서 사용되며 분야별로 검색의 결과 향상을 위해 많은 연구가 진행 중이다. 이 중에서 본 논문은 XML 문서에서 이뤄지는 키워드 검색을 효율적으로 처리하는 방법을 다룬다. XPath[3]나 XQuery[4]와 같은 별도의 XML 문서를 검색할 수 있는 질의 언어 기술이 연구되어 있지만 사용자에게 복잡한 질의 문법을 익히고 XML 스키마 구조 또한 파악할 것을 요구하기 때문에 일반적인 사용자에게는 알맞지 않다.

본 논문에서는 XML 문서의 각 원소간의 관계를 새로운 관점에서 접근하여 그 관계 표현을 위한 원소 번호 부여 기법을 기존의 방법을 수정하여 제안하고 그것을 관계형 데이터베이스 상에 저장할 수 있는 방법을 구상한다. 그리고 예제를 통해 실제로 어떻게 동작하는지 살펴보도록 한다.

논문 구성은 다음과 같다. 2장에서는 본 논문과 관련되고 내용을 뒷받침 해 줄 수 있는 연구들에 대해 알아본다. 3장에서는 본 논문에서 제안하는 원소의 정보 비교 검색 모델의 정의와 의미를 설명하고 XML 원소의 정보를 활용한 비교 검색 기법을 제안한다. 4장에서 본 논문에서 제안하는 기법을 예제를 통해 자세하게 설명한다. 그리고 마지막으로 5장에서는 결론을 맺고 향후 연구 계획을 기술한다.

본 연구(논문)는 산업자원부 지원으로 수행하는 21세기 프론티어 연구개발사업(인간기능 생활지원 지능로봇 기술개발사업)의 일환으로 수행되었습니다.
이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No.R01-2007-000-20135-0).

2. 관련 연구

2.1 원소 번호 부여 기법

XML 문서는 기본적으로 각 원소(Element)와 속성(Attribute)을 트리의 노드로 하는 트리노드형태로 표현이 된다. XML의 ID/IDREF 속성을 이용해서 그래프형태로 표현하여 키워드 검색을 수행한 방법[6]도 있지만 트리 형태나 그래프 형태 모두 XML의 원소 또는 속성 정보로 인덱스를 구성하게 된다. 이 때, 각 원소를 식별하는 방법으로 잘 알려진 듀이 넘버링(Dewey node labeling)방법을 기초로 각 원소를 식별한다[5,6,11]. 그 외에 각 원소의 접미사 정보를 활용한 식별 방법[8]은 XML 문서 내의 구조적인 관계를 원소 이름의 접미사로 구성된 트리로 표현하는 기법으로 각 원소를 검색할 때 불필요한 조인 연산을 줄이는 방법이다.

넘버링 기법의 장점은 XML 문서별로 원소에 번호를 부여하여 관리하므로 구조가 다른 XML문서를 쉽게 삽입할 수 있으며, RDB에 저장하는 것이 용이하고 원소의 노드 정보를 통해서 두 원소가 주어졌을 때 부모-자식 관계를 파악할 수 있어 전체 XML문서를 살피지 않아도 원소 번호를 통해 구조 정보의 파악이 가능하다는 것이다. 반면에 질의에 포함된 원소의 개수가 많을수록 인덱스에 공간낭비를 초래하는 단점이 있다. 특히, 듀이 넘버링 기법은 XML 문서 중간에 원소를 임의로 추가하거나 삭제할 때 전체 원소에 대한 넘버링을 다시 해야

하며 XML 문서를 트리로 표현했을 때 트리 노드의 깊이가 깊어질수록 표현해야할 듀이 넘버가 많아지는 단점이 있다. 듀이 넘버링 기법의 단점을 극복하기 위한 방법으로 Durable[9], XISS[10] 번호 부여 기법이 있지만, 역시 XML 자료의 양이 많고 질의에 포함된 원소의 수가 많을수록 검색 성능은 떨어지게 되며, 데이터(원소나 속성)값의 비교 연산이 포함되면 그 성능은 더욱 떨어진다.

2.2 XML 문서에서의 키워드 검색

반구조적(Semi-Structured)인 특성을 지니는 XML 문서는 표현하고자 하는 데이터가 같아도 일정하게 고정된 스키마를 따르지 않는다. 그래서 여러 가지 구조를 갖는 같은 의미의 XML 문서가 존재할 가능성이 높다. 그래서 일반적인 방법의 XML 원소 매칭방법은 모든 XML 문서에 적용되지 못할 수도 있다. XML 키워드 검색 방법과 기존의 일반 텍스트 문서 검색 방법의 가장 큰 차이점은 검색 결과에 있는데, 기존의 일반 문서 검색의 경우 검색 결과로 질의어가 포함된 문서 전체를 반환하는 반면에 XML 키워드 검색 결과는 질의어와 관련된 보다 작은 단위인 특정 부분을 결과값으로 반환하여 상대적으로 정확한 검색 결과를 제공할 수 있다. 보통 XML 키워드 검색의 결과값은 XML의 구조 정보를 활용한 트리 형태로 변환되며 질의한 모든 키워드를 포함하는 최소공통조상(Lowest Common Ancestor) 노드를 반환하는 것이 일반적이다. 관련된 연구로는 XML 문서에서 ID/REF 속성을 이용하여 랭킹 기법을 적용한 XRANK [6], 가장 작은 LCA를 반환하는 SLCA [14], 가장 의미 있는 LCA를 반환하는 MLCA [15], 일반 사용자도 쉽게 접근할 수 있도록 하는 질의 언어를 제공하는 XSEarch [13] 등이 대표적이다. XML 문서는 또한 구조 정보를 의미 정보로도 활용이 가능하기 때문에 XSeek 등의 의미 검색도 고려한 연구가 있다[5]. 하지만 의미 검색을 위해선 키워드 검색이 지정된 XML 스키마에 한정되거나 의미 검색을 위한 온톨로지를 구성해야 하는 별도의 전처리 단계가 필요하다[7].

3. 원소 비교 검색 모델

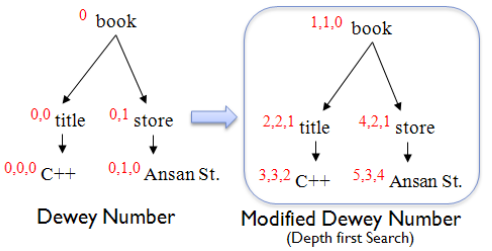
3.1 XML 저장 및 표현 형태

보통 XPath나 XQuery 기술을 활용하기 위해서는 XML 문서 자체를 메모리에 DOM[2] 형태로 저장하여 처리하게 된다. 하지만 XML DOM의 경우 많은 양의 메모리와 처리 시간을 요하기 때문에 대용량의 XML 문서에는 적합하지 않다. 때문에 RDB에 XML문서를 저장하여 처리하는 방법이 연구되고 있는데, 이 경우 XML의 구조 정보를 해석해서 RDB로 저장하기 위한 별도의 소프트웨어가 요구된다[11,12]. 본 논문에서는 XML DOM을 사용하여 XML을 모델링 하지 않고 RDBMS에 XML문서를 저장하여 처리한다. 근래의 RDBMS는 여러 가지의 XML 문서 저장을 위한 방법을 제공한다. RDB에 저장된 XML 문서는 DBMS에 의해서 바로 XPath나 XQuery가 지원된다. 하지만 XML DOM을 DBMS에서도 사용하기 때문에 어떤 방식으로 XML 문서를 저장하느냐에 따라서 성능차이가 많이 난다. 때문에 XML 문서의 사용 용도를 고려하여 적절한 저장방법으로 DB에 저장하여야 한다[12].

XML 문서는 원소, 속성, 값(Content or Value)으로 구성되는데 이 정보를 바탕으로 트리 형태로 표현이 가능하다. 이 때 속성도 원소와 같이 트리의 노드 형태로 표현한다. XML 문서가 트리형태로 표현될 때 XML의 원소가 트리의 노드와 일치하기 때문에 지금부터는 원소와 노드를 같은 의미로 간주한다.

3.2 수정된 듀이 넘버링 기법

본 논문에서는 듀이 넘버링 기법을 기초로 하는 원소 넘버링 기법을 사용하게 되는데 넘버링 기법의 장점을 유지하면서 듀이 넘버링 기법의 단점을 극복할 수 있는 수정된 듀이 넘버(Modified Dewey Number)를 제안한다. 임의 노드 추가/삭제 시 이미 넘버링이 된 다른 노드와 독립적으로 적용될 수 있도록 하고 XML 트리의 노드 깊이가 증가함에 따라 점점커지는 듀이 넘버를 수정하여 인덱스 구성에 공간 효율을 높이고 듀이 넘버 비교 검색 시 비교 연산의 횟수를 줄여서 빠른 연산을 수행할 수 있도록 한다.



(그림 1) 수정된 듀이 넘버

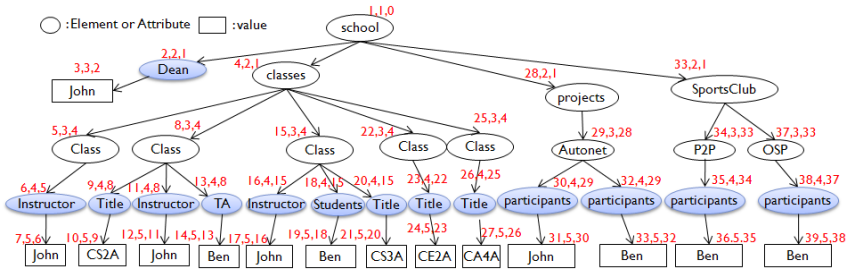
듀이 넘버의 가장 큰 특징은 (a) 중복되지 않으면서 각 원소를 정확하게 식별한다는 것과 (b) 노드 넘버 자체가 노드의 위치 정보를 가지고 있다는 것이다. 노드의 부모-자식 관계와 형제 노드 관계 그리고 트리에서 깊이 정보까지 노드 넘버를 통해 알 수 있다. MDN의 형태는 <노드번호, 노드깊이, 부모노드번호>의 키 값으로 구성된다. 위의 (a)를 위해서 기본적으로 모든 노드에 고유의 숫자 값인 노드번호를 할당한다. 그리고 (b) 정보를 유지하기 위해 노드깊이, 부모노드번호 정보를 유지한다.

(그림 1)에서 사각형 안의 트리노드의 윗첨자 숫자가 바로 MDN이 된다. MDN은 고유의 식별 번호가 부여되기 때문에 임의의 노드 추가/삭제에도 다른 노드 넘버에 영향을 끼치지 않는다. 또한 3쌍의 수 정보가 트리 노드의 깊이에 상관없이 항상 유지되므로 노드 넘버에 의한 인덱스 공간 낭비가 초래되지 않고 듀이 넘버의 장점인 부모-자식 관계 및 형제 관계를 확인할 수 있다.

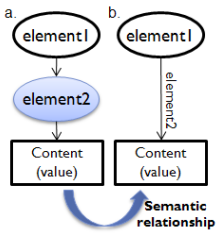
3.3 XML 원소의 의미 관계

XML 원소가 표현하는 정보를 통해 검색 질의의 의미를 파악할 수 있다[5,7]. 일반적인 XML 문서에서 (그림 3-a)와 같은 구조를 발견할 수 있는데, 이것을 (그림 3-b)와 같이 element1과 Content는 element2의 관계를 가지고 있다고 해석 할 수 있다. 예를 들어, (그림 2)에서 school{1,1,0} - Dean{2,2,1} - John{3,3,2}을 살펴보면 School의 Dean은 John이다 라는 관계가 성립하는 것을 알 수 있다. 두 노드의 관계를 표현해 주는 Dean{2,2,1}과 같은 중간 노드를 Mid-Element(정의 1)라고 하자. 모든 노드의 집합은 N으로 표기하고 각 노드는 n_i로 (i는 노드 고유번호) 표기된다. 예를 들어, (그림 2)에서 Dean{2,2,1} 노드의 노드 고유번호는 2이기 때문에 Dean 노드는 n_2 로 표기 된다.

***정의 1**: (Mid-Element) 세 노드 n_a, n_b, $n_c \in N$를 고려하자. n_c의 부모 노드가 n_b이고 n_b의 부모노드가 n_a 일 때, n_a, n_b는 원소를 표현한 것이고 n_c는 n_b 의 값을 표현한 것이면 n_a 와 n_c 사이에는 n_b 의 관계가 성립한다. 이때, n_b를 Mid-Element라 한다.*



(그림 2) MDN으로 노드 라벨링된 XML 트리



(그림 3) 원소 의미 관계

(그림 2)에서 Mid-Element는 음영 처리해서 나타냈다. (그림 4)는 Mid-Element를 고려하여 (그림 2)의 XML 트리를 재구성한 것이다.

3.4 검색 모델

검색 질의에 Mid-Element가 포함될 경우 Mid-Element는 의미 정보이기 때문에 검색 결과에 당연히 포함하는 것으로 하여 Mid-Element의 부모와 자식 노드만 비교 대상으로 고려한다. 예를 들어 "Dean" 이라는 키워드가 검색 질의에 포함되게 되면 다른 비교 연산 없이 바로 Dean{2,2,1}의 부모와 자식 노드를 검색 결과 대상으로 고려한다. 동시에 Dean{2,2,1}의 부모와 자식 노드에 "Dean"의 의미정보를 부여한다.

실제로 XML 트리를 구성 할 때 (그림 4)에서와 같이 Mid-Element를 트리의 Edge에 표현하지는 않는다. 하지만 노드가 질의 키워드에 속할 때 Mid-Element의 여부를 판단할 수 있도록 하는 정보는 <표 1>로 표현된 노드 관계 인덱스(Node Relationship Index)에 유지해서 키워드 검색을 처리할 때 비교 연산을 수행해야 할 노드 개수를 Mid-Element의 개수만큼 줄일 수 있도록 한다.

각 노드를 탐색하기 위해서는 부모노드를 판단할 수 있는 *parent()* 함수(정의 2)와 질의된 키워드와 매칭되는 후보 LCA 노드를 판단할 수 있는 *canlca()* 함수(정의 3)가 필요하다. 각 함수는 NRI의 인덱스 정보를 참조하며 NRI는 <표 1>에서 살펴 볼 수 있다.

정의 2: *(parent()) 두 노드 n_a, n_b ∈ N를 고려하자. (n_a.노드번호 = n_b.부모노드) 면 n_a가 n_b의 부모 노드임이 증명된다. parent(n_b)는 n_b 의 MDN에서 n_b.부모노드 를 반환한다. 만약 NRI를 통해 질의된 노드가 MId-Element 이면 parent(Mid-Element)를 호출하여 반환한다.*

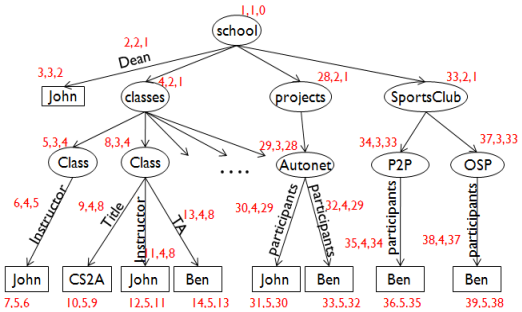
parent() 함수의 중요한 역할 중에 하나는 Mid-Element 가 감지되면 *parent(Mid-Element)*를 호출하여 반환하는 것이다. 원소가 Mid-Element라는 정보를 통해 전체 연산의 횟수를 줄이는 방향이 본 논문에서 제안하는 기법의 핵심이라고 볼 수 있다.

정의 3: *(canlca()) 노드 n_a ∈ N를 고려하자. canlca(n_a)는 n_a의 노드 Type 정보를 고려한다. n_a가 value이거나 Mid-Element일 때, 그리고 입력 키워드가 하나일 경우 canlca(n_a) = parent(n_a) 이며, 그 외의 경우 canlca(n_a) = ∅.*

canlca() 함수는 질의 키워드를 그 인자 값으로 한다. LCA는 주어진 키워드를 모두 포함하는 가장 작은 공통 선조 노드를 지칭하기 때문에 *canlca()* 함수는 본 논문에서 간단하지만 효과적으로 사용된다. *canlca()*를 통해서 구한 후보 LCA 중 입력된 키워드와의 일치 여부를 따지면 2차적으로 질의 검색어와 연관성이 높은 LCA를 선별할 수 있다. 4장에서 예제를 통해 살펴본다.

모든 후보 LCA 노드가 고려되었다면 후보 LCA 중에서 사용자가 원하는 가장 의미 있는 LCA(Most-Meaningful LCA, 이하 MMLCA)를 선별해 내야 한다. MMLCA는 *mmlca()* 함수를 수행하여 얻어 낸다.

정의 4 *: (mmlca()) 후보 LCA들의 노드를 n_a, n_b,…, n_i ∈ N (i >= 질의 검색어 개수)라고 하자. mmlca(lca node list)는 {n_a.부모노드, n_b.부모노드, …, n_i.부모노드} 리스트에서 공통 부모 노드를 찾는다. 가장 많은 중복이 있는 노드를 최종적으로 반환한다.*



(그림 4) Mid-Element를 고려한 XML 트리

3.5 제안하는 기법을 위한 인덱스

XML 키워드 검색에 활용되는 인덱스는 일반적으로 역인덱스 구조를 따른다. XML 문서가 트리 구조로 표현이 되기 때문에 트리의 깊이가 얕은 쪽에 위치한 노드가 질의 검색의 키워드로 등장하는 경우는 깊이가 깊은 노드보다 상대적으로 적다. 즉, 개체수가 많은 자식 노드로 갈수록 검색 질의에 출현하는 빈도수가 높아지게 되므로 인덱스 구축 시 인덱스 검색 방향을 고려하여 빈도수가 많은 자식 노드부터 인덱스를 작성하면 인덱스 검색 성능 향상을 기대할 수 있다.

<표 1> 노드 관계 인덱스(NRI)

Node	Type	MDNs
school	element	(1,1,0)
Dean	mid-element	(2,2,1)
…	…	…
Ben	value	(14,5,13),(39,5,38),(33,5,35)

<표 1>에 제안하는 인덱스를 (그림 2) XML 트리의 일부분을 표현하여 테이블 형태로 표현했다. 인덱스의 Type 속성을 통해서 질의에 있는 키워드의 정보를 판단할 수 있다. MDNs 속성은 해당 노드를 식별할 수 있는 정보로 MDN 정보를 리

스트로 가지고 있다. Type 속성은 Element, Mid-Element, Value(원소의 값, 속성의 값 모두 포함)로 구성된다.

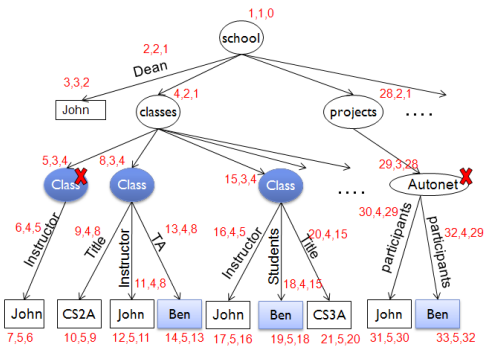
4. 원소 비교 검색 기법

본 장에서는 3장에서의 정의를 바탕으로 예제를 통해서 원소 비교 검색을 수행하는 과정을 살펴본다. 예제에 사용되는 XML은 (그림 2)의 XML 트리를 통해서 설명한다.

질의 : *Query(Ben, Class)*

Step 1 *: canlca() 함수를 각각의 질의 키워드 리스트에 대해서 수행한다.*

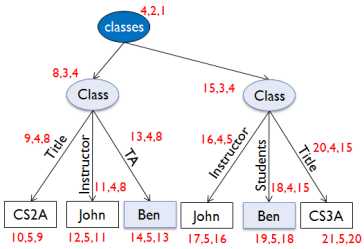
Step 1 수행 시 (Ben, Class) 키워드와 일치하는 문서상의 모든 노드가 LCA로 고려되게 된다. (그림 5)에서 확인할 수 있다. 하지만 정의 3 에 의해 class{5,3,4} 는 LCA에 후보에서 제외되고 Autonet{29,3,28}은 질의된 키워드를 모두 포함하지 않는 LCA이기 때문에 후보에서 제외된다. P2P{34,3,33}, OSP{37,3,33} 두 노드 또한 마찬가지로 후보에서 제외된다. 결국 후보 LCA로 class{8,3,4}, class{15,3,4} 가 남게 된다.



(그림 5) 질의(ben, class)를 처리하는 과정

Step 2 *: mmlca()를 후보 LCA 대상으로 수행한다.*

Step 2 과정을 통해서 MMLCA를 후보 LCA에서 선별하게 되는데 이 때 두 후보 노드의 LCA로 Classes{4,2,1}가 mmlca()의 결과로 반환된다. (그림 6)에서 보이는 트리의 루트 노드가 된다.



(그림 6) 질의(ben, class)의 MMCLA 트리

Step 3 *: MMLCA를 루트 노드로 하고 canlca()를 포함하는 서브트리를 최종 결과값으로 사용자에게 반환한다.*

Step 3 과정은 키워드 검색 질의 처리 단계의 마지막 단계로 MMLCA와 후보 LCA를 모두 포함하는 서브 트리 형태의 결과를 반환한다. NRI를 통해 후보 LCA의 자식 노드들을 파악할 수 있기 때문에 최종 결과는 (그림 6)과 같다. XML 문서의 노드깊이가 깊은 경우 위의 Step 1과정과 Step 2 과정을 반복 수행하여 결과값을 도출한다.

5. 결론 및 향후 연구

XML 키워드 검색에서 사용되는 기본적인 노드 라벨링 기법을 분석하고 그 단점을 극복할 수 있는 새로운 넘버링 스킴을 제안했다. XML 문서에 키워드 질의를 수행할 때 필요한 노드 비교 연산을 줄이기 위해서 XML 원소의 정보를 활용하여 원소의 의미 관계를 파악하고 Mid-Element를 추출하여 의미 검색이 가능하도록 하였다.

본 연구에서는 XML 네임 스페이스(Name space)가 고려되지 않았다. 네임 스페이스를 고려하여 여러 XML 문서에서 제안하는 키워드 검색을 적용할 수 있는 방안을 향후 연구 내용으로 삼는다. 또한 다양한 XML 스키마의 문서에서도 제안하는 기법이 효율적으로 적용됨을 확인할 수 있는 실험이 고려되도록 한다.

참고문헌

[1] G Weikum, “DB&IR: both sides now”, In Proc. of the 27th ACM SIGMOD, pp. 25-30, 2007.

[2] Document Object Model (DOM), http://www.w3.org/DOM/

[3] XML Path Language(XPath), http://www.w3.org/TR/xpath

[4] An XML Query Language(XQuery 1.0), http://www.w3.org/TR/xquery/

[5] Z Liu, J Walker, Y Chen, "XSeek: a semantic XML search engine using keywords", In Proc. of the 33rd VLDB, pp.1330- 133, 2007.

[6] L Guo, F Shao, C Botev, J Shanmugasundaram, "XRANK: ranked keyword search over XML documents", In Proc. of the 23th ACM SIGMOD, pp. 16-27, 2003.

[7] 이형동, 김성진, 김형주, "의미 기반의 XML 키워드 검색을 위한 효율적인 인덱스 구조", 한국정보과학회, 정보과학회논문지 : 데이타베이스 제33권 제5호, pp. 513-525 (13pages), 2006.

[8] H. Wang, S. Park, W. Fan, P. S. Yu, “ViST: A Dynamic Index Method for Querying XML Data by Tree Structures”, In Proc. of the 23rd ACM SIGMOD, pp.110-121, 2003.

[9] Quanzhong Li, Bongki Moon, “Indexing and Querying XML Data for Regular Path Expressions”, In Proc. of the 27th VLDB, pp.361-370, 2001.

[10] P. Harding, Q. Li and B. Moon, "XISS/R:XML Indexing and Storage System Using RDBMS ", In Proc. of the 29th VLDB, pp. 1073-1076, 2003.

[11] T. Igor, D. V. Stratis, B. Kevin, S. Jayavel, S. Eugene and Z. Chun: “Storing and querying ordered XML using a relational database system”, In Proc. of the 22th ACM SIGMOD, 2002.

[12] J. Garmany, S. Karam, V.J. Jain, Lutz Hartmann, V. J. Jain, B. Carr, "Oracle 11g New Features", Chapter 2, Rampant techpress.

[13] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv., “XSEarch: A Semantic Search Engine for XML”, In Proc. of the VLDB, 2003.

[14] Y. Xu and Y. Papakonstantinou., "Efficient Keyword Search for Smallest LCAs in XML Databases", In Proc. of the SIGMOD, 2005.

[15] Y. Li, C. Yu, and H. V. Jagadish., "Schema-Free XQuery". In Proc. of the VLDB, 2004.