한국산학기술학회논문지
Vol. 10, No. 2, pp. 396-406, 2009

XML 문서의 구조기반 검색성능 평가

김수희[1*]

Performance Evaluation on Structure-based Retrievals of XML Documents

Su-Hee Kim[1*]

요 약 이 논문에서는 XML 문서의 효율적인 구조검색을 위하여 기존의 연구에 이어 엘리먼트들의 순서를 명시하는 메타데이터들을 추가로 개발하였고, 이들을 바탕으로 구조기반 인덱싱 모델을 설계하였다. 설계한 구조검색 인덱스들은 문서의 계층구조에서 수직관계에 있는 엘리먼트들 뿐만 아니라 수평관계에 있는 엘리먼트들을 효율적으로 검색할 수 있게 한다. 제안한 구조기반 인덱스의 성능을 평가하기 위해 프로토타입 XML 문서 검색 시스템 개발하였고, XML 코퍼스를 대상으로 검색 실험을 수행하였다. 자손검색, 조상검색, 형제검색에서 ETID 모델보다 평균 검색 시간이 약 12% 정도 향상되었으며, 특정 엘리먼트 타입의 순서를 명시한 검색에서는 평균 검색 시간이 ETID 모델보다 25% 이상의 향상률을 보였다. 이것은 이 논문에서 제시한 Etype, Asso, LSSO를 이용한 검색이 엘리먼트의 순서를 명시한 검색 성능 향상에 큰 기여를 한 것으로 분석된다.

Abstract In extension to our previous study, we develop metadata that specify elements' structural orders, to increase the efficiency level of XML document's retrieval process. Then, we proposed a structure-based indexing model. We expect the model to generate a more efficient retrieval process of horizontally and vertically related elements. To evaluate the model's performance level, we developed an experimental prototype and conducted an experiment on an XML corpus. On average, descendant, ancestor and sibling retrievals were approximately twelve percent faster than the ETID model. And retrievals specifying structural orders of particular element types were approximately twenty-five percent faster than the ETID model. In conclusion, metadata, such as Etype, Asso and Lsso, may make a meaningful contribution to retrieval processes that specify elements' order.

Key Words : XML, Structure-based Retrieval, Indexing Model, Structure Information Representation

1. 서론

XML 문서를 대상으로 임의의 엘리먼트에 대해 빠르고 직접적인 검색을 위하여 Bottom Up Schema (BUS) 기법 등 많은 인덱스 기법들이 제시되어 왔다.

데이터베이스 시스템 관점에서 볼 때, XPATH, XQ , XML-QL 혹은 Quilt와 같은 대표적인 XML 질의 언어들은 XML 문서의 구조와 레이블들을 패턴 매칭과 전통적인 SQL 형태의 논리 조건과 결합하여 이용한다 [1,2,3,4]. Hypertext나 정보검색 분야(IR)에서는 문서 전체를 기본 단위로 취급하고, 문서의 내부 구조를 거의 무시하여 이질적인 데이터 처리에 대한 한계를 보여 주고 있다. XML 문서를 구조적인 측면에서 볼 때 계층적으로 수직 관계와 수평 관계에 있는 유형들에 대한 검색을 지원하는 것이 필요하다.

이 논문에서는 XML 문서의 효율적인 구조 기반 검색을 위하여, 기존의 연구[5,6]에 이어 XML 데이터의 엘리먼트들의 순서를 명시하는 메타데이터를 추가로 개발하고, 구조기반 인덱스 구축한다. 그리고, 그 성능을 평가하기 위해 XML 문서 검색 시스템의 프로토타입을 개발

이 연구의 일부는 한국과학재단 지역대학우수과학자 (과제번호: R05-2003-000-12405-0)지원으로 수행되었음
[1]호서대학교 컴퓨터공학부 [*]교신저자: 김수희(shkim@hoseo.edu)
접수일 09년 01월 18일 수정일 09년 02월 12일 게재확정일 09년 02월 18일

하고 XML 코퍼스를 대상으로 다양한 검색을 위한 실험을 수행하고 그 검색 성능을 측정한다. 부모, 자식, 형제 엘리먼트들에 대한 검색 성능뿐만 아니라, 조상검색, 자손검색에 대한 성능을 평가한다. 또한, 조상들, 자손들, 형제들 중에서 순서를 명시한 검색에 대한 실험을 수행한다.

2. 관련 연구

구조적인 문서로부터 정보를 검색하기 위해 여러 방법들이 제안되었다. Lee et al. [7]은 SGML 문서를 *k*-ary 완전 트리에 매핑하여 구조검색을 지원하는 방법을 개발하였다. *K*-ary 완전 트리 인덱스 모델에서는 부모 엘리먼트가 *k*개의 자식들을 가지며 문서 트리를 *k*-ary 완전 트리로 표현한 후에, 각 엘리먼트 노드에 식별자 UID가 할당된다. 이 구조에서는 각 엘리먼트 노드의 위치에 따라 UID가 정해지기 때문에 다음의 식을 이용하여 해당 노드의 부모나 자식 노드들을 계산할 수 있다.

$$Parent(UID) = \frac{UID-2}{k}+1$$

$$Jth\text{-}Child(UID) = k(UID\text{-}1) + j + 1$$

이 스킴에서는 노드의 깊이가 깊어질수록 사용하는 노드에 비해 데이터의 양이 커지는 (가상 노드가 존재하므로) 단점이 있다. 또한 *k*-ary 완전 트리의 특성상 계산을 통해 구한 자식이나 형제의 노드 번호가 실제 존재하는 노드인지 확인해야 하는 단점을 가지고 있다.

Lee의 연구를 바탕으로 구성된 Bottom Up Scheme (BUS)는 트리에서 엘리먼트 노드의 UID와 엘리먼트 타입 및 레벨을 포함하는 General element IDentifier (GID)를 생성한다[8]. 또한, BUS는 리프 노드에서만 인덱스를 함으로써 인덱스의 오버헤드를 많이 줄였다. 이 스킴에서는 *k* 값이나 노드의 변경 (삽입, 삭제)에 따라 UID의 재구성이 발생하는 단점을 가지고 있어 동적 환경에선 부적합하다.

문서의 갱신으로 인한 데이터베이스의 인덱스들을 더 효율적으로 갱신하기 위해, Jang et al.은 Oracle에서 인덱스 구성 테이블들을 사용함으로써 BUS 방법을 확장하였다[9]. 이 방법의 단점은 인덱스를 하고 질의를 처리하는 시간이 오래 걸린다는 점이다.

Oria et al.은 저장 장소의 오버헤드, 인덱스하는 시간, 질의하는 시간을 줄이기 위해 계층적인 딕셔너리를 사용하여 인덱스하고 질의하는 modified BUS method (MBM)을 제안하였다[10]. 갱신이 일어나는 경우 전체 문서를 다시 인덱스하는 필요성을 없도록 하기 위해 각 엘리먼트 GID에 부모 UID를 추가하였다. 각 엘리먼트가 인덱스됨에 따라, 순차적으로 UID가 할당된다. MBM에서는 한 노드의 자식 노드들이나 형제 노드들에 대한 UID를 사용하지 않으므로 이와 관련된 질의 처리는 매우 비효율적이다.

김성완 외 3명은 MBM과 동일한 UID 할당 방법을 채택하고 XML 문서의 DTD를 이용하여 문서의 경로 정보 테이블을 구성한다. 이 경로 정보 테이블은 각 엘리먼트 타입에 해당하는 절대 경로, 이 경로의 레벨과 고유하게 할당된 식별자 값으로 구성된다. XML 문서의 각 노드에 해당하는 포스팅을 (DID, UID, ETY, PUID, FCHD, LSIB, RSIB, NCHD) 등으로 구성한다 [11,12]. 포스팅의 구성자들은 각각 다음의 의미를 가지고 있다.

- DID : 문서 식별자
- UID : 해당 엘리먼트의 식별자
- ETY : 해당하는 절대 경로의 식별자
- PUID : 부모 엘리먼트의 UID
- FCHD : 첫 번째 자식 엘리먼트의 UID
- LSIB : 왼쪽 형제의 UID
- RSIB : 오른쪽 형제의 UID
- NCHD : 자식 엘리먼트들의 수

이 포스팅을 기반으로 ETY 기반의 인덱스, UID 기반의 인덱스를 구축한다. ETY 기반 인덱스를 이용하여 어떤 기준 경로의 부모, 첫째 자식, 왼쪽 형제, 오른쪽 형제는 쉽게 찾을 수 있다. 그러나, 조상 검색을 하기 위해서는 부모노드를 반복적으로 접근하여야 하며 자손 검색을 하기 위해서는 포스팅 내에서 FCHD를 추출하며 LSIB 혹은 RSIB에 대한 반복적인 접근으로 처리할 수 있다. 또한 형제검색을 하기 위해서는 LSIB 혹은 RSIB에 대한 반복적인 접근으로 처리할 수 있다. 그러므로, 조상, 자손과 형제에 대한 검색은 매우 비효율적이라 할 수 있다.

엘리먼트 단위 구문 트리를 이용한 방법(이하 IM-E 모델)[13]은 SGML 파서를 통해 나온 구문 트리 정보를 여러 개의 엘리먼트 단위 레코드로 저장하는 방식이다. 이 저장방식은 문서의 부분 삽입이 발생하면 기존의 인덱스의 정보 변경없이 효율적으로 인덱스 구조에 변경부분을 반영할 수 있다. 하지만 순서와 관련한 구조 검색을 위해서는 반복적인 노드 검색이 필요하여 매우 비효율적이다.

박종관 외 4명[6]은 문서의 구조정보를 다음과 같이 구성하였다:

EID : 각 엘리먼트를 구별하는 식별자

ETID : 문서상의 엘리먼트들 간의 계층정보, 즉 경로

정보를 EID들을 이용하여 표현

SORD : 문서상에서 한 노드의 위치를 레벨별로 형제 엘리먼트들간의 순서정보로 표기

SSORD : 문서상에서 레벨별로 동일한 타입의 엘리먼트들간의 순서 정보를 표기

그들은 EID를 2바이트를 사용하여 유일하게 표현하고, 이 EID들의 조합으로 ETID를 표현한다. 즉 부모의 ETID에 자신의 EID를 붙여서 자신의 ETID를 생성하는데 부모가 없는 루트 엘리먼트인 경우는 자신의 EID가 ETID가 된다. 이들의 표현법에 의하면 EID는 2바이트의 고정길이를 가진 유일한 식별자 역할을 하지만 동일한 엘리먼트가 서로 다른 경로에 여러 번 나타나게 되면 이에 대응하는 ETID는 여러 개가 된다. SORD와 SSORD 요소들도 역시 가변 길이의 문자열 형태이다. 표 2와 표 3은 표 1에 있는 DTD를 이용하여 각 엘리먼트 타입에 EID와 ETID를 할당한 예이다. 이러한 방법으로 ETID를 표현하면 저장공간이 많이 차지하고 활용하는 데 다소 불편함이 있다.

이 모델(이하 ETID 모델)에서는 문서의 구조정보를 <DID, ETID, SROD, SSORD, NCHD...> 요소들로 표현하였다. [6]에서는 ETID 모델을 이용한 검색 성능을 *k*-ary 기반 인덱스 모델과 엘리먼트 단위 구문트리에 기반한 IM-E 인덱스 모델을 이용한 검색 성능과 비교하는 실험을 수행하였다. ETID 모델을 이용한 부모검색, 자식검색, 형제검색의 성능이 *k*-ary 모델과 IM-E 모델을 이용한 검색성능보다 훨씬 우수함을 보였다.

[표 1] DTD의 예

```
<!ELEMENT paper (head, section+, reference+)>
<!ELEMENT head (title, author+, abstract)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (name, department)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT department (#PCDATA)>
<!ELEMENT abstract (#PCDATA)>
<!ELEMENT section (num, title, para, subsection+)
<!ELEMENT num (#PCDATA)>
<!ELEMENT subsection (num, title?, para)>
<!ELEMENT para (#PCDATA|img)*>
<!ELEMENT img EMPTY>
<!ELEMENT reference (#PCDATA)>
```

[표 2] EID 매핑 테이블

엘리먼트 타입	EID	엘리먼트 타입	EID
paper	00	section	07
head	01	num	08
title	02	subsection	09
author	03	para	0A
name	04	img	0B
department	05	reference	0C
abstract	06		

[표 3] ETID 매핑 테이블

엘리먼트 타입	ETID	엘리먼트 타입	ETID
paper	00	section	0007
head	0001	num	000708
title	000102, 000702, 00070902	subsection	000709
author	000103	para	00070A, 0007090A
name	00010304	img	NULL
department	00010305	reference	000C
abstract	000106		

3. 구조정보의 표현

XML 문서의 검색은 크게 세 종류로 분류할 수 있다: 구조기반 검색, 내용기반 검색, 구조와 내용이 혼재하는 혼합검색. 이 장에서는 XML 문서의 구조와 관련한 다양한 검색들을 효율적으로 수행하기 위해 XML 문서들과 그 DTD(Data Type Definition) 문서를 대상으로 사용할 요소들을 개발한다. 접근하는 방법은 지금까지 제안된 구조정보의 표현들 중 장점과 단점을 분석하여 좋은 점들을 반영하고, 다양한 검색 기능을 지원하기 위해 추가로 필요한 요소를 찾아내고 활용방법을 연구한다.

XML DTD를 읽고 작성할 수 있는 메타데이터로써 주어진 각 엘리먼트의 절대 경로, 이를 구별하기 위한 식별자, 그 레벨 등이 있다. 그리고 XML 문서를 대상으로 추출할 수 있는 메타데이터는 XML 문서를 계층적인 구조를 가진 트리로 볼 때 각 노드를 구별하기 위한 식별자, 각 노드의 구조정보, 형제 노드간의 순서 정보, 형제 노드들 중 동일한 엘리먼트 타입 내에서의 순서 정보 등이 있고 각 XML 문서를 구별하기 위하 식별자가 있다.

3.1 XML DTD로부터 구조정보 추출

■ Path

XML DTD를 읽고 작성할 수 있는 메타데이터로써 주어진 엘리먼트의 절대경로를 나타낸다.

■ Etype (Element Type)

Etype은 각 Path에 고유하게 부여되는 식별자이다. 절대 경로는 XML 문서 상의 엘리먼트의 계층 구조를 쉽게 알 수 있는 장점이 있으나, 그 길이가 가변이므로 효율적인 검색을 위해 각 절대 경로에 대해서 고유한 식별자를 부여한다.

■ Level

각 엘리먼트 타입의 레벨로 XML 문서 트리 상의 각 노드는 해당 Etype과 동일한 레벨이다. Level은 레벨 값을 필요로 하는 구조기반 검색에 사용할 수 있다. 표 4는 표 1에 있는 DTD를 대상으로 추출한 경로에 대한 식별자 Etype과 그 Level 정보를 작성한 것이다.

[6]에서는 표 1에 있는 DTD를 표 2와 표 3으로 표현한 반면, 이 논문에서는 Path와 Etype을 이용하여 표4로 표현하였다.

3.2 XML 문서로부터 구조정보 추출

■ DID(Document IDentifier)

여러 XML 문서들이 어떤 형태(파일)로 저장되어 있을 때, 이들을 구별하기 위한 식별자이다.

■ UID(Unique element IDentifier)

XML 문서를 트리 형태로 모델링할때 각 노드를 구별하기 위한 식별자이다. 초기에는 XML 문서를 읽으면서 각 노드의 UID를 1부터 순차적으로 할당한다. 추후에 새로운 노드를 추가하는 경우에는 그 다음 가능한 번호를 할당한다.

[표 4] 경로 정보의 표현

Etype	Path	Level
1	/paper	1
2	/paper/head	2
3	/paper/head/title	3
4	/paper/head/author	3
5	/paper/head/author/name	4
6	/paper/head/author/department	4
7	/paper/head/abstract	3
8	/paper/section	2
9	/paper/section/title	3
10	/paper/section/subsection/title	4
11	/paper/section/num	3
12	/paper/section/subsection	3
13	/paper/section/para	3
14	/paper/section/subsection/para	4
15	/paper/reference	2

■ SORD(Sibling ORDer)

[6]에서 제안한대로 주어진 노드의 형제 엘리먼트간의 순서 정보와 그 상위 노드들의 순서 정보를 나타낸다. 이 SORD를 이용하여 각 노드를 식별할 수 있으며 XML 문서 내에서의 계층 정보와 순서 정보를 쉽게 구할 수 있다. SORD는 가변길이로 표현된다. 각 레벨에 대해 하나의 숫자로 그 형제들간의 순서정보를 표현하는데, SORD /23은 해당 엘리먼트가 루트노드의 둘째 자식 중 세 번째 자식이라는 것을 의미한다.

■ ASSO(Ancestor Same Sibling Order)

기준 노드를 중심으로 그 상위 노드들의 동일한 엘리먼트 타입들 간의 순서 정보를 나타낸다. ASSO는 *j* 번째 <B>의 부모이면서 *i* 번째 <A>의 검색 등에 사용될 수 있다.

■ LSSO(Last SSO)

동일한 엘리먼트 타입들 간의 순서 정보를 나타낸다. LSSO는 특정 엘리먼트의 몇 번째 형제를 검색하기 위해 효율적으로 사용될 수 있다.

■ NCHD(Number of CHiID): 자식들의 수

이 요소는 자식을 검색하기 위해 SORD를 생성할 때에 사용될 수 있다.

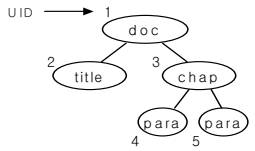
■ NRSIB(Number of Right SIBling)

오른쪽 형제들의 수를 나타낸다. 형제들을 검색하기 위해 SORD와 함께 사용될 수 있다.

■ SPOS(Start POSition)와 LENG

XML 문서 내에서 해당 노드의 시작 위치와 그 내용의 길이를 각각 나타낸다.

그림 1은 XML 문서트리의 예가 되겠으며 표 5는 그림 1의 트리구조를 이용하여 UID, SORD, ASSO, LSSO를 작성한 예이다. 구조정보를 표현하기 위해 이 연구에서는 [6]에서 제안한 SORD는 그대로 사용하고 EID와 ETID를 다른 방법, 즉, Etype과 Path를 이용하여 표현한다. 그리고 동일한 타입의 엘리먼트들 중 순서 정보를 요구하는 질의를 효율적으로 처리하기 위해 Asso와 Lsso를 제안한다. 또한 형제들을 효율적으로 검색하기 위해 RSIB를 제안한다. 제안하는 구조정보 메타데이터는 표 6과 같다.



[그림 1] XML 문서트리의 예

[표 5] 메타데이터 작성의 예

UID	SORD	ASSO	LSSO
1	/		/
2	/1	/	1
3	/2	/	1
4	/21	/1	1
5	/22	/1	2

[표 6] 구조정보 메타데이터

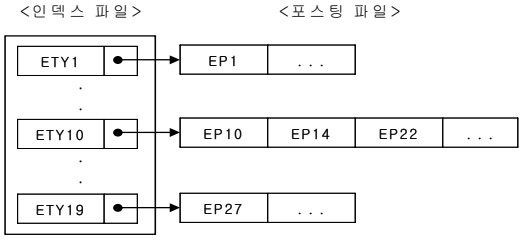
Etype	UID	DID	SORD	ASSO	LSSO	NRSIB	SPOS	LENG

3.3 구조기반 인덱스

구조기반 검색은 특정 엘리먼트 이름 및 타입으로 주로 표현되며, 경우에 따라 순서 정보를 포함하기도 한다. 또한, 어떤 결과를 추출하기 위해서 특정 엘리먼트의 식별자를 이용한 검색이 필요할 때가 있다. 따라서 엘리먼트 단위로 추출한 구조 정보를 이용하여 용도별로 다양한 인덱스들을 구축하는 것이 필요하다. XML DTD를 이용하여 <Etype, Path, Level>로 이루어지는 테이블을 만들 수 있으며, XML 문서로부터 추출할 수 있는 구조 정보의 포스팅은 <DID, UID, Etype, SORD, ASSO, LSSO, NCHD, NRSIB, SPOS, LENG>이다. 이 포스팅으로 구조기반 인덱스를 구축할 수 있다.

■ Etype 인덱스

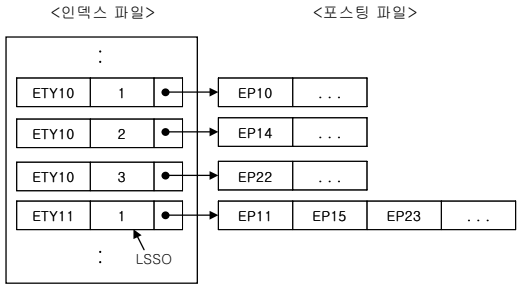
엘리먼트 이름이나 타입으로 표현되는 검색을 위한 인덱스이다. 어떤 엘리먼트 타입을 기준으로 수직 관계에 있는 엘리먼트들을 효율적으로 검색할 수 있다. 특히, 특정 엘리먼트 타입의 자식/후손 검색은 해당 Etype에 연결된 포스팅에 접근하여 별도의 연산 작업 없이 결과 값을 추출할 수 있다. 특정 엘리먼트 타입의 부모/조상 검색은 간단한 연산으로 그 결과를 추출할 수 있다([그림 2] 참조).



[그림 2] Etype 인덱스 구조

■ (Etype, LSSO) 인덱스

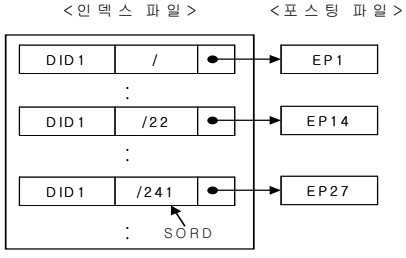
엘리먼트 타입과 순서 정보를 포함하는 검색을 위한 인덱스이다. 특정 순서에 있는 엘리먼트 타입에 관련된 검색을 효율적으로 처리할 수 있다([그림 3 참조]).



[그림 3] (Etype, LSSO) 인덱스 구조

■ (DID, SORD) 인덱스

특정 문서의 엘리먼트의 구조 정보에 접근하거나 구조기반 검색에 사용될 수 있다. 또한, 이 인덱스는 구조기반 검색 후 얻어지는 {DID, SORD} 집합에 해당하는 내용을 출력하기 위해 사용될 수 있다. {DID, SORD}는 구조정보테이블의 키 역할을 하므로 각 인덱스 엔트리에 연결되는 포스팅은 하나씩 존재한다([그림4] 참조).



[그림 4] (DID, SORD) 인덱스

3.4 구조기반 인덱스의 활용

1) Etype 이용

표 7은 Etype 인덱스를 이용할 수 있는 검색 유형 및 추출되는 Etype 집합을 요약한다.

■ 자식 검색

추출된 각 Etype이 존재하는 포스팅들에 접근하여 {DID, SORD} 집합을 결과로 반환한다. k 번째 자식을 검색하는 경우에는 추출된 Etype에 연결된 포스팅들에 접근하여 SORD에 k 번째에 해당하는 문자 값을 덧붙여서 추출된 {DID, SORD} 집합을 결과로 반환한다. 예를 들어, [/22]의 첫 번째 자식은 [/221]이다. {DID, UID} 집

합으로 결과를 반환할 수 있으나{DID, SORD} 집합을 반환하면 반환된 SORD로 추가적인 처리를 필요한 대로 할 수 있다. 최종적으로 해당 구조에 저장되어 있는 내용을 검색하는 것이 목적인 경우는 {DID, UID} 집합을 반환하는 것이 더 효율적이다.

■ 후손 검색

추출된 각 Etype에 연결된 포스팅들에 접근하여 {DID, SORD} 집합을 결과로 반환한다.

■ 부모 검색

추출된 각 Eype에 연결된 포스팅들에 접근하여 SORD의 마지막 한문자를 절단해서 추출된 {DID, SORD} 집합을 결과로 반환한다. 예를 들어 [/221]의 부모는 [/22]이다.

■ 조상 검색

추출된 각 Etype에 연결된 포스팅들에 접근하여 SORD의 끝에서 필요한 만큼 절단해서 추출된 {DID, SORD} 집합을 결과로 반환한다. 조상 검색은 검색 유형에 따라 처리하는 방식이 다르다. 각 포스팅에 대해서 해당 DID와 함께, 모든 조상을 검색하는 경우에는 끝에서부터 한문자씩 반복적으로 절단하여 한 문자만 남을 때까지의 모든 SORD들을 반환하고, <A>의 조상 중 <B>만 검색하는 경우에는 끝에서 <A>와 <B>의 레벨 차이 만큼의 문자를 절단한 SORD를 반환하며, *k* 번째 조상을 검색하는 경우에는 끝에서 *k*개의 문자를 절단한 SORD를 반환한다.

■ 형제 검색

추출된 각 Etype에 연결된 포스팅들에 접근하여 SORD의 마지막 문자를 필요한 만큼 -, +해서 추출된 {DID, SORD} 집합을 결과로 반환한다. 모든 형제는 앞에 있는 형제와 뒤에 있는 형제를 이용하여 추출한다. *k* 번째 앞/뒤에 있는 형제 검색은 마지막 문자에 '-*k*'/'+*k*' 한 SORD를 이용한다.

2) (Etype, LSSO) 이용

(Etype, LSSO)로 구성된 복합 인덱스를 이용하기 위해서는 우선 검색 유형별로 필요한 Etype 집합과 LSSO를 추출하여야 한다. 표 8은 Etype과 LSSO로 구축된 인덱스를 이용할 수 있는 검색 유형 및 추출하는 Etype 집합과 LSSO를 요약한다. "i 번째 <A>의 부모 혹은 조상 중 j 번째 <B>" 검색일 경우에는 해당 {Etype, LSSO}에 연결된 포스팅들의 ASSO 값을 확인해야 한다. 부모는 마지막 문자가 [J]인 ASSO, 조상은 <B>의 레벨위치의 문자가 [J]인 ASSO를 가지는 {DID, SORD} 집합을 선별한다. 그 다음으로 각 검색에 대응하는 {DID, SORD} 집합을 결과 값으로 반환한다. 예를 들어, 부모 검색은 선별한 SORD의 마지막 한문자를 절단하여 결과를 구한다.

[표 7] Etype을 이용하는 검색 유형

검색 유형		추출할 Etype
자식	<A>의 자식	[/A]를 포함하면서 [/A]이후로 [/]가 1개인 경로
	<A>의 자식중 <B>	[/A/B]로 끝나는 경로
	<A>의 k번째 자식	[/A]로 끝나는 경로
후손	<A>의 후손	[/A/]를 포함하는 경로
	<A>의 후손중 <B>	[/A]를 포함하면서 [/B]로 끝나는 경로
	<A>의 k번째 후손	[/A]를 포함하면서 [/A]이후로 [/]가 k개인 경로
부모	<A>의 부모	[/A]로 끝나는 경로
	<A>의 부모중 <B>	[/B/A]로 끝나는 경로
조상	<A>의 조상	[/A]로 끝나는 경로
	<A>의 조상중 <B>	[/B]를 포함하면서 [/A]로 끝나는 경로
	<A>의 k번째 조상	[/A]로 끝나는 경로
형제	<A>의 형제	[/A]로 끝나는 경로
	<A>의 앞/뒤에 있는 형제	
	<A>의 k번째 앞/뒤에 있는 형제	

[표 8] Etype과 LSSO를 이용하는 검색 유형

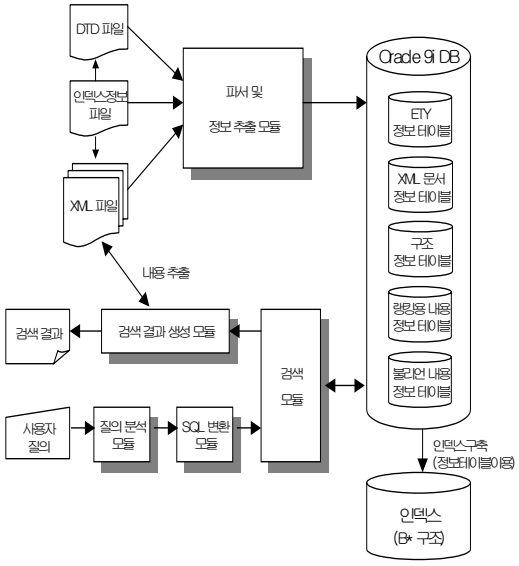
검색 유형		Etype 형태	LSSO
자식	<A>의 자식 중 j 번째 <B> 검색	[../A/B]	[J]
	i 번째 <A>의 자식중 j 번째 <B>검색		
	i 번째 <A>의 k 번째 자식 검색	[../A]	[I]
후손	<A>의 후손 중 j 번째 <B> 검색	[../A../B]	[J]
	i 번째 <A>의 후손중 j 번째 <B>검색		
부모	i 번째 <A>의 부모 검색	[../A]	[I]
	i 번째 <A>의 부모중 j 번째 <B>검색	[../B/A]	
조상	i 번째 <A>의 조상중 j 번째 <B>검색	[../B../A]	[I]
	i 번째 <A>의 k 번째 조상 검색	[../A]	
형제	i 번째 <A>의 형제 검색	[../A]	[I]
	i 번째 <A>의 앞/뒤의 형제 검색		
	i 번째 <A>의 k번째 앞/뒤의 형제검색		

4. 프로토타입 XML 문서 검색 시스템

XML 문서를 검색하기 위한 프로토타입 XML 문서 검색 시스템을 개발하였다. 그림 5는 프로토타입 XML

문서 검색 시스템의 아키텍처로 구조기반 검색, 내용기반 검색, 혼합 검색을 하기 위한 전체적인 개관을 나타내고 있다.

이 논문에서는 개발한 시스템을 이용한 구조기반 검색에 대하여 논의한다. 구조기반 검색을 위해서는 XML 문서의 DTD로부터 얻을 수 있는 경로 정보 (Etype, Path, Level), 검색대상이 되는 XML 문서에서 얻을 수 있는 (DID, SORD, ASSO, LSSO, NCHD, NRSIB, SPOS, LENG), 실제 XML 문서의 위치하는 디렉토리 및 파일 이름, DID 등이 필요하다. 이 데이터들을 테이블 형태로 오라클 9i에 저장한다. XML 문서의 구조에 기반한 다양한 검색을 수행하기 위해 3장에서 논의한 다양한 검색 유형을 표현하는 포맷을 개발하고 구현하였다.



[그림 5] 프로토타입 XML 문서 검색 시스템 아키텍처

4.1 질의 표현

구조기반 질의에서는 특정 엘리먼트의 검색, 특정 엘리먼트의 자손, 조상, 형제의 검색으로 분류할 수 있다. 표 9와 표 10은 구조기반 검색의 유형 및 표현 방법을 나타낸다. 표 9에 있는 질의는 엘리먼트 타입 Etype을 이용하여 처리할 수 있으며, 표 10에 있는 질의는 (Etype, LSSO)를 이용하여 처리할 수 있다. 계층구조의 관계표현을 위해 다음의 기호를 사용한다.

- E(element) : 기준이 되는 엘리먼트
- C(child) : 자식
- D(descendant) : 후손
- P(parent): 부모
- A(ancestor) : 조상
- S(sibling) : 형제

[표 9] 구조기반 검색 유형 및 질의표현 1

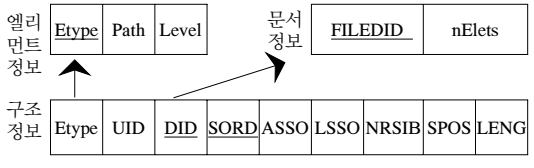
검색 유형		표 현
자식	<E>의 자식	E/C::*
	<E>의 자식 중 <B>	E/C::B
	<E>의 *k* 번째 자식	E/C::*[*k*]
후손	<E>의 후손	E/D::*
	<E>의 후손 중 <B>	E/D::B
	<E>의 *k* 번째 후손	E/D::*[^*k*]
부모	<E>의 부모	E/P::*
	<E>의 부모 중 <B>	E/P::B
조상	<E>의 조상	E/A::*
	<E>의 조상 중 <B>	E/A::B
	<E>의 *k* 번째 조상	E/A::*[^*k*]
형제	<E>의 형제	E/S::*
	<E>의 앞뒤에 있는 형제	E/S::*[-|+]
	<E>의 *k* 번째 앞뒤에 있는 형제	E/S::*[-*k*|+*K*]

[표 10] 구조기반 검색 유형 및 질의표현 2

검색 유형		표 현
자식	<E>의 자식 중 *j* 번째 <B>	E/C::B[*j*]
	i 번째 <E>의 자식 중 *j* 번째 <B>	E[*i*]/C::B[*j*]
	i 번째 <E>의 *k* 번째 자식	E[*i*]/C::*[*k*]
후손	<E>의 후손 중 *j* 번째 <B>	E/D::B[*j*]
	i 번째 <E>의 후손 중 *j* 번째 <B>	E/D::B[*j*]
부모	*i* 번째 <E>의 부모	E[*i*]/P::*
	i 번째 <E>의 부모 중 *j* 번째 <B>	E[*i*]/P::B[*j*]
조상	*i* 번째 <E>의 조상 중 *j* 번째 <B>	E[*i*]/A::B[*j*]
	i 번째 <E>의 *k* 번째 조상	E[*i*]/A::*[^*k*]
형제	*i* 번째 <E>의 형제	E[*i*]/S::*
	i 번째 <E>의 앞뒤에 있는 형제	E[*i*]/S::*[-|+]
	*i*번째 <E>의 *k*번째 앞뒤있는 형제	E[*i*]/S::*[-*k*|+*K*]

4.2 데이터베이스 스키마

추출한 구조정보 요소들을 Oracle 9i 데이터베이스에 저장하기 위해 ER 다이어그램을 작성하고 데이터베이스 테이블들의 스키마를 생성한다. XML DTD를 이용하여 추출한 Etype 정보 테이블, XML 문서들을 이용하여 생성하는 포스팅 파일인 구조정보 테이블, XML 문서 정보 테이블을이 모두 Oracle 9i의 데이터베이스에 저장하고, 설계한 인덱스들을 생성한다.



엘리먼트 정보 테이블에서는 Etype이 기본키가 되며 구조정보 테이블에서는 (DID,SORD)가 기본키이다. 구조정보 테이블의 Etype과 DID는 엘리먼트 정보 테이블의 Etype과 문서정보 테이블의 DID를 각각 참조하는 외래키이다.

4.3 질의 처리기

XML 문서의 구조를 대상으로 하는 질의를 처리하기 위해 필요한 모듈들을 개발하였다. 3절에서 개발한 메타데이터들을 오라클 9i에 저장하고 4.1절에서 논의한 다양한 형태의 질의를 처리하는 모듈들을 Proc C로 구현하였다.

1) 질의 분석 모듈 및 SQL 변환 모듈

앞서 서술한 질의의 표현에 대해서 이를 분해하고 분석하여 어떤 유형인지를 파악하고 SQL로 변환하는 모듈이다.

2) 검색 모듈

사용자의 질의를 분석하여 SQL 질의문으로 변환한 후, 이를 이용하여 검색하기 위한 검색 모듈이다. 사용자의 질의는 그 성격에 따라 여러 개의 SQL문이 생성될 수 있다.

3) 검색 결과 생성 모듈

검색 모듈을 통해 나온 결과를 사용자들이 쉽게 파악할 수 있는 포맷으로 제공하는 모듈이다. 기본적으로 엘리먼트 단위로 검색을 하기 때문에 동일한 엘리먼트들이 중복되어 질의의 결과로 나올 수 있다. 한 엘리먼트에 포함되는 다수의 엘리먼트들이 결과로 역시 리턴될 수 있다.이러한 점들을 처리하여 사용자들에게 결과를 쉽게 접근하고 파악할 수 있는 모듈을 개발한다.

5. 실험 및 결과

실험에 사용한 XML 문서 코퍼스는 자연과학 분야, 생명과학 분야, 공학 분야의 과제 제안서 형식으로 설계된 DTD에 의거하여 작성된 한글 문서들이다.

[표 11] XML 문서 코퍼스

분 야	형태	문서 수
자연과학, 생명과학, 공학	보고서	800

이 XML 문서 코퍼스들은 서버의 특정 디렉토리에 저장하였으며, 이들로부터 추출한 메타데이터들을 Oracle 9i 데이터베이스에 저장하였다. 각 질의에 대한 처리는 Proc C로 구현하였다. 그리고 본 연구에서 제안한 방법의 검색 성능을 [6]에서 제안한 ETID 모델과 비교한다.

[표 12] 서버 현황

항 목	사 양	항 목	사 양
서버 모델	Sun Ultra-60	NET	100 Base
CPU	Ultra SPARC-Ⅱ 360MHz X2	OS	Solaris 8(Sun OS 5.8)
메모리	1 GB	DB	Oracle 9i
디스크	27 GB		

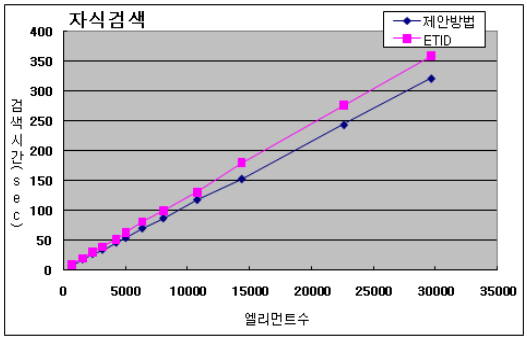
구조검색에서는 질의를 처리하여 원하는 결과를 사용자에게 제공하는 데 걸리는 시간이 성능평가의 주된 이슈가 된다. 3절에서 개발한 메타데이터는 [6]에서 제안한 EID와 ETID를 Etype과 Path를 이용하여 표현하였으며 동일한 타입의 엘리먼트들 중 순서 정보를 요구하는 질의를 처리하기 위해 Asso와 Lsso를 제안하였다. 또한 형제들을 효율적으로 검색하기 위해 NRSIB를 제안하였다. 즉, 이 논문에서 제안한 인덱싱 모델은 문서의 구조정보를 <DID, UID, Etype, SORD, ASSO, LSSO, NCHD...>로 표현하고 ETID 모델은 <DID, ETID, SORD, SSORD, NCHD...>로 표현한다.

(그림 6)은 자식검색의 성능을 나타낸다. 자식검색에서 해당 엘리먼트의 자식 엘리먼트들의 Etype을 엘리먼트 정보 테이블에서 찾아낸후, 이를 구조정보 테이블에서 검색하여 해당되는 행의 (DID, SORD)를 결과로 리턴한다. ETID 기반에서는 해당 엘리먼트를 EID 테이블에서 찾아 그것의 식별자를 알아내어 ETID 테이블에서 그것의 자식 엘리먼트들의 문자열을 파악한다. 이 문자들을 구조정보 테이블에서 검색하여 해당하는 (DID, SORD)를 결과로 리턴한다.

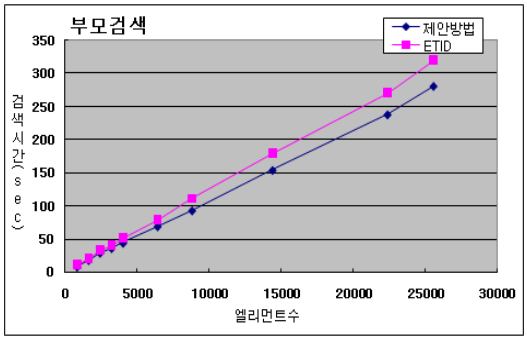
(그림 7), (그림 8), (그림 9), (그림 10)은 각각 부모검색, 형제검색, 조상검색, 후손검색의 성능을 나타낸다. 자식검색, 부모검색, 형제검색, 조상검색, 후손검색에서 제안한 방법은 ETID 방법보다 각각 평균 11.73%, 13.16%, 9.12%, 11.30%, 14.98% 향상되었다.

특정 타입의 엘리먼트 순서를 명시한 검색에서는 ETID 방법보다 25% 이상의 향상률을 보였다. (그림 11), (그림 12), (그림 13), (그림 14)는 각각 자식 중 특정 타입의 순서기반 검색, 후손 중 특정 타입의 순서기반 검색, 특정타입의 순서기반 형제검색, 순서기반 특정타입의 부모곰색

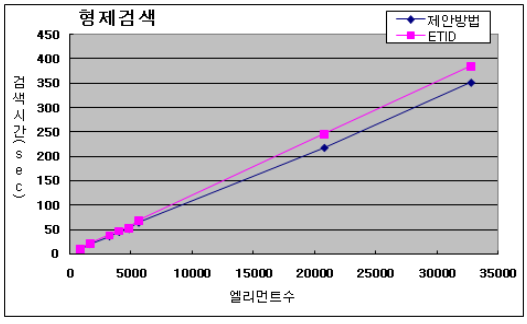
성능을 나타낸다. 이들의 검색 성능은 ETID 방법보다 각각 평균 27.73%, 28.24%, 31.32%, 25.86% 향상되었다. 이러한 성능향상은 이 논문에서 제시한 문서구조 정보 요소 (Etype, LSSO)를 이용한 인덱스를 이용하여 특정 엘리먼트가 문서에서 나타나는 순서를 명시한 검색 속도에 큰 기여를 한 것으로 판단된다.



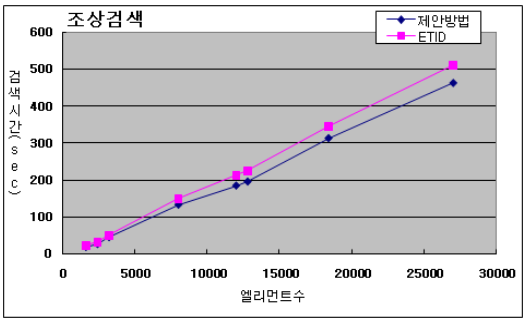
[그림 6] 자식검색의 성능



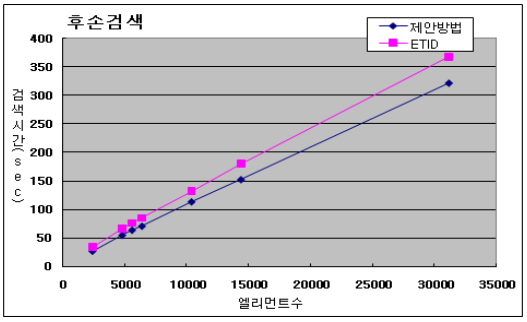
[그림 7] 부모검색의 성능



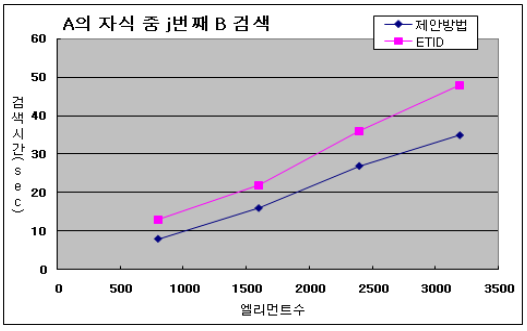
[그림 8] 형제검색의 성능



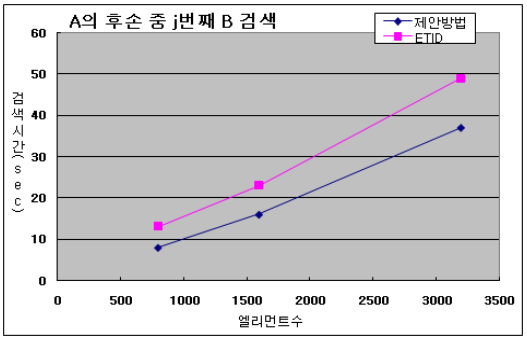
[그림 9] 조상검색의 성능



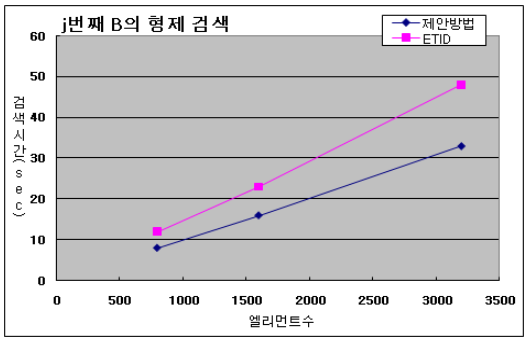
[그림 10] 후손검색의 성능



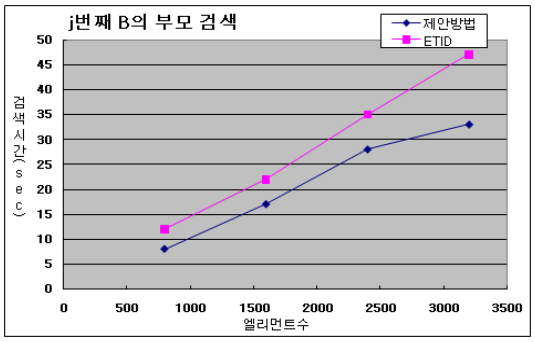
[그림 11] 자식 중 특정타입의 순서기반 성능



[그림 12] 후손 중 특정타입의 순서기반 성능



[그림 13] 특정타입의 순서기반 형제검색 성능



[그림 14] 특정타입의 순서기반 부모검색 성능

[표 13] 검색 성능및 향상도

	검색형태	ETID 모델	제안하는 모델	향상도 (%)
평균검색시간(sec)	자식	111.5	98.4	11.73
	부모	111.7	97.0	13.16
	형제	108.3	98.4	9.12
	조상	193.5	171.6	11.30
	후손	134.4	114.3	14.98
	특정타입의 순서기반 자식	29.8	21.5	27.73
	특정타입의 순서기반 후손	28.3	20.3	28.24
	순서기반 특정타입의 형제	27.7	19.0	31.32
	순서기반 특정타입의 부모	29.0	21.5	25.86

6. 결 론

XML 문서를 구조적인 측면에서 볼 때 계층적으로 수직 관계와 수평 관계에 있는 유형들에 대한 검색을 지원하는 것이 필요하다. 또한, 내용적인 측면에서 볼 때 데이터 중심의 검색과 문서 중심의 검색을 통합하여 지원하는 것이 필요하다. 이를 위해서는 문서와 질의에 가중치를 부여하고 질의의 결과에 랭킹을 부여하는 기법이 필요하다.

이 논문에서는 XML 문서의 효율적인 구조검색을 위하여 필요한 메타데이터들을 기존의 연구에 이어 추가로 개발하였다. 특히, 특정 엘리먼트가 문서의 트리구조에서 나타나는 순서를 명시하는 ASSO와 LSSO를 제안하였다. 이들을 이용하여 구조검색을 위한 인덱스들을 설계하였다. 제안한 구조기반 인덱스 모델은 주어진 계층구조에서 수직관계에 있는 부모, 자녀, 조상, 자손들뿐만 아니라 수평관계에 있는 형제 엘리먼트들을 효율적으로 검색할 수 있게 한다.

그리고, 그 성능을 평가하기 위해 XML 문서 검색 시스템의 프로토타입을 개발하였고, 이를 이용하여 이공계 과제제안서 형태의 XML 코퍼스를 대상으로 다양한 검색을 위한 실험을 수행하였으며, 그 검색 성능을 ETID 모델의 검색 성능과 비교하였다. 이미 언급한대로 ETID에 기반한 인덱스 모델의 검색 성능은 *k*-ary 기반 인덱스 모델과 엘리먼트 단위 구문 트리에 기반한 IM-E 인덱스 모델의 검색 성능보다 훨씬 우수함을 보인 모델이다.

자식검색, 부모검색, 형제검색, 조상검색, 후손검색에서 제안한 모델은 ETID 모델보다 평균 검색 시간이 각각 11.73%, 13.16%, 9.12%, 11.30%, 14.98% 향상되었으며, 특정 엘리먼트 타입의 순서 정보를 포함하는 검색에서는 평균 검색 시간이 ETID 모델보다 25% 이상의 향상률을 보였다. 이것은 이 논문에서 제시한 문서구조 정보 요소 Etype, Asso, LSSO를 이용한 검색이 특정 엘리먼트가 문서에서 나타나는 순서를 명시한 검색을 위한 성능 향상에 큰 기여를 한 것으로 판단된다.

참고문헌

[1] S. Aviteboul, S. Buneman, and D. Suciu, “Data on the Web-From Relations to Semistructured Data,” Morgan Kaufman Publishers, San Francisco, 2000.

[2] A. Deutch D. Fernandez, M. and Florescu, and D. Suciu, “A query language for XML,” WWW8/ Computer Networks, 31(11-16): 1155-1169, 1999.

[3] D. Kossmann, “Special Issue on XML,” IEEE Data Engineering Bulletin, 22(3), 1999.

[4] D. Chamberlin, J. Robie, and D. Florescu, Quilt: “An XML query language for heterogeneous data sources,”

In D. Suciu and G. Vossen, editors, 3rd International Workshop on the Web and Databases, Dallas, Texas 2000.

[5] 김수희, 한예지, “XML 문서의 효율적인 구조검색을 위한 인덱싱,” 정보과학회 데이터베이스연구회, 2003

[6] 박종관, 손충범, 강형일, 유재수, 이병엽, “XML 문서의 효율적인 구조기반 검색을 위한 색인모델,” 정보처리학회논문지 D 제8-D권 제5호, pp. 451-460, 2001.

[7] Yong Kyu Lee, Seong-Joon Yoo, Kyoungro Yoon and P. Bruce Berra, "Index Structure for Structured Documents," In DL 96, Bethesda, 1996.

[8] Dongwook Shin, Hyuncheol Jang and Honglan Jin, "Bus: An Effective Indexing and Retrieval Scheme in Structured Documents," In DL 98, Pittsburgh, 1998.

[9] Hyunchul Jang, Youngil Kim, and Dongwook Shin, "An Effective Mechanism for Index Update in Structured Documents," In CIKM 99, Kansas City, 1999.

[10] Vincent Oria, Amit Shah, Samuel Sowell, “Indexing XML Documents: Improving the BUS Method,” 7th Workshop on Multimedia Information Systems, 7-9 November 2001, Villa Orlandi, Capri, Italy, Proceedings, pp. 51-60, 2001.

[11] Sung Wan Kim, Jaeho Lee, Hae Chull Lim, “Indexing and Retrieval of XML-Encoded Structured Documents in Dynamic Environment,” Springer -Verlag Berlin Heidelberg 2002, pp. 141-154, 2002.

[12] 김성완, 정현석, 이재호, 임해철, “XML 문서에서 엘리먼트 타입을 이용한 구조적 검색 기법의 설계,” 2003년도 한국정보과학회 봄학술발표논문집 Vol.30, No.1, pp. 584-586, 2003.

[13] 한성근외 4명, “동적 환경에 적합한 SGML 인덱스 관리자의 설계 및 구현”, 한국정보처리논문지, 제6권 제 10호, 1999. (IM-E)

[14] Norbert Gövert, Norbert Fuhr, Mohammad Abolhassani, and Kai Großjohann. “Contentoriented XML retrieval with HyREX,” In Proceedings of the 1st INEX Workshop, Sophia Antipolis, France, pp. 26-32, 2003.

김 수 희(Su-Hee Kim) **[정회원]**

- 1979년 2월 : 부산대학교 졸업 (학사)
- 1986년 12월 : University of Georgia (전산학 석사)
- 1988년 8월 : University of Georgia (수학 석사)
- 1993년 8월 : University of South Carolina (전산학 박사)
- 1993년 8월 ~ 1994년 2월 : Benedict College 조교수
- 1994년 3월 ~ 현재 : 호서대학교 컴퓨터공학부 교수

<관심분야>
XML 정보검색, 데이터베이스, 유비쿼터스 데이터 처리