XML 문서에서 엘리먼트 타입을 이용한 구조적 검색 기법의 설계[1)]

김성완, 정헌석[*2)], 이재호[**], 임해철[*]
삼육의명대학 컴퓨터정보과, [*]홍익대학교 컴퓨터공학과, [**]경인교육대학교 컴퓨터교육과
swkim@syu.ac.kr, jhlee@gin.ac.kr, lim@cs.hongik.ac.kr

Design of Structural Retrieval Scheme Using Element Type in XML Documents

Sung Wan Kim, Hun Suk Chung[*2)], Jaeho Lee[**], Hae Chull Lim[*]
Dept. of Computer Information, Sahm Yook College,
[*]Dept. of Computer Eng., Hong Ik Univ., [**]Dept. of Computer Education, Gyeongin Nat'l Univ. of Education

요 약

XML 문서의 검색을 위한 많은 연구들이 수행되고 있지만 순수하게 구조적 관계성만을 대상으로 하는 검색 즉, 구조적 검색 처리 기법에 대해서는 많이 다루지 않고 있거나 XML 문서 트리에 대한 반복적인 순회를 기반으로 처리하는 방법들이 제안되었다. 또한, 사용자가 원하지 않는 엘리먼트들을 제외하기 위해서는 부가적인 필터링 과정을 필요로 한다. 한편, 대부분의 XML 문서의 검색 관련 연구들은 엘리먼트의 삽입 또는 삭제 등 XML 문서의 부분적인 갱신 및 변경이 발생하는 환경을 고려하지 않고 있다. 본 논문에서는 사용자로부터 주어지는 질의에 포함된 엘리먼트 타입 정보 이용하여 XML 문서 트리에 대한 순회를 없애거나 최소화시키고, 필터링 과정도 필요로 하지 않는 구조적 검색 기법을 설계한다. 또한, 엘리먼트의 삭제 및 삽입 등 동적인 변경에 빠르고 유연하게 대처할 수 있는 인덱스 구조를 설계하고 이를 기반으로 구조적 검색 질의의 주요 유형에 대한 처리 방안을 예를 들어 설명한다.

1. 서 론

인터넷의 발전과 XML의 등장으로 인해 XML(eXtensible Markup Language) 문서의 인덱싱 및 검색에 대한 연구가 활발히 진행되어 왔다. 먼저 데이터베이스 분야에서 XML 문서의 저장 및 검색에 대한 연구[3, 6]가 진행되어 왔으며, 최근에는 정보검색 분야에서도 XML 문서를 구조화 문서(structured document)의 인스턴스로 해석하고 전통적인 정보검색 기법들을 적용하는 연구[4, 8, 9]가 진행되고 있다. 특히, XML 문서에 대한 인덱싱 및 검색 기법들은 구조적 특징을 고려한 내용 검색이 많은 부분을 차지하고 있다.

이러한 연구들에서는 주로 내용 검색의 정확도를 높이기 위해 구조적 정보를 이용하는 방법을 제시하고 있으나, 순수하게 구조만을 대상으로 하는 처리 방법에 대해서는 크게 언급되지 않고 있다. 본 논문에서는 키워드 등의 내용 요소 없이 순수하게 구조적 엘리먼트에 대한 결합만으로 구성된 검색을 '*구조적 검색(structural retrieval)*'이라고 정의한다. 이러한 구조적 검색 질의의 기본적인 처리 방법은 상향식 또는 하향식 트리 순회를 기반으로 하는 반복적 혹은 재귀적 처리를 필요로 한다[3, 6, 10]. 그러나 이 경우 트리 전체를 순회하여야 하기 때문에 처리 영역이 넓게 되어 효율성이 떨어진다. 또한, [4, 7]의 연구들에서는 XML 문서를 트리 형태로 표현하고, 각 노드로 매핑된 엘리먼트에 구조적 정보를 포함하는 특별한 식별자를 부여하여 트리의 순회 없이 구조적 검색 질의 처리가 가능하도록 설계하였다. 그러나 일부 구조적 검색 질의 유형에는 효율적이나 다른 유형의 처리를 위해서 추가적인 연산 및 반복적 접근인 요구되는 한편 부가적인 필터링 과정이 요구된다.

한편, XML 문서에 대한 엘리먼트 삽입 및 삭제와 같은 구조적인 변경이 자주 발생하는 환경에서는 위의 방법들은 구조적 정보를 포함하는 엘리먼트 식별자에 대한 많은 재할당이 요구되므로 동적인 환경에서는 매우 부적합하다[9]. 따라서, 본 논문에서는 XML 문서에 대해 엘리먼트의 삽입 및 삭제 등 동적인 부분 변경을 빠르게 처리하고, 구조적 검색 질의를 엘리먼트 타입 정보를 이용하여 트리의 순회 없이 또는 최소화하여 처리하기 위한 기법을 설계한다. 2장에서는 관련 연구를, 3장에서는 본 논문에서 제안하는 인덱스 구조에 대해 설명한다. 4장에서는 엘리먼트 타입 정보를 이용한 구조적 검색 질의 처리 전략에 대해 설명한다. 5장에서는 결론을 맺는다.

2. 관련 연구

구조적 검색은 그 처리 방향성에 따라 조상/후손, 부모/자식 검색과 같은 계층적(수직적) 관계에 기반한 질의와 형제 검색과 같은 수평적 관계에 기반한 질의로 분류될 수 있으며, 이외에 순서 또는 레벨 등의 부가적 조건을 포함하여 상세 분류될 수 있다[5, 10]. 구조적 검색 질의의 대상은 문서 또는 엘리먼트가 될 수 있으나, 본 논문에서는 XML 문서에서의 최소 검색 단위인 엘리먼트만을 대상으로 한다. 또한 간편성을 위해 텍스트, 링크 정보와 애트리뷰트는 본 논문에서 다루지 않는다.

구조적 검색 질의는 XPath와 같은 경로식을 이용하여 표현할 수 있다[3, 6, 12]. Xpath의 목적은 단일 XML 문서내의 엘리먼트와 같은 부분에 대한 접근을 제공하는 것이다. 일반적인 구조적 문서를 위한 정보 검색 시스템에서는 문서 컬렉션 전체를 대상으로 특정 조건을 만족하는 문서 또는 엘리먼트를 검색해야 하므로 XPath를 기반으로 하는 XML 질의어의 확장이 필요하며, 향후 XPath의 요구사항에 포함되어 있다[12].

한편, 구조적 검색 질의의 형태는 '*<검색 기준 엘리먼트> 방향 조건<검색 대상 엘리먼트>*'와 같은 표현으로 주어질 수

1) 본 연구는 한국과학재단 목적기초연구(과제번호:R01-2002-000-00080-0) 사업의 지원으로 수행되었음.

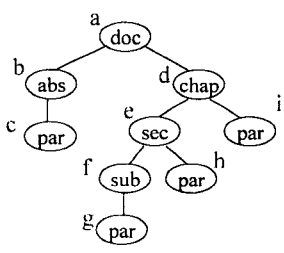
2) 2003년 3월부터 극동대학교 정보통신학부 전임강사로 재직중임

있다. 검색 조건 엘리먼트는 질의의 시작이 되는 엘리먼트이며, 검색 대상 엘리먼트는 질의의 최종 타겟이 되는 엘리먼트이다. 방향 조건은 부모/자식, 조상/후손, 형제 등이 된다.

이러한 구조적 검색 질의는 우선 XML 문서에 대해 각 엘리먼트를 노드로 매핑한 트리 형태로 표현 후 각 노드에 식별자를 할당하고 인덱스를 구축하여 주어진 경로식 조건에 따라 XML 문서 트리를 상향식 또는 하향식 트리 순회를 기반 처리할 수 있으며[3]. 이는 반복적 혹은 재귀적 처리를 필요로 한다. 그러나 이 경우 트리 전체를 순회하여야 하기 때문에 처리 영역이 넓게 되며, 대상 엘리먼트가 아닌 엘리먼트들을 제거하기 위한 추가적인 필터링 과정이 필요하는 등 효율성이 떨어진다. Lore 시스템에서는 루트부터 시작하는 단순 경로들에 대한 구조적 요약 정보를 유지하는 DataGuide[2]를 통해 주어진 경로식을 통해 도달할 수 있는 모든 엘리먼트들을 유지함으로 드리를 직접 순회하지 않고 해당 엘리먼트만을 접근할 수 있도록 하고 있으나, 정규 경로식 등을 포함하거나 루트가 아닌 엘리먼트로부터 시작되는 질의의 처리에 대해서는 언급하지 않고 있다.

또한, [4]연구에서는 트리로 표현된 XML 문서의 각 노드에 구조적 정보를 포함하는 특별한 식별자를 부여하여 트리의 순회 없이 구조적 검색 질의 처리가 가능하도록 설계하였다. 그러나 일부 구조적 검색 질의 유형에는 효율적이나 다른 유형의 처리를 위해서 추가적인 연산 및 반복적 접근인 요구된다. 또한, 한 문서내의 엘리먼트에 대한 식별자 값을 주고 처리하는 방안에 대해서만 언급되고 문서 컬렉션에 대한 처리는 다루고 있지 않다. [7]의 논문에서는 임의의 두 엘리먼트의 이름이 주어진 경우에만 두 엘리먼트간의 자손/후손 관계를 구조적 정보를 갖고 있는 식별자를 통해 빠르게 계산할 수 있으나, 부모/자식의 구별이 안되며, 엘리먼트 이름이 하나만 주어진 경우, 형제 검색, 부모 및 자식 검색은 반복적 접근방식을 취하고 있다. 또한, 엘리먼트 이름을 기반으로 검색이 처리되므로, 사용자가 원하는 구조적 조건에 부합하지 않는 엘리먼트들도 처리 대상에 포함되므로 필터링 과정이 항상 요구된다.

또한, [4, 7]의 방법은 XML 문서에 대한 엘리먼트 삽입 및 삭제와 같은 구조적 변경이 자주 발생하는 환경에서는 구조적 정보를 포함하는 엘리먼트 식별자에 대해 많은 재계산이 요구되므로 동적인 환경에서는 매우 부적합하다. 한편, 현재까지 연구된 XQL, XQuery 등과 같은 대부분의 XML 문서 조작 언어들은 XML 문서에 대한 검색만을 위주로 설계된 질의어들이다. 그러나 XML이 완전한 데이터 교환 포맷으로 되기 위해서는 검색 위주의 질의뿐만 아니라 내용 갱신 및 구조 갱신 등 갱신 연산도 지원해야 할 것이다. 최근에 이러한 XML에 대한 동적 갱신 연산의 필요성이 인식됨에 따라 관련 연구가 시도[11]되고 있으며, W3C에서 규정한 XML 표준 질의어인 XQuery에서도 갱신 측면에 대한 지원을 차후 버전의 요구사항으로 언급하고 있다[12].



단순절대경로	ETY
/doc	a
/doc/abs	b
/doc/abs/par	c
/doc/chap	d
/doc/chap/sec	e
/doc/chap/sec/sub	f
/doc/chap/sec/sub/par	g
/doc/chap/sec/par	h
/doc/chap/par	i

<그림 1> 논리적 구조와 경로 정보 테이블

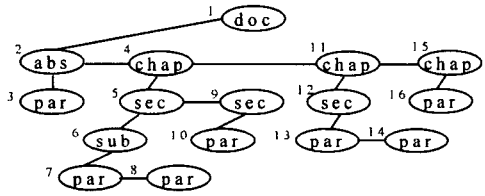
3. 인덱스의 생성

3.1 경로 정보 테이블의 구성

주어진 DTD 등을 분석하여 <그림 1>의 왼쪽처럼 명세된 논리적 구조 정보에 나타나는 단순 절대 경로들을 바탕으로 <그림 1>의 오른편과 같이 *경로 정보 테이블(Path Information Table)*을 구성한다. 여기서 엘리먼트 타입 즉, ETY(Element Type)는 각 경로마다 고유하게 할당된 값으로 해당 경로의 마지막 엘리먼트에 대한 유일한 식별자 값을 의미한다.

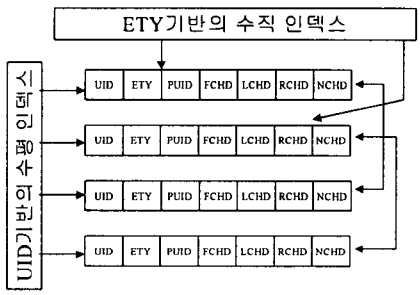
3.2 XML 문서 모델링 및 인덱싱

설명을 단순화 시키기 위해 하나의 XML 문서만을 가정하자. 먼저 소스 XML 문서를 <그림 2>와 같이 왼쪽 자식 오른쪽 형제의 구조를 갖는 이진 트리 형태[1]로 표현하고, 각 엘리먼트에 임의의 고유한 식별자(UID)를 할당한다. 그리고 각 엘리먼트에 대해 <UID, ETY, PUID, FCHD, LCHD, RCHD, NCHD>로 구성된 엔트리로 매핑한다. 여기서 PUID, FCHD, LCHD, RCHD는 각각 부모 엘리먼트의 UID, 첫 번째 자식의 UID, 바로 왼쪽 형제의 UID, 그리고 바로 오른쪽 형제의 UID를 의미한다. NCHD는 자식 엘리먼트의 수를 의미한다. 예를 들어 <그림 2>에서 UID가 4인 'chap' 엘리먼트에 대한 엔트리는 <4, d, 1, 5, 2, 11, 2>로 구성된다.



<그림 3> XML 문서 트리의 예

이렇게 모든 엘리먼트에 대해 생성된 엔트리들을 포스팅으로 매핑하고 UID를 기준으로 역 과정을 통하여 <그림 3>과 같은 역 인덱스를 구성한다. 여기서 주목할 점은 동일한 ETY를 갖는 엔트리 포스팅들을 그룹핑하기 위해 ETY를 키로 하는 새로운 차원의 인덱스 파일을 추가한 것이다. UID를 키로하는 인덱스 파일을 '*수직 인덱스*', ETY를 키로하는 인덱스를 '*수평 인덱스*'라 한다. 즉, 수직 인덱스를 통해 특정 ETY에 대한 모든 엘리먼트만을 직접 접근할 수 있다. 인덱스의 구현으로는 일반적으로 사용되는 B+ 트리 등을 활용할 수 있다.



<그림 4> 제안 인덱스 구조

이렇게 인덱스를 구성할 경우 특정 엘리먼트에 대한 삭제 또는 새로운 엘리먼트의 삽입에 대해 유연하게 대처할 수 있다[10]. 기존의 다른 방법처럼 삭제 또는 삽입된 노드에 대한 오른쪽 형제를 기준으로 하는 서브트리 전체에 대한 UID의 변경

이 발생하지 않는다. 즉, 삭제 또는 삽입이 발생하는 엘리먼트에 직접 연관된 엘리먼트들에 대한 UID만 적절하게 수정하면 되므로 갱신에 대한 처리 시간을 대폭 줄일 수 있다.

4. 구조적 검색 질의 처리 방안

일반적으로 정보 검색 시스템에서 구조적 검색은 특정 엘리먼트에 대한 식별자 값이 아닌 엘리먼트의 이름 및 엘리먼트 타입을 기반으로 표현된다. 본 논문에서는 이러한 사용자로부터 주어진 질의에 포함된 엘리먼트 이름과 타입 정보를 이용하여 트리 순회를 하지 않거나 최소화하여 구조적 검색 질의를 처리할 수 있도록 하였다.

이를 위해 사용자 질의를 적절한 스트링 형태로 표현하고, <그림 1>에서 설계된 경로 정보 테이블의 단순절대경로 필드의 값들과 스트링 매칭을 통해 최종 결과에 포함될 엘리먼트의 타입 값을 추출하여 목표 대상만을 직접 접근하게 함으로써 질의 처리 영역을 축소하였다. 스트링 매칭을 위한 표현에 사용된 표기 중 와일드 카드 문자인 '*'는 0개 이상의 문자를 의미한다. 각 질의의 최종 반환 값은 UID의 집합으로 한다.

4.1 자식 검색 질의

<그림 2>와 같이 트리로 표현된 XML 문서에서 'chap' 엘리먼트의 자식 중 'sec' 엘리먼트를 검색하라는 질의(//chap/sec)는 다음과 같이 처리된다. 여기서 검색 기준 엘리먼트는 'chap'가 되며 검색 대상 엘리먼트는 'sec'이 된다. 또한, 괄호내의 표현은 질의문에 대한 이해를 돕기 위한 XPath에 따른 경로식 표현이다. 먼저 사용자 질의를 '*/chap/sec' 스트링으로 변환하고 <그림1>의 경로 정보 테이블의 각 단순절대경로 필드에 대한 각 엔트리의 스트링 값들과 스트링 매칭을 수행하여 '/chap/sec'으로 끝나는 엔트리들만을 추출하여 후보 ETY 집합을 추출한다(이 예에서는{e}). 추출된 각 ETY 값을 키로하여 대해 수직 인덱스 이용하여 해당 ETY 값을 갖는 포스팅들만을 접근하여 UID들을 최종 결과 집합에 포함시켜 반환한다(이 예에서는 {5, 9, 12}).

결국, 'chap'의 자식이면서 'sec'이 아닌 타입을 갖는 엘리먼트들 즉, 주어진 질의와는 다른 경로상에 있는 엘리먼트들에 대한 접근이나 필터링 과정이 요구되지 않는다.

4.2 후손 검색 질의

'chap' 엘리먼트의 후손 중 모든 'par' 엘리먼트를 검색하라는 질의(//chap/descendant::par)는 다음과 같이 처리된다. 사용자 질의를 '*/chap*/par' 스트링으로 변환하고 경로 정보 테이블에서 스트링 매칭을 통해 후보 ETY 집합 {g, h, i}를 구해낸다. 각 ETY를 키로하여 수직 인덱스를 통해 포스팅들을 접근하여 UID들의 집합 즉, {7, 8}, {10, 13,14}, 그리고 {16}를 추출하고 최종적으로 이들에 대해 합집합 연산을 수행하면 최종 결과 집합인 {7, 8, 10, 13, 14, 16}을 구할 수 있다. 여기서도 후손 검색에 대해 이전 트리 순회 등과 별도의 필터링 과정 없이 목표 엘리먼트 타입에 대한 엘리먼트들만을 직접 접근하여 결과를 구하게 된다.

4.3 부모 검색 질의

'par' 엘리먼트의 부모 중 'sec' 엘리먼트를 검색하라는 질의(//par/parent::sec)는 다음과 같이 처리된다. 'par' 엘리먼트를 자식으로 갖고 있는 'sec' 엘리먼트를 검색하라는 질의(//sec[child:par])도 이 범주에 속한다.

사용자 질의를 '*/sec/par' 형태로 변환 하고, 경로 정보 테이블과의 스트링 매칭을 통해 ETY 후보 집합 {h}를 추출한다. 그리고 ETY 값을 키로하여 수직 인덱스를 접근하여 연결된 포스팅들을 접근하여 각 포스팅에서 PUID들의 집합인 {9, 12}를 추출하여 반환한다. 이 경우에도 타겟 엘리먼트의 타입에 대한 필터링이 없이 바로 원하는 엘리먼트 타입에 대한 엘리먼트들만을 구해낼 수 있다.

4.4 조상 검색 질의

'par 엘리먼트의 모든 조상 중 'sec' 엘리먼트들 검색하라는 질의(//par/ancestor::sec)는 다음과 같이 처리된다. 먼저 사용자 질의를 '*/sec*/par' 형태로 변환하고, ETY 후보 집합 {g, h}을 추출한다. 그리고 각 ETY 값을 키로하여 수직 인덱스를 통하여 포스팅들을 접근하여 PUID들을 추출하고 결과 집합에 포함시킨다. 추출된 PUID를 키로 수평 인덱스를 이용하여 엔트리 포스팅을 접근하여 PUID 추출하고 결과 집합에 포함시키다. 위의 과정을 sec 레벨까지 반복 수행 후 중지하면 최종 결과 엘리먼트 집합 {5, 4, 1}을 구할 수 있다.

4.5 형제 검색 질의

경로식 표현 자체가 계층적인 수직 관계성만을 정의하므로, 수평적 관계에 대한 형제 질의는 경로식을 이용하여 처리할 수 없다. 우선, 경로식에 대한 스트링 매칭을 통해 검색 기준 엘리먼트 타입에 대한 후보 집합을 추출하고, 포스팅내의 FCHD 추출 후 LCHD 또는 RCHD에 대한 반복적인 접근으로 최종 결과 UID 집합을 구해낸다.

5. 결 론

본 논문에서는 엘리먼트 타입 정보를 포함한 사용자 질의를 적절한 스트링 형태로 변환 후, 경로 정보 테이블과의 스트링 매칭을 통해, 후보 엘리먼트 타입을 추출하여 구조적 검색 질의에 대한 처리 영역을 줄일 수 있는 방법을 제안하였다. 특히, 후손 검색의 경우 여러 개의 하향 경로를 반복적 혹은 재귀적으로 탐색하지 않고 타겟 엘리먼트들만을 접근하여 결과를 구하므로 처리시간을 단축할 수 있다. 본 논문에서 사용된 후보 엘리먼트 추출 접근 방식은 [4, 9]과 같은 논문에서 주어진 목표 엘리먼트 타입을 기반으로 한 내용 기반 질의 처리시 질의 처리 영역을 줄이기 위한 필터링 과정에서 목표 엘리먼트 타입 정보를 구하는 구체적인 대안으로 활용될 수 있다.

<참고문헌>

1.E. Horowitz, et al., Fundamentals of Data Structures in C, Computer Science Press, 1993

2.R. Goldman and J. Widom, DataGuides:Enabling Query Formulation and Optimization in Semistructured Databases, VLDB 1997

3.J. McHugh et. al., Indexing Semistructured Data, Technical Report, Stanford Univ., 1998

4.D. Shin et al., "BUS:An Effective Indexing and Retrieval Scheme in Structured Documents", ACM Digital Libraries, 1998

5.T. Arnold-Moore and R. Sacks-Davis, Models for Structure Document Database Systems, In Markup Technologies, 1998

6.J. McHugh, J. Widom, Query Optimization for XML, VLDB 1999

7.Q. Li, B. Moon, Indexing and Querying XML Data for Regular Path Expressions, VLDB 2001.

8.Evangelos Kotsakis, Structured Information Retrieval in XML documents, ACM SAC 2002

9.S.W. Kim, et al, "Indexing and Retrieval of XML-encoded Structured Documents in Dynamic Environment", Lecture Notes in Computer Science (LNCS) Vol.2480, 2002

10.김성완 외 3인, XML 문서에서 순수 구조 질의에 대한 인덱싱 및 질의 처리, 제29회 한국정보과학회 추계학술발표논문집(I), 2002.

11.Igor Tatarinov et al., Updating XML, ACM SIGMOD 2001

12.W3C, http://www.w3c.org