한국산학기술학회논문지
Vol. 6, No. 1, pp. 1-11, 2005

XML 정보검색의 효율적 전처리를 위한 문서여과 알고리즘

김명숙* · 공용해

Document Filtering Algorithm for Efficient Preprocessing of XML Information Retrieval

Myung-Sook Kim* and Yong-Hae Kong

요 약 본 논문은 다수의 XML 문서들을 대상으로 하는 XML 정보검색에서, XML의 효율적 질의검색을 위한 전처리 방법을 제안한다. 기존의 전처리 방법은 질의의 키워드에 대하여 XML 문서를 파싱하거나, 질의와 XML 문서로부터 생성된 시그너처 정보를 비교하여 XML 문서를 여과한다. 그러나 이러한 방법은 질의에 종속적이며 다량의 XML 문서들이 존재할 경우 매우 비효율적이다. 이를 위하여, 본 연구는 온톨로지를 사용하여 서로 다른 구조와 속성을 갖지만 동일 영역의 정보를 포함하고 있는 XML 문서에 적용 가능한 포괄적 DTD를 생성하고, 이를 이용하여 검색 영역에 포함되지 않는 불필요한 XML 문서를 여과한다. 예제 XML 문서를 적용하여 제안한 문서여과 알고리즘의 성능을 테스트한다.

Abstract The paper proposes a preprocessing method for efficient processing of XML queries in information retrieval with a large amount of XML documents. The conventional preprocessing methods filter out XML documents by parsing XML document for keyword of query or by comparing query signatures with signatures of XML document to be generated. But these methods are dependent on a query and are very inefficient for a large amount of XML documents. For this, we generate a universal DTD based on ontology of a domain. The universal DTD is applicable to the XML documents when they contain information of a same domain even when they have different structures and attributes. Then, using the universal DTD, we filter out the XML documents that are not bounded in the domain. We evaluate the performance of this method through experiments.

Key Words : XML, Ontology, DTD, XML-Filtering

1. 서 론

XML은 최근 인터넷상에서의 자료교환을 위한 표준으로 자리잡고 있으며, 웹상에서 동작하는 다양한 응용들에서 사용되는 반 구조적 데이터를 표현하는 방법이다. XML은 사용자들이 정의하는 태그를 사용하여 표현되며, 문서에 따라 다양한 구조를 가지고 있다[1]. XML에 대한 연구는 대부분 XML 문서에 대한 효율적인 질의처리 방법에 대한 연구이며, 또한 XML 정보검색을 위해 다양한 질의어들이 연구되었다[2~6]. 그러나 다양한 질의처리 방법은 대규모 XML 문서에 대하여 수행되며 이러한 질의처리의 효율은 문서의 양에 종속적으로 결정된다. 기존에 연구된 XML 문서여과 방법들은 각 XML 문서를 트리구조로 파싱하여 인가된 질의를 분석한다. 즉, 질의에 포함된 키워드나 속성값들이 문서의 구조에 정확하게 포함되어 있는지를 확인하기 위해 XML 문서를 구조적으로 파싱하거나, 각 XML 문서와 질의에서 시그너처 정보를 생성하여 문서와 질의 사이의 유사도를 계산함으로써 문서의 적합성 여부를 판별한다[7, 8]. 이러한 방법은 인가되는 각 질의에 대하여 수행되어야 하므로, 질의와 XML 문서의 양이 많아지면 매우 비효율적이다. 따라서 웹상의 수많은 XML 문서를 대상으로 정보검색을 수행하기 위해서는, 관심영역의 XML 문서들을 사전에 선별할 수 있는 보다 개선된 방법이 필요하다.

본 논문은 검색에 불필요한 문서를 사전에 제거함으로써 실제로 질의검색을 시행할 문서의 개수를 줄임으로써 검색의 효율을 증대시킬 수 있는 온톨로지 기반의 XML 문서여과 알고리즘을 제안하였다. 정보검색 전의 전처리를 위한 문서여과 과정은 시간적 소모를 더하는

이 논문은 2002학년도 순천향대학교 산업기술연구소 학술연구조성비 일반연구과제의 지원에 의하여 연구되었음.
순천향대학교 정보기술공학부
*교신저자: 김명숙(krhkms@sch.ac.kr)

과정으로 생각될 수 있겠지만, 검색 영역에 포함되는 문서만을 선별할 수 있는 바탕이 제공된다면 매우 간단한 처리를 통하여 검색의 효율을 증대시킬 수 있다. 제안한 문서여과 알고리즘은 적합한 문서를 선별하는 기준을 위해 온톨로지를 사용하였다. 온톨로지는 정보 교환용으로 합의된 어휘나 구조를 만들기 위해 사용되는 개념들의 집합을 의미하며, 특히, W3C 등에서 응용 영역에 대한 온톨로지 표준화 작업을 진행하고 있다 [9, 10, 11]. 또한 온톨로지는 동일 영역의 구조가 다양한 XML 문서들의 구조를 모두 포함할 수 있으므로, 온톨로지로부터 정보검색 대상인 XML 문서를 선별하기 위한 기반을 제공할 수 있다.

우선, 문서여과 알고리즘은 온톨로지로부터 동일 영역의 XML 문서 구조를 모두 포함할 수 있는 포괄적 DTD 생성기를 구현하였다. 포괄적 DTD 생성기는 온톨로지 프로세서와 DTD 생성 프로세서로 구성된다. 온톨로지 프로세서는 온톨로지의 개념과 속성을 파싱하여 개념 및 속성 클래스를 생성하고, DTD 생성 프로세서는 생성된 개념 및 속성 클래스를 이용하여 포괄적 DTD를 생성한다. 생성된 포괄적 DTD는 수많은 XML 문서중에서 검색 대상이 되는 XML 문서를 사전에 여과한다. 그림 1은 이와 같은 문서여과 알고리즘의 전체적인 구성을 나타낸다.

온톨로지로부터 생성되는 포괄적 DTD는 검색에 적합한 영역의 문서들을 사전에 선별함으로써 불필요한 문서로의 접근을 제거하여 정보검색의 효율을 증대시킨다. 또한 본 논문은 작은 크기의 XML 문서들이 대규모로 존재하는 정보시스템을 가정하므로, 하나의 영역에 대하여 이미 생성된 포괄적 DTD는 다량의 XML 문서에 모두 적용될 수 있다.

본 논문은 2장에서 연구배경을 설명하고 3장에서 포괄적 DTD 생성 및 문서여과의 세부 알고리즘을 설명한다. 4장에서는 XML 문서여과의 효과를 예제 XML 문서에 적용해 보았다. 아직까지는 제안된 알고리즘의 성능평가를 위한 표준 데이터베이스가 없기 때문에, 본 실험에서는 연구진에 의해 다양한 구조로 작성된 예제를 사용하였다.

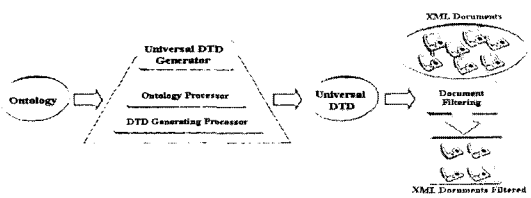


그림 1. 온톨로지 기반의 문서여과 과정

2. 배경연구

본 장에서는 XML 정보검색을 위한 전처리 기법으로 제안한 문서여과에 사용된 XML과 온톨로지에 대하여 간단히 설명한다. 또한 문서여과를 위해 연구된 기존 방법들을 살펴보고 본 연구의 접근 방법을 기술한다.

2.1 XML

XML은 시맨틱 웹이라는 개념과는 별개로 HTML의 비구조성을 극복하기 위해 제시된 언어로서, 구조화된 문서의 작성을 가능하게 한다. XML은 문서 생성시 태그의 이름을 사용자가 자유롭게 정의할 수 있기 때문에, 태그에 의미정보를 부여할 수 있다[1, 12]. 이러한 XML의 구조적 표현은 XML 문서의 정보검색을 가능하게 하며, 의미를 부여한 태그의 정의는 의미적 정보검색을 가능하게 한다. 그러나 XML 문서는 서로 다른 사람이 같은 내용의 문서를 작성하더라도, 같은 의미를 뜻하면서도 다양한 이름의 태그가 정의될 수 있으므로 상호 운용에 어려움이 따른다. 또한 같은 주제에 대해서 XML 문서의 내용은 동일하지만 구조가 다양하게 표현되므로, 인간 또는 컴퓨터가 XML 문서를 일관된 프로그램으로 처리하기가 매우 어렵다. 이를 위하여 XML 문서의 구조를 정의한 DTD(document type definition)가 사용되는데, DTD 역시 사용자에 의해 다양한 구조로 정의될 수 있으므로 XML 문서의 효율적인 운용을 위해서는 공통으로 적용될 수 있는 구조적 표준이 필요하다.

2.2 온톨로지

온톨로지는 단어와 관계들로 구성된 사전으로서, 특정 영역에 관련된 단어들에 대하여 상위개념과 하위개념, 그들 간의 관계를 계층구조로 표현하고 추가적으로 이를 확장할 수 있는 추론 규칙을 포함한다. 온톨로지의 역할은 서로 다른 데이터베이스가 같은 개념에 대해서 서로 다른 단어나 식별자를 사용할 경우에 이를 동일 개념으로 처리하고, 웹 기반의 지식처리나 응용 프로그램 사이의 지식공유 및 재사용을 가능하게 한다 [10, 11, 14]. 앞서 살펴본 XML은 서로 다른 태그이름이 같은 의미를 갖는다는 정보를 별도로 표현해야 하며, 동일 영역의 주제에 대하여 여러 구조로 문서가 표현된다. 이 경우 문서의 내용은 동일하지만 다양한 구조를 사람이나 컴퓨터가 인지하여 처리하기가 매우 어려우므로, 온톨로지를 이용한 연구방법이 매우 중요하게 부각된다. 현재 응용영역에 대한 온톨로지 표준화 작업이 W3C 등에서 활발히 진행되고 있으므로, 가까운 미래에 문서

의 다양한 구조에 기인한 문제를 극복할 수 있는 기반이 제공될 것이다.

2.3 기존 문서여과 방법

XML 정보검색을 위한 전처리 방법으로 기존에 연구된 문서여과 방법은 사용자의 질의에 대하여 문서의 적합도를 결정하기 위해 각 XML 문서의 트리구조에 대하여 인가된 질의를 분석한다. 즉, 질의에 포함된 키워드나 속성값들이 문서의 구조에 정확하게 포함되어 있는지를 확인하기 위해 XML 문서를 구조적으로 파싱하거나, 각 XML 문서와 질의에서 시그너쳐 정보를 생성하여 문서와 질의 사이의 유사도를 계산함으로써 문서의 적합성 여부를 판별한다[8, 9]. 이러한 방법은 각 질의에 대하여 독립적으로 문서여과가 수행되므로 수많은 질의와 다량의 XML 문서를 고려할 때 매우 비효율적인 방법이 될 수 있다.

본 논문은 보다 효과적인 문서여과를 수행하기 위해 동일 영역의 구조를 정의한 온톨로지를 사용하였다. 온톨로지는 응용영역에 대한 공통구조를 정의하고 있으므로, 이를 적용하면 다양한 구조의 XML을 일관된 방식으로 운용할 수 있는 방법이 제공될 수 있다. 또한 이러한 온톨로지로부터 특정 영역에 적용할 수 있는 DTD를 생성하면, 수많은 XML 문서 중에서 검색을 위한 관심영역의 문서만을 선별할 수 있게 된다. 따라서 본 연구는 온톨로지를 이용하여 특정영역의 모든 XML 문서의 구조를 포함할 수 있는 포괄적 DTD를 생성하고, 생성된 DTD를 이용하여 사전에 검색영역에 포함되는 문서만을 선별함으로써, XML 정보검색의 효율을 도모할 수 있는 알고리즘을 개발하고자 하였다.

3. 포괄적 DTD를 이용한 XML 문서 여과

XML 정보 검색은 특정 영역의 XML 문서만을 대상으로 하기 때문에 해당 XML 문서를 선별하고 불필요한 문서는 여과해야 한다. 본 논문에서는 정보검색 대상이 되는 XML 문서를 선별하기 위해 온톨로지의 개념구조 및 연관관계를 고려하여 포괄적 DTD를 생성하였다.

온톨로지는 특정영역을 정의하기 위해 개념, 개념 계층구조, 속성 그리고 연관관계 등으로 표현된다. 표 1은 본 논문에서 문서여과 및 질의확장을 위해 Frame-Logic 문법을 이용하여 설계한 "대학연구센터(College-Research-Center)" 온톨로지의 내용이며, 그림 2는 이러한 온톨로지의 개념구조와 개념 및 속성 간의 연관관계를 나타낸 것이다.

포괄적 DTD 생성모듈은 그림 3과 같이 온톨로지 프로세서와 DTD 생성 프로세서로 구성된다. 온톨로지를 입력으로 하여 각 프로세서의 처리과정을 통해 포괄적 DTD가 생성된다. 생성되는 포괄적 DTD는 동일영역에 대한 다양한 구조의 모든 XML 문서들을 포함할 수 있는 문서의 구조를 정의한 DTD로써 문서여과에 사용될 수 있다. 각 프로세서의 세부 내용은 표 1과 그림 2에 제시된 온톨로지를 사용하여 3.1절과 3.2절에서 세부적으로 설명하겠다.

3.1 온톨로지 프로세서

온톨로지는 표 1의 예와 같이 개념의 계층구조를 나타내는 부분과 개념에 포함된 속성 및 속성과 다른 개

표 1. "대학연구센터" 온톨로지의 내용

```
Object[].
        ResearchCenter :: Object.
                Person :: ResearchCenter.
                        Professor :: Person.
                        Engineer :: Person.
                        Student :: Person.
                                MasterStudent :: Student.
                                        PhDStudent :: MasterStudent.
                Project :: ResearchCenter.
                        Product :: Project.
                        Paper :: Project.
                                Journal :: Paper.
                                Conference :: Paper.
                        Patent :: Project.

Person[name =>> STRING; email =>> STRING; skill =>> STRING].
Professor[professorID =>> NUMBER; attendLecture =>> STRING; project =>> Project; paper =>> Paper].
Engineer[project =>> Project; company =>> STRING].
Student[studentID =>> NUMBER; takeLecture =>> STRING; paper =>> Paper; project =>> Project].
MasterStudent[major =>> STRING; assistant =>> STRING].
PhDStudent[attendLecture =>> STRING].
Project[title =>> STRING; chief =>> Professor; member =>> Student; technicalAdvisor =>> Engineer; skill =>> STRING; background =>> STRING].
Product[developementTool =>> STRING; developer =>> Person].
Paper[title =>> STRING; author =>> Person; abstract =>> STRING; year =>> NUMBER].
Journal[volume =>> NUMBER; number =>> NUMBER].
Patent[register =>> Person; number =>> NUMBER].
```

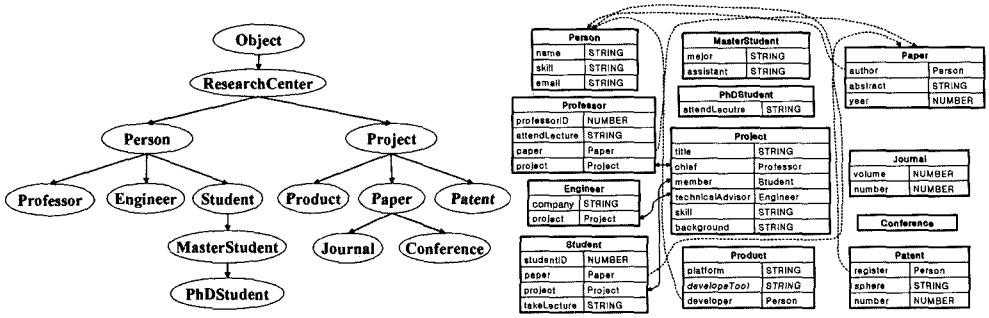


그림 2. "대학연구센터" 온톨로지의 개념 및 속성 관계

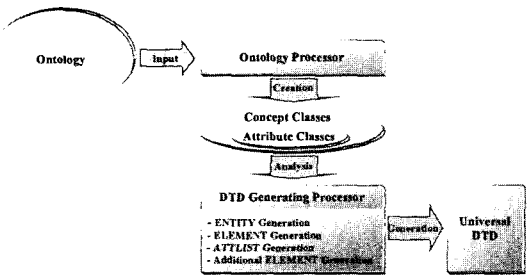


그림 3. 포괄적 DTD 생성 과정

념과의 관계를 나타내는 부분으로 구성되며, Frame-Logic의 구문형식으로 표현되어 텍스트 파일로 저장되어 있다. 그림 3의 포괄적 DTD 생성 과정에서, 온톨로지 프로세서는 이러한 텍스트 형식의 온톨로지를 입력받아 "::, ;, [,], =>>" 등의 기호들로 구분하여 온톨로지를 파싱한 후, 개념과 속성을 분리해 낸다. 분리된 개념과 속성은 자바 언어에서 제공하는 벡터형식의 개념클래스와 속성클래스로 저장된다. 생성된 개념클래스와 속성클래스는 DTD 생성 프로세서에서 사용되며 3.2절의 여러 알고리즘에 의해 일정한 형식을 갖는 DTD로 표현된다.

3.2 DTD 생성 프로세서

DTD 생성 프로세서는 온톨로지 프로세서에서 생성된 개념클래스와 속성클래스를 바탕으로 ENTITY 생성 알고리즘, ELEMENT 생성 알고리즘, ATTLIST 생성 알고리즘, 추가 ELEMENT 생성 알고리즘을 적용하여 포괄적 DTD를 생성한다. 다음은 각각의 세부알고리즘이다.

3.2.1 ENTITY 생성 알고리즘

온톨로지의 개념들은 계층구조를 이루며, 계층구조는 개념간의 상속관계를 나타낸다. 각 개념들의 상속관계를 알기 위해 온톨로지의 개념들을 순차적으로 탐색한다. 개념이 선택되면 해당 개념에 대하여 ENTITY를 선언하고 자신의 하위개념들을 탐색하여 ENTITY의 원소로 추가한다. ENTITY 생성 알고리즘은 표 2와 같다.

표 2. ENTITY 생성 알고리즘

1. 온톨로지의 개념을 순차적으로 선택한다.
2. 선택한 개념을 ENTITY로 ENTITY-List에 정의한다.
3. 선택한 개념의 하위개념들을 탐색한다.
4. 탐색된 하위개념들을 정의된 ENTITY의 원소로 추가한다.
5. 단계 1~4를 반복한다.

3.2.2 ELEMENT 생성 알고리즘

온톨로지의 개념을 순차적으로 탐색하여 ELEMENT로 선언하고 자신의 모든 상위개념으로부터 상속받은 모든 속성들과 자신의 속성들을 ELEMENT의 원소로 추가한다. ELEMENT 생성 알고리즘은 표 3과 같다.

표 3. ELEMENT 생성 알고리즘

1. 온톨로지의 개념을 순차적으로 선택한다.
2. 선택한 개념을 ELEMENT로 ELEMENT-List에 정의한다.
3. 선택된 개념에 대한 하위개념들을 검색하여 순차적으로 선택한다.
4. 선택된 하위개념의 속성들을 정의된 ELEMENT의 원소로 추가한다.
5. 선택된 개념의 모든 하위개념에 대하여 단계 3~4를 재귀적으로 반복한다.
6. 선택된 하위개념의 속성들을 정의된 ELEMENT의 원소로 추가한다.
7. 단계 1~6을 반복한다.

3.2.3 ATTLIST 생성 알고리즘

온톨로지의 각 개념에 대한 속성은 ELEMENT에 정의되고, 또한 ATTLIST에도 정의된다. ATTLIST 생성 알고리즘 역시 개념의 속성을 정의하는 것이므로, ATTLIST 구문 표현법을 제외하면 표 3의 ELEMENT 생성 알고리즘과 같다. 표 4는 ATTLIST 생성 알고리즘이다.

표 4. ATTLIST 생성 알고리즘

1. 온톨로지의 개념을 순차적으로 선택한다.
2. 선택한 개념을 ATTLIST에 정의한다.
3. 선택한 개념의 상위개념을 탐색하여 순차적으로 선택한다.
4. 선택된 상위개념의 속성명, 속성의 CDATA 타입, #IMPLIED 옵션 등을 2에서 정의한 ATTLIST의 새로운 항목으로 추가한다.
5. 선택한 개념의 모든 상위개념에 대하여 단계 3~4를 반복한다.
6. 단계 1~5를 반복한다.

3.2.4 추가 ELEMENT 생성 알고리즘

추가 ELEMENT 생성 알고리즘은 개념의 각 속성에 대한 ELEMENT를 정의한다. 개념의 속성값은 기본적으로 STRING으로 정의되지만 다른 개념으로 정의될 수도 있다. A라는 개념의 속성 중에 속성값이 개념 B로 정의된 것이 있다면, 개념 B의 속성 중에서 개념 A를 속성값으로 갖는 속성을 탐색한다. 만약 개념 A가 속성값으로 존재하면, 개념 A의 ELEMENT에 매개변수 ENTITY 형태로 개념 B를 추가한다. 알고리즘은 표 5와 같다.

표 5. 추가 ELEMENT 생성 알고리즘

1. 온톨로지상의 개념을 순차적으로 선택한다.
2. 선택한 개념의 모든 속성을 검색하여 순차적으로 선택한다.
3. 선택한 속성을 추가 ELEMENT-List에 정의한다.
4. 개념의 속성값들을 검색한다.
   ①만약 선택한 개념의 속성값이 STRING 또는 NUMBER면 #PCDATA로 정의된 속성의 원소로 추가한다.
   ②만약 속성값으로 다른 개념이 존재하면,
   - 개념리스트에서 해당 개념을 다시 검색한다.
   - 해당 개념의 속성값을 검색하여 단계 1에서 선택했던 개념이 속성값으로 존재하면 단계 3에서 정의한 속성의 원소로 해당 개념을 추가한다.
5. 단계 1~4를 반복한다.

3.3 포괄적 DTD 예

포괄적 DTD는 앞서 설명한 방법에 의해 온톨로지로부터 생성된다. 표 6은 "대학연구센터" 온톨로지로부터 생성된 포괄적 DTD의 일부이며, 동일 영역의 정보를 적절히 내포하고 있는 모든 문서들을 수용하게 된다.

4. 포괄적 DTD를 이용한 XML 문서여과 실험

본 장에서는 포괄적 DTD를 이용하여 다양한 구조의 XML 문서들을 대상으로 문서여과 알고리즘의 효과를 실험하였다. 현재 문서여과의 성능평가를 위한 온톨로지 기반의 표준 데이터베이스가 없기 때문에, 실험을 위해 사용된 예제 XML 문서는 본 연구진에 의해 다양한 구조로 작성되었다. 문서여과 실험은 총 9개의 예제를 사용하였으며, 각 XML 문서에 대하여 여과된 결과를 확인할 수 있도록 결과 화면을 구성하였다. 3장에서 그림 2의 온톨로지로부터 생성된 포괄적 DTD는 "대학연구센터" 영역에 관한 XML 문서만을 포괄할 수 있도록 설계되었으므로, 이를 이용하여 "대학연구센터" 영역의 XML 문서를 선별하는 실험을 수행하였다.

표 7의 예제 example01.xml은 프로젝트의 제목 구성원 등의 항목으로 "대학연구센터" 프로젝트 관련 내용으로 구성되며, 표 8과 같은 DTD 구조를 갖는다.

"대학연구센터" 포괄적 DTD는 example01.dtd의 모든 엘리먼트를 포함하므로 그림 4와 같이 example01.xml 문서를 검색 대상 XML 문서로 선별한다.

표 9의 예제 XML 문서 example02.xml은 제품의 제

표 6. "대학연구센터" 포괄적 DTD의 일부

```
.....
<!ENTITY % Person "Person | Student | MasterStudent | PhDStudent | Engineer | Professor" >
<!ENTITY % Student "Student | MasterStudent | PhDStudent"
<!ENTITY % MasterStudent "MasterStudent" >
.....
<!ELEMENT Person (#PCDATA|Student|MasterStudent|PhDStudent|Engineer|Professor|name|email|skill)* >
<!ELEMENT Student (#PCDATA|MasterStudent|PhDStudent|name|email|skill|studentID|takeLecture|paper|project)* >
.....
<!ATTLIST Person
        name                    CDATA   #IMPLIED
        email                   CDATA   #IMPLIED
        skill                   CDATA   #IMPLIED >
.....
<!ELEMENT attendLecture (#PCDATA) >
<!ELEMENT project (#PCDATA | %Project;)* >
<!ELEMENT paper (#PCDATA | %Paper;)* >
.....
```

목, 사용 환경, 제작도구 그리고 구성원 등 업체의 제품 생산에 관련된 내용으로 구성되며, 표 10과 같은 DTD 구조를 갖는다.

example02.xml 문서는 표 7의 example01.xml 문서와 비슷한 구조와 항목을 갖는다. 그러나 Example02에서 "member"라는 항목은 제품을 개발할 기술자, 즉 "Engineer"를 의미하지만, 포괄적 DTD에서는 "member"라는 항목은 "Student"를 의미한다. 이는 포괄적 DTD가 "대학연구센터" 온톨로지를 기반으로 하기 때문에, "member"라는 항목에 대해 "Student, MasterStudent, PhDStudent"를 요구하며, "Engineer"라는 항목은 "member"가 아닌 "technicalAdvisor"을 의미한다. 따라서 example02.dtd와 포괄적 DTD에 사용된 Engineer라는 항목은 사용된 개념 영역이 다르므로 의미적으로 서로 상이하다. 따라서 example02.xml은 포괄적 DTD에 의한 여과 과정에서 그림 5와 같이 검색에 적합하지 않은 XML 문서로 여과된다.

Example01.xml과 example02.xml 문서에 대한 여과 결과는 각각 그림 4, 5와 같이 실행화면으로 나타내었으며, 동일한 방법으로 다른 예제 XML 문서에 대한 유효성 여부를 실험하였다. 표 11은 문서여과 실험에서 추가로 사용된 예제 XML 문서의 내용과 구조이다.

표 11의 (a), (b), (c), (d), (e), (f) 그리고 (g)의 XML 문서는 "대학연구센터" 온톨로지와 비교해 볼 때 매우 다양한 구조를 갖는다. (a)의 example03.xml은 "대학연구센터"에 소속된 "Person"과 "Project"에 관한 내용으로 구성되어 있으며, (b)의 example04.xml은 개념 "Person"과 하위개념들로 구성되어 있다. (c)의 example05.xml은 오직 "Product"라는 단일 개념으로 구성되어 있다. 또한 표 (d)의 example06.xml는 온톨로지에서는 개념 "Person"에 포함되어 있는 "Master Student", "PhDStudent" 그리고 "Student" 등이 개념 "Project"의 속성인 "member"의 하위개념에 포함되어 있으며, (e)의 example07.xml은 개념 "Person"이 "Paper"의 하위에 포함되어 있다. 이러한 다양한 구조의 문서들은 "대학연구센터" 온톨로지에 비해 복잡하거나 표면적으로 상이하게 보일 수 있으나, 모두 "대학연구센터" 온톨로지에 포함되는 유효한 문서로 선별되었다.

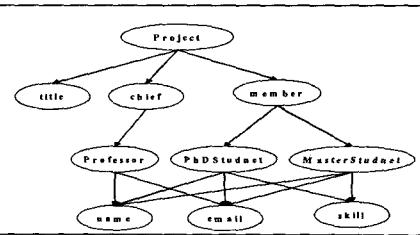
표 11(f)의 example08.xml은 학사업무와 관련된 구성원들에 관한 내용으로서, "Professor", "Student", "MasterStudent" 그리고 "PhDStudent" 등으로 구성되어 있으며, "대학연구센터" 온톨로지의 개념 "Person"에 포함되어 있는 개념과 동일하다. 그러나 연구센터에 관련된 구성원의 속성과 학사 업무와 관련된 구성원들의 속성에는 차이가 발생한다. 또한 (g)의 example09.xml은 제품, 논문, 프로그램, 등록날짜, 제작자, 고유번호 그리고 제목 등 특허와 관련된 내용으로 구성되어 있으며, "대학연구센터"에 속한 "Project" 또는 "Person"의 하위개념들의 일부를 포함하고 있다. 비록 example09.xml을 구성하는 개념들이 온톨로지에 속한 개념들로만 구성되어 있지

표 7. example01.xml

```
<Project>
  <title>Laser Slice</title>
  <chief>
    <Professor>
      <name>Y.H.Kong</name>
      <email>yhkong@sch.ac.kr</email>
    </Professor>
  </chief>
  <member>
    <PhDStudent>
      <name>M.S.Kim</name>
      <email>mskim@sch.ac.kr</email>
      <skill>C++</skill>
    </PhDStudent>
    <MasterStudent>
      <name>H.J.Lee</name>
      <skill>C++</skill>
    </MasterStudent>
  </member>
  <title>Semantic XML Query</title>
  <chief>
    <Professor>
      <name>Y.H.Kong</name>
      <email>yhkong@sch.ac.kr</email>
    </Professor>
  </chief>
  <member>
    <MasterStudent>
      <name>K.S.Lee</name>
      <email>kslee@sch.ac.kr</email>
      <skill>XML</skill>
      <skill>Java</skill>
    </MasterStudent>
  </member>
</Project>
```

표 8. example01.dtd

```
<!ELEMENT Project (title|chief|member)*>
<!ELEMENT Professor (name|email)*>
<!ELEMENT title (#PCDATA)>
<!ELEMENT chief (#PCDATA|Professor)*>
<!ELEMENT member (PhDStudent|MasterStudent)*>
<!ELEMENT PhDStudent (name|email|skill)*>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT skill (#PCDATA)>
<!ELEMENT MasterStudent (name|email|skill)*>
```



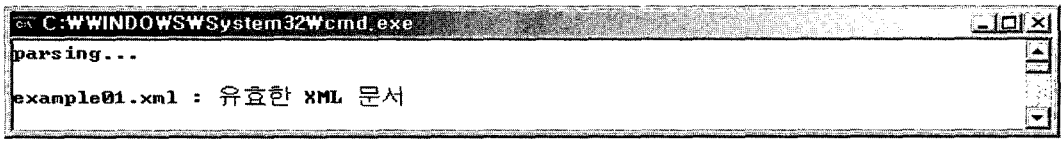


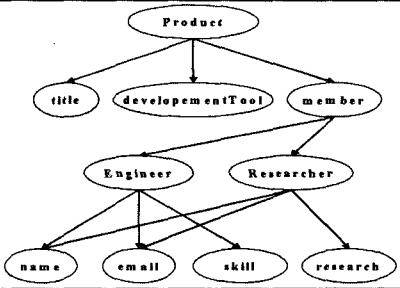
그림 4. 선택된(selected) XML 문서

표 9. example02.xml

```
<Product>
  <title>DSP Board Tester</title>
  <developementTool>Matlab
  </developementTool>
  <member>
     <Engineer>
        <name>B.I.Jin</name>
        <email>bij@mail.test</email>
        <skill>DSP</skill>
     </Engineer>
     <Researcher>
        <name>T.H.Lee</name>
        <email>thl@mail.test</email>
        <research>Logic Circuit</research>
     </Researcher>
  </member>
  <title>Garbage Process Eraser</title>
 <developementTool>Java</developementTool>
  <member>
     <Researcher>
        <name>K.S.Lee</name>
        <email>ksl@mail.test</email>
        <research>Operating System
        </research>
     </Researcher>
     <Engineer>
        <name>C.W.Lee</name>
        <email>cwl@mail.test</email>
        <skill>Unix Programming</skill>
     </Engineer>
  </member>
</Product>
```

표 10. example02.dtd

```
<!ELEMENT Product (title|developementTool|member)*>
<!ELEMENT title (#PCDATA)>
<!ELEMENT developementTool (#PCDATA)>
<!ELEMENT member (Engineer|Researcher)*>
<!ELEMENT Engineer (name|email|skill)*>
<!ELEMENT Researcher (name|email|research)*>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT skill (#PCDATA)>
<!ELEMENT research (#PCDATA)>
```



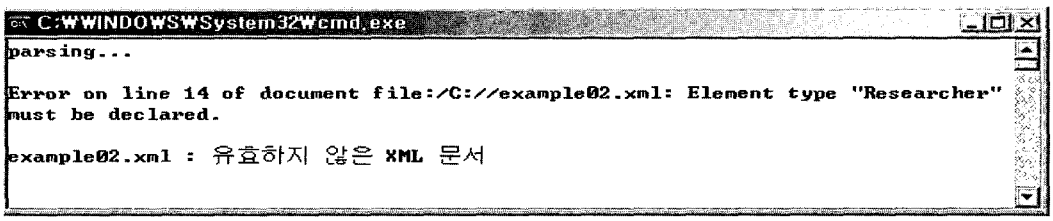


그림 5. 여과된(filtered-off) XML 문서

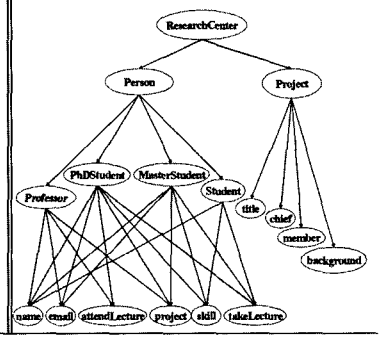
만, 서로 다른 영역에 대한 내용을 표현한 문서이기 때문에 개념구조나 속성의 차이가 발생한다. 따라서 example08.xml, example09.xml은 "대학연구센터"에 포함되지 않는 부적합한 문서로서 검색 대상에서 제거되었다.

다양한 구조의 예제 XML 문서들을 대상으로 검색 대상 문서를 선별한 결과, example02.xml, example08.xml 그리고 example09.xml은 검색 대상에서 제외되고 나머지 6개의 문서는 "대학연구센터" 온톨로지의 영역에 대하여 검색에 적합한 문서로 선별되었다. 비록 개념 항목이나 그 구조가 유사한 문서라 해도 관심영역의 문서가 아닐 경우, 그 문서는 검색에 적용될 수 없는 부적합한 문서가 된다. 따라서 포괄적 DTD에 의한 XML 문서여과는 검색에 필요한 문서들을 사전에 효과적으로 선별함으로써 보다 효율적인 정보검색을 수행할 수 있도록 한다.

표 11. 문서여과 실험을 위한 예제 XML 문서들의 내용과 구조

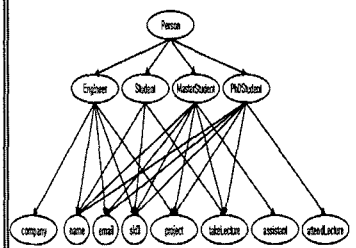
(a) example03.xml

```
<ResearchCenter>
 <Person>
  <Professor>
   <name>Y.H.Kong</name>
   <email>yhkong@sch.ac.kr</email>
   <attendLecture>Artificial intelligence
</attendLecture>
   <project>Laser Slice</project>
   <project>Semantic XML Query</project>
  </Professor>
  <PhDStudent>
   <name>M.S.Kim</name>
   <skill>C++</skill>
   <takeLecture>Artificial
 Intelligence</takeLecture>
   <attendLecture>Assembly</attendLecture>
   <project>Laser Slice</project>
  </PhDStudent>
  <Student>
   <name>J.K.Shin</name>
   <skill>Visual Basic</skill>
   <takeLecture>C</takeLecture>
  </Student>
 </Person>
 <Project>
  <title>Semantic XML Query</title>
  <chief>Y.H.Kong</chief>
  <member>K.S.Lee</member>
  <skill>Java</skill>
  <background>XML</background>
 </Project>
 <Project>
  <title>Data Structure</title>
  <chief>Y.H.Kong</chief>
  <member>K.S.Lee</member>
  <member>J.K.Shin</member>
 </Project>
</ResearchCenter>
```



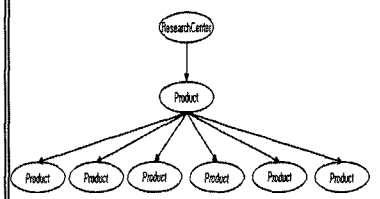
(b) example04.xml

```
<Person>
 <Engineer>
  <name>S.H.Yun</name>
  <email>shyun@sch.ac.kr</email>
  <company>AI.LAB</company>
  <skill>C</skill>
  <skill>Java</skill>
  <project>Semantic XML Query</project>
 </Engineer>
 <Engineer>
  <name>H.D.Kwon</name>
  <email>hdkwon@sch.ac.kr</email>
  <company>AI.LAB</company>
  <skill>C++</skill>
  <skill>C#</skill>
  <skill>Java</skill>
  <project>Semantic XML Query</project>
 </Engineer>
 <Student>
  <name>J.K.Shin</name>
  <skill>Visual Basic</skill>
  <takeLecture>C</takeLecture>
 </Student>
 <Student>
  <name>S.G.Min</name>
  <skill>C</skill>
  <takeLecture>Visual Basic</takeLecture>
 </Student>
 <MasterStudent>
  <name>K.S.Lee</name>
  <email>kslee@sch.ac.kr</email>
  <skill>C</skill>
  <skill>Java</skill>
  <takeLecture>Artificia Intelligebce
  </takeLecture>
  <project>Semantic XML Query</project>
  <assistant>S.H.Yun</assistant>
  <assistant>H.D.Kwon</assistant>
 </MasterStudent>
 <PhDStudent>
  <name>M.S.Kim</name>
  <email>mskim@sch.ac.kr</email>
  <skill>C++</skill>
  <skill>Open GL</skill>
  <takeLecture>Artificial Intelligence
  </takeLecture>
  <attendLecture>C</attendLecture>
  <project>Laser Slice</project>
 </PhDStudent>
</Person>
```



(c) example05.xml

```
<ResearchCenter>
 <Product>
  <title>SQL Processor System</title>
  <member>Y.S.Hong</member>
  <technicalAdvisor>Y.H.Kong
  </technicalAdvisor>
  <skill>VB</skill>
  <developementTool>MS VS6.0, MS-SQL
  </developementTool>
  <developer>H.K.Kang</developer>
 </Product>
 <Product>
  <title>XML Processor2.0</title>
  <member>H.D.Kwon</member>
  <technicalAdvisor>Y.H.Kong
  </technicalAdvisor>
  <skill>JAVA</skill>
  <developementTool>JDK1.3
  </developementTool>
  <developer>H.D.Kwon</developer>
 </Product>
 <Product>
  <title>WebSearch Engineen ver.3.5
  </title>
  <member>M.S.Kim</member>
  <technicalAdvisor>Y.H.Kong
  </technicalAdvisor>
  <skill>C++</skill>
  <developementTool>Visual C++
  </developementTool>
  <developer>M.S.Kim</developer>
 </Product>
</ResearchCenter>
```



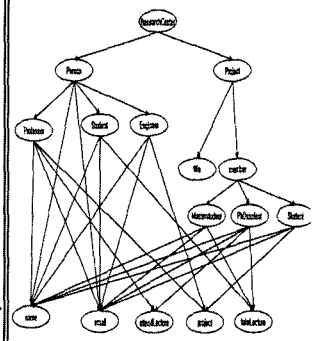
5. 결 론

최근 인터넷상에서의 자료 교환을 위한 표준으로 자리 잡은 XML은 사용자들이 자유롭게 정의하는 태그를 사용하여 표현되며, 문서에 따라 다양한 구조를 가지고 있다. 현재 XML을 대상으로 한 XML 정보검색에 관한 연구가 활발히 수행되고 있으나, 이러한 연구는 대량의 XML 문서에 대하여 수행되므로 검색의 효율은 검색 대상 문서의 양과 질의에 종속적으로 결정된다. 질의검색의 효율을 높이기 위해, 기존에 연구된 XML 문서의 전처리 방법들은 질의에 포함된 키워드나 속성값들이 문서의 구조에 정확하게 포함되어 있는지를 확인하기 위해 XML 문서를 구조적으로 파싱하거나, 각 XML 문서와 질의에서 시그너쳐 정보를 생성하여 문서

표 11. 문서여과 실험을 위한 예제 XML 문서들의 내용과 구조(계속)

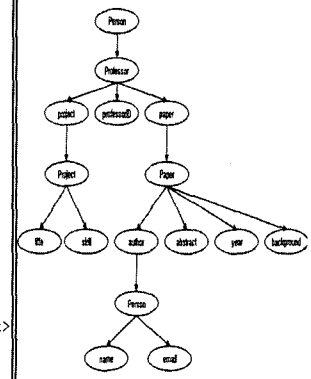
(d) example06.xml

```
<ResearchCenter>
  <Project>
    <title>XML Query Expander</title>
    <member>
      <Student>
        <name> yshyub </name>
        <email> ysh@sch.ac.kr </email>
        <takeLecture> Compiler </takeLecture>
      </Student>
      <Student>
        <name> khDony </name>
        <email> dony@sch.ac.kr </email>
        <email> dony@hotmail.com </email>
        <takeLecture>Object Oriented
        </takeLecture>
        <takeLecture>C-Language</takeLecture>
      </Student>
      <MasterStudent>
        <name> Leeks </name>
        <email> lks@sch.ac.kr </email>
      </MasterStudent>
      <PhDStudent>
        <name> kimsuk </name>
        <email>jks@sch.ac.kr</email>
        <attendLecture> Object Oriented
        </attendLecture>
        <attendLecture> Wireless Internet
        </attendLecture>
        <takeLecture>Data Structure
        </takeLecture>
      </PhDStudent>
    </member>
  </Project>
  <Person>
    <Professor>
      <name> yhkong </name>
      <email>yhkong@sch.ac.kr</email>
      <project> <Project>
          <title> Shopping Mall </title>
          <member><Student>
              <name>Kim chul soo</name>
              <email>ksu@sch.ac.kr</email>
          </Student> </member>
      </Project> </project>
    </Professor>
    <Engineer>
      <name> Hong gildong </name>
      <email> jgd@sch.ac.kr</email>
      <project> <Project>
          <title> Autimation System </title>
          <member> <MasterStudent>
              <name>Leekyungsoo</name>
              <email>kslee@sch.ac.kr</email>
          </MasterStudent></member>
          <member>AI-Lab Students
          </member>
      </Project></project>
    </Engineer>
  </Person>
</ResearchCenter>
```



(e) example07.xml

```
<Person>
  <Professor> <project> <Project>
      <title>A Novel Accurate design method</title>
      <skill>PhotoShop</skill>
    </Project> </project>
    <professorID>1319</professorID>
    <paper> <Paper>
      <author> <Person>
          <name>s.j.Lee</name>
          <email>sjlee@sch.ac.kr</email>
      </Person> </author>
      <abstract>Controlling home appliances
      using XML</abstract>
      <year>2004</year>
      <background>XML</background>
  </Paper> </paper> </Professor>
  <Professor><project> <Project>
      <title>Q and A ActiveX Component</title>
      <skill>MySQL</skill>
    </Project> </project>
    <professorID>1331</professorID>
    <paper> <Paper>
      <author> <Person>
          <name>d.ahn</name>
          <email>dahn@ramrec.sch.ac.kr</email>
      </Person></author>
      <abstract>Design Method of a Dual
      Band Balun and Divider</abstract>
      <year>2002</year>
      <background>Divider</background>
  </Paper> </paper> </Professor>
  <Professor> <project> <Project>
      <title>Electric Commerce Using Java
      Expert System Library</title>
      <skill>Java</skill>
    </Project> </project>
    <professorID>1348</professorID>
    <paper> <Paper>
      <author> <Person>
          <name>b.g.Kang</name>
          <email>bgkang@sch.ac.kr</email>
      </Person> </author>
      <abstract>Performance of an OFDM/CDMA
      system with AutoregressiveModeling</abstract>
      <year>1997</year>
      <background>CDMA</background>
  </Paper> </paper> </Professor>
</Person>
```



(f) example08.xml

```
<Person>
  <Professor>
    <name>J.D.KIM</name>
    <major>OS</major>
  </Professor>
  <Professor>
    <name>D.E.Y</name>
    <major>Compiler</major>
  </Professor>
  <Professor>
    <name>D.E.LEE</name>
    <major>Data Base</major>
  </Professor>
  <Student> <student>
      <name>K.H.LEE</name>
      <ID>19982334</ID>
      <year>4</year>
  </student> </Student>
  <Student> <student>
      <name>M.Y.KIM</name>
      <ID>19983245</ID>
      <year>4</year>
  </student></Student>
    <Student> <MasterStudent>
      <name>K.H.LEE</name>
      <ID>19962334</ID>
      <year>2</year>
      <project>
        <member>9</member>
        <skill>JAVA</skill>
      </project>
  </MasterStudent></Student>
  <Student><PhDStudent >
      <name>K.E.JUNG</name>
      <ID>19942334</ID>
      <year>3</year>
      <project>
        <member>6</member>
        <skill>C+ + </skill>
      </project>
  </PhDStudent ></Student>
</Person>
```

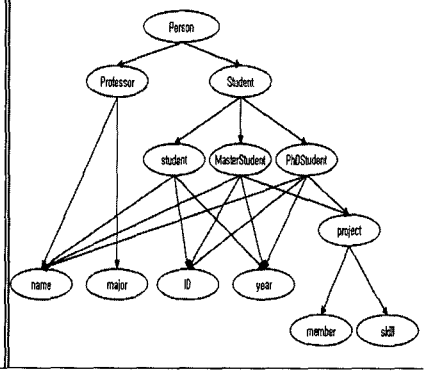
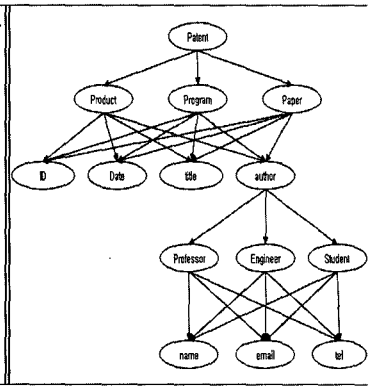


표 11. 문서여과 실험을 위한 예제 XML 문서들의 내용과 구조(계속)

(g) example09.xml

```
<Patent>
  <Product>
    <ID>2004 - 01 - 187 - 002715</ID>
    <Date> 2004.05.04></Date>
    <title> MP3 playler </title>
    <author>
      <Student>
        <name>S.H.Yun</name>
        <email>shyun@sch.ac.kr</email>
        <tel>011-3333-4444</tel>
      </Student>
    </author>
  </Product>
  <Program>
    <ID>2004 - 01 - 187 - 002777</ID>
    <Date> 2004.05.15 </Date>
    <title> XML System </title>
    <author>
      <Student>
        <name>H.D.Kwon</name>
        <email>hdkwon@sch.ac.kr</email>
        <tel>016-5555-6666</tel>
      </Student>
    </author>
  </Program>
  <Paper>
    <ID>2004 - 01 - 187 - 002812</ID>
    <Date> 2004.06.01 </Date>
    <title>XML Document Filting and
    Query Expand</title>
    <author>
      <Engineer>
        <name>M.S.Kim</name>
        <email>mskim@sch.ac.kr</email>
        <tel>016-7777-8888</tel>
      </Engineer>
    </author>
  </Paper>
</Patent>
```



와 질의 사이의 유사도를 계산함으로써 문서의 적합 여부를 판별하였다. 이러한 방법은 인가되는 각 질의에 대하여 수행되어야 하므로, 질의와 XML 문서의 양이 많아지면 매우 비효율적이다. 따라서 본 논문에서는 웹 상의 수많은 XML 문서를 대상으로 정보검색을 수행하기 위해서 관심영역의 XML 문서들을 사전에 선별할 수 있는 보다 개선된 알고리즘을 제안하였다.

제안된 알고리즘은 특정 영역에 대한 구조를 정의한 온톨로지를 사용하여 동일 영역에서 다양한 구조의 XML 문서를 포함할 수 있는 DTD를 생성하는 포괄적 DTD 생성 모듈을 구현하였다. 포괄적 DTD 생성 모듈은 온톨로지 프로세서와 DTD 생성 프로세서로 구성되며, 온톨로지 프로세서는 온톨로지의 개념과 속성을 파싱하여 개념 클래스와 속성 클래스를 생성한다. 또한 DTD 생성 프로세서는 온톨로지 프로세서에서 생성된 개념 클래스와 속성 클래스를 바탕으로 ENTITY 생성 알고리즘, ELEMENT 생성 알고리즘, ATTLIST 생성 알고리즘, 추가 ELEMENT 생성 알고리즘을 적용하여 포괄적 DTD를 생성하였다.

포괄적 DTD를 이용한 문서여과 실험은 실험에 사용된 총 9개의 예제 XML 문서 중에서 검색 영역에 포함되지 않는 3개의 문서를 여과하였다. 여과된 문서에 정의된 개념 및 속성들은 "대학연구센터" 온톨로지의 그것들과 매우 흡사하였지만, 다른 영역의 구조와 내용을 표현한 문서이기 때문에 검색 대상에서 효과적으로 여과되었다. 결과적으로 포괄적 DTD에 의한 문서여과 알고리즘은 대상 XML 문서를 사전에 선별함으로써 검색에 불필요한 문서를 효과적으로 제거하였다. 또한, 한번 생성된 포괄적 DTD는 동일 영역의 XML 문서여과에 재사용이 가능하며, 선별된 XML 문서에 대하여 질의 검색을 수행한다면, 반복적으로 인가되는 다양한 질의에 대하여 검색의 정확도를 향상시킬 수 있을 뿐 아니라, 검색에 부적합한 수많은 문서로의 접근을 제거하므로 검색의 효율을 증대시킬 수 있다.

참고문헌

[1] T. Bray, et al., "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C Recommendation, http://www.w3.org/TR/2000/REC-xml/, February, 2004.

[2] Y. Diao and M. Franklin, "Query Processing for High-Volume XML Message Brokering", Proceedings of the 29th VLDB Conference, 2003.

[3] J. Kim, et al., "Efficient structural joins with clustered extents", Information Processing Letters 91, pp. 69-75, 2004.

[4] M. Altinel and M. J. Franklin, "Efficient Filtering of XML Documents for Selective Dissemination of Information", Proceedings of the 26th VLDB Conference, Cairo, Egypt, pp. 53-64, 2000.

[5] S. Boag, et al., "XQuery 1.0: An XML Query Language", W3C Working Draft 29, http://www.w3.org/TR/xquery/, October, 2004.

[6] A. Deutsch, et al., "XML-QL: A Query Language for XML", Submission to the World Wide Web Consortium, August, 1998.

[7] Y. D. Chung, et al., "Efficient Preprocessing of XML queries using structured signatures", Information Processing Letters 87, pp. 257-264, 2003.

[8] C. Y. Chan, et al., "Efficient Filtering of XML documents with XPath expression", The VLDB Journal, pp. 354-379, 2002.

[9] V. R. Bennjamins, et al., "(KA)2:Building Ontologies for the Internet : a Mid Term Report", Int. Journal of

Human-Computer Studies(IJHCS), No.51, pp. 687-612, 1999.

[10] D. L. McGuinness and F. V. Harmelen(eds.), "OWL Web Ontology Language Overview", W3C Recommendation, 10, February, 2004.

[11] M. Erdmann and R. Studer, "Ontologies as Conceptual Models for XML Document", KAW '99-Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management, 1999.

[12] S. Decker, et al., "The Semantic Web-on the respective Roles of XML and RDF", IEEE internet Computing, Vol.4, No.5, pp. 63-73, 2000.

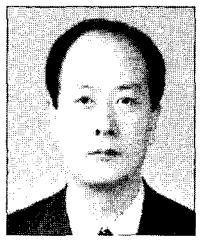
[14] T. R. Gruber, "A Translation Approach to Portable Ontologies Specifications", Knowledge Acquisition, Vol.5, No.2, pp. 199-220, 1993.

김 명 숙(Myung-Sook Kim) **[정회원]**

- 1995년 : 순천향대학교 전산학과 (공학사)
- 1999년 : 순천향대학교 전산학과 (공학석사)
- 2002년~현재 : 순천향대학교 전산학과 박사과정

<관심분야>
웹응용, 영상처리, 알고리즘, 인공지능 등

공 용 해(Yong-Hae Kong) **[정회원]**



- 1982년 : 연세대학교 전자공학과 (공학사)
- 1986년 : 미국 Polytechnic Univ. 전산학과 (공학석사)
- 1991년 : 미국 Polytechnic Univ. 전산학과 (공학박사)
- 1982년 : 한진중공업 연구원
- 1983년 : 삼성전자 연구원
- 1991년~현재 순천향대학교 정보기술공학부 교수

<관심분야>
시멘틱웹, 신경회로망, 멀티미디어 등