# 인터넷 멀티미디어 전자 문서 ·질의 언어의 설계

° 김용훈*, 연제원*, 장동준*, 조정수*, 이강찬*, 이규철*
김완석**
*충남대학교 컴퓨터공학과 데이타베이스 연구실
** 한국전자통신연구원 컴퓨터.소프트웨어 기술연구소 데이터공학연구부

# Design of A Query Language Internet Electronic Documents

°Yonghun Kim*, Jewon Yeon*, Dongjun Jang*, Jungsu Cho*, Kangchan Lee*, Kyuchul Lee*
Wanseok Kim**,
*Database Lab, Dept. of Computer Engineering , Chungnam National University
**Department of Data Engineering, ETRI-Computer & Software Technology Lab

요 약

21 세기는 고도의 정보화 사회가 될 것이다. 이러한 정보화의 사회의 가장 중요한 요소는 수많은 문서 정보를 전자 문서로 만들고 관리하는 것인데, 최근 들어 OIS(Office Information System), 디지털 도서관, CALS/EC 등의 다양한 응용 분야에서 이러한 전자 문서의 저장, 처리, 검색이 요구되고 있다. 이러한 상황에서 다양한 전자 문서의 표준들이 나오고 있고, 그 중 차세대 전자 문서의 표준의 선두로서 XML 이 대두되고 있다. 따라서, 최근에는 XML 문서를 저장하고 검색하는 다양한 응용들이 개발 중이고, 이러한 응용들은 XML 문서의 다양한 문서 정보를 모두 만족할 수 있는 검색 기능들을 요구하고 있다. 본 논문에서는 이러한 다양한 XML 응용들의 요구에 따라서 XML 문서가 지닌 다양한 문서 정보에 대해서 검색할 수 있는 XML 질의 언어의 개발에 목적이 있다.

## 1. 서론

다가오는 21 세기는 고도화된 정보화 사회가 될 것이다. 이런 정보화의 사회에서는 가장 필수적인 요소는 기존의 정보 및 이후에 발생하게 될 각종 문서 정보를 전자 문서화하여 이를 관리하는 것이다. 최근 들어와서, OIS(Office Information System), 전자 도서관(Digital Library), CALS/EC(Commerce At Light Speed/Electronic Commerce) 등의 멀티미디어 문서 정보 처리를 요구하는 다양한 응용 분야의 출현으로 대량의 전자 문서 정보를 효율적으로 저장하고, 처리하며, 검색할 수 있는 정보 서비스가 요구되고 있다.

본 논문은 한국전자통신연구원의 "인터넷 멀티미디어 문서 DBMS 기술 개발" 과제의 일환으로 지원된 위탁과제의 결과물임

이러한 요구에 따라 전자 문서들을 체계적으로 조직, 생성, 전송을 위해서 여러 가지의 다양한 전자 문서의 표준들이 제시되고 있는데, 현재 대표적인 전자 문서로는 HTML (HyperText Markup Language), SGML(Standard Generalized Markup Language), XML(eXtensible Markup Language) 등이 있다. 이 중 HTML 의 한계성과 SGML 의 복잡성을 극복하기 위해 제안된 전자 문서의 표준이 XML 로서, 현재 차세대 인터넷 문서의 표준으로 각광을 받고 있으며, 이에 대한 다양한 연구가 진행되고 있다[1, 2, 3].

이러한 XML 문서를 처리하기 위해서는 XML 문서의 다양한 정보를 손실 없이 저장하기 위한 데이터 모델이 필요 하고, 이를 기반으로 다양한 구조적 정보를 검색할 수 있는 질의 언어가 필요하게 된다. 따라서, 본 논문에서는 이러한 XML 문서의 다양한 구조적 정보를 손실 없이 유지하고 이들에

대해 다양하게 검색할 수 있는 데이터 모델과 검색 질의 언어를 설계하여 제시하였다.

본 논문에서는 먼저 2 장에서는 차세대 인터넷 문서의 표준인 XML 에 대해서 알아보고, 3 장에서는 XML 문서의 다양한 정보를 유지할 수 있는 데이터 모델을 제시하고, 4 장에서는 본 논문에서 제시하고 있는 질의 언어의 기반이 되는 XML 표준의 스펙 중의 하나인 XPointer 를 자세히 알아보고, 5 장에서 제시한 데이터 모델링을 바탕으로 XML 문서의 다양한 구조 정보를 검색할 수 있는 질의 언어를 설계하고 이들의 실제적인 검색 예들을 살펴보고, 마지막으로 결론을 내린다.

## 2. XML

XML 은 HTML 의 한계점과 SGML 의 복잡성을 극복하기 위해 1996 년부터 W3C(World Wide Web Consortium)에서 제안한 차세대 인터넷 문서의 표준이다. XML 을 SGML 과 비교하면, SGML 에서 거의 사용하지 않던 요소는 없애고 꼭 필요한 기능만 수용하였다.

또한 HTML 과 비교하면, 기존의 HTML 을 확장·보완하였기 때문에 HTML 을 그대로 사용할 수 있으며 지금보다 더욱 복잡한 문서의 생성이 가능하며, 구조적인 정보를 포함 할 수 있다.

XML 표준은 다음과 같은 세 가지의 표준으로 나누어 진다.

### 2.1 XML(eXtensible Markup Language)

문서의 구조를 정의하는 부분으로써 문서 구조인 DTD(document type declaration) 기술 방법에 대한 표준안이다. 즉, 어떤 문서의 구조를 기술하는 방법과 그 문서 구조에 맞게 내용을 생성하는 방법을 제시하고 있다.

이 XML 은 기존의 SGML 에서 자주 사용하지 않은 대부분을 생략하였다. 마크업 선언 안에서의 주석문 선언과 내용 모델(content model)에서 inclusion, exclusion, '&' 연결자(connector) 선언 등과 복잡한 공백 문자 처리와 같은 부분을 지원하지 않는다. 따라서, SGML 의 꼭 필요한 중요한 기능만을 XML 에서 제공하기 때문에 SGML 의 서브셋(subset)이라고 말할 수 있다. 따라서, SGML 을 XML 로 변환하기가 용이하고, XML 을 수정 없이 SGML 의 응용에 이용할 수 있다.

XML 이 SGML 과 비교해서 추가된 기능으로는, 모든 XML 이 unicode 를 사용하고, DTD 가 존재하지 않는 인스턴스라도 구문에 맞게 태그 사용이 잘 구성된 문서로 "well-formed XML 문서'를 지원하고 있다.

### 2.2 XLL(XML Linking Language)

인터넷 상의 문서의 링크를 기술 부분 및 문서의 특정 부분을 가리키는 기술 부분을 정의한다. XML 의 링크는 HyTime(Hypermedia/ Time-based Structuring Language)과 확장된 포인터 메커니즘을 포함하는 TEI(Text Encoding Initiative)와 같은 표준들을 기반으로 하며, 다양한 링크 기능을 제공한다. 특히, XLL 은 XLink[4]와 XPointer[5] 두 가지의 표준으로 구분할 수 있다. XLink 는 기존의 HTML 에서의 링크 기능을 확장하여 1:N 의 링크 기능으로까지 확대를 하고 있는데 엘리먼트 선언 시에 XML:LINK 애트리뷰트를 선언하여 링크 기능을 정의할 수 있다. XPointer 는 TEI 의 확장 포인터(extended pointer)로부터 유래한 위치 지정자(locator) 구문을 사용하는데, XML 문서의 엘리먼트에 의해 정의된 엘리먼트 트리상에 바로 접근할 수 있도록 해 준다.

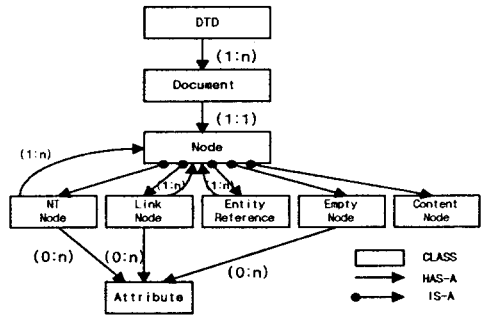
### 2.3 XSL(XML Stylesheet Language)

논리적 구조만 가지고 있는 XML 인스턴스는 외부로 보여지기 위해서 포맷팅 처리가 필요하게 되는데, 이를 제공하기 위해 XML 에서는 SGML 의 포맷팅 언어인 DSSSL(Document Style Semantics and Specification Language)을 간소화해서 사용한다. 따라서, XSL 은 문서의 엘리먼트들과 관련된 포맷팅 정보로부터 포맷 결과를 제공할 수 있도록 해준다. 포맷 결과는 플로우 객체(flow object)로부터 만들어지는 포맷팅 트리에 의해 생성된다.

XSL 의 기본적인 구조는 생성 규칙(construct-ion rule)에 의해 작성된다. 생성 규칙은 특정 엘리먼트를 어떠한 스타일 포맷으로 변환할지 기술하는데 두 부분(즉, pattern 과 action)으로 구성된다. pattern 은 소스 트리내에 있는 특정 엘리먼트를 명시하기 위한 것이며, action 은 플로우 객체를 이용하여 서브 트리의 결과를 지정하기 위한 것이다[6].

## 3. XML 문서의 데이터 모델

XML 문서의 모델링은 데이터베이스 시스템에 문서 정보를 체계적으로 저장하고, 효율적으로 관리하기 위한 선행 단계라 할 수 있다.




[그림 1] XML 문서의 데이터 모델

본 모델(그림 1)의 구조는 하나의 DTD 가 존재하고, 이 DTD 를 통해 수많은 문서가 만들어지며, 하나의 문서는 다양한 형태의 노드들의 트리를 지니고 있고, NT Node 와 Link Node, Empty Node 는 속성을 지니고 있는 형태이다. 노드 트리는 하나의 루

트 노드를 가지고 자식과 형제 관계를 통하여 문서의 구조정보를 가질 수 있는 형태의 노드 트리를 구성하게 된다[7].

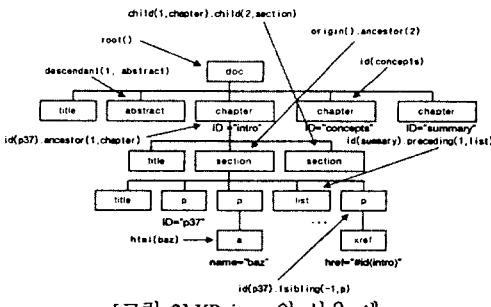
본 모델을 통하여 다양한 문서의 정보를 유지할 수 있는데 내용을 가지고 있는 Content Node 를 통하여 내용 정보 검색을 지원하고, NT Node, Link Node, Entity Reference Node, Empty Node, Content Node 들로 구성된 Node 트리를 통하여 문서의 구조 정보 검색을, Link Node 와 속성들을 통하여 링크 정보 검색을, Attribute 를 통하여 어트리뷰트 검색을 지원하게 된다.

## 4. XPointer(XML Pointer Language)

본 논문에서 제시하는 XML 검색 질의어는 XML 스펙 중의 하나인 XPointer 를 기반으로 하고 있다. 이는 XPointer 가 XLink 와 함께 링크를 정의하는 스펙으로서 일반 사람들이 XML 문서를 작성하는 경우에 링크를 문서에 넣기 위해서는 이러한 XLink 나 XPointer 를 사용해야 하므로 사람들에게 친숙하게 될 것이기 때문이다. 따라서, 본 XML 검색 질의어를 이해하기 위해서는 XPointer 에 대해 보다 자세히 알 필요가 있게 된다.

XPointer 는 Text Encoding Initiative(TEI)에 기반하여 기존에 문서 내지는 fragment identifier 나 ID 를 지닌 부분만을 링크를 하던 것에 반해 엘리먼트 단위의 링크를 가능하게 해 주는 방법으로 다양한 키워드를 통해서 ID 없이도 엘리먼트 구조의 세세한 부분까지 링크 대상으로 지정할 수가 있다.

XPointer 에서는 문서 내부의 세세한 부분까지 지정을 하기 위하여 다양한 키워드를 지원하고 있는데 사용하고 있는 키워드는 문서의 세세한 부분을 지정하기 위한 URI 의 시작 위치를 지정하는 Absolute term 으로 root(), id(), origin()을, Absolute term 으로부터 문서 내부의 세세한 부분까지 지정하기 위해 사용되는 relative term 으로 ancestor(), descendant(), child(), preceding(), following(), psibling(), fsibling()을, 문서 내의 문자열을 지정하기 위한 string(), 어트리뷰트를 이용하기 위한 attr()들이 있다. 이러한 것들을 사용한 예는 다음 그림 2 와 같다.




[그림 2] XPointer 의 사용 예

그림 2 의 XPointer 의 사용 예에서 보는 것과 같이 XPointer 는 다양한 키워드를 이용하여 문서 내의 세세한 부분까지 포인터를 지정할 수 있게 된다. 따라서, 이러한 XPointer 의 특성을 이용하여 본 논문에서는 구조 정보 검색을 위한 검색 질의어로 XQL 을 제안하고자 한다.

## 5. XQL(XML Query Language)

본 논문에서는 XML 문서에 대한 다양한 구조 정보 질의를 위해서 새로운 질의 언어 XQL 을 제시한다. 먼저 내용 정보 검색과 구조 정보 검색을 위하여 4 장에서 설명한 XPointer 기반의 질의 언어를 제시하고, 링크 정보의 검색을 위하여 새로운 링크 질의 언어를 제시한다

### 5.1 XQL 의 구조

XQL 은 XML 문서가 지닌 다양한 구조적 정보에 검색할 수 있는 검색 질의어이다.

[그림 3] XQL 의 BNF

```
XQL ::= XQL (COND XQL)*
XQL ::= XPQL ( ('.' XLQL)* ('.' XPQL)* )*
COND ::= ∪ | ∩ | -
```

XQL 은 내용 검색과 구조 검색을 위해서 XPointer 를 기반으로 하는 XPQL(XPointer based Query Language)와 링크 정보의 검색을 위해 새롭게 추가된 XLQL(XML Link Query Language)로 구성되어져 있다.

본 논문에서 제안하는 XQL 에서는 서로 다른 XQL 질의 사이에 ∪ | ∩ | - 와 같은 조건을 사용하여 결과들을 취합할 수 있도록 설계되어져 있다. 즉, 'XQL ∪ XQL' 과 같은 경우에는 두개의 질의의 결과들의 합을 최종 결과로 추출하는 것이고, 'XQL ∩ XQL'은 두개의 질의의 결과의 교집합을 최종 결과로 추출하게 되는 것이다. 또한, 두 결과의 차(difference)를 위하여 '-' 연산자를 두고 있다.

### 5.2 XPQL 의 구조

XPQL 은 XML 문서의 내용 정보와 구조 정보를 검색하는 질의 언어로서 XPointer 의 Syntax 를 기반으로 하여 설계되어져 있다.

그림 4[그림 4] XPQL 의 BNF는 본 논문에서 정의한 XPQL 의 구조를 나타낸 그림이다. XPQL 은 XPointer 의 Syntax 를 거의 따르고 있는데, 현재 XPQL 에서는 XPointer 의 SpanTerm 은 지원을 하지 않는 상태이고, AttrTerm 은 OtherTerm 의 descendant 키워드를 이용하여 대체하고 있다. 또한, String 키워드에서는 XPointer 에서는 Position 과 Length 와 같은 속성을 이용할 수 있었으나 본 논문의 XPQL 에서는 단지 비교 문자열과 "+"나 all 의 인자만으로 지원되게 된다.

```
XPQL ::= XPointer
XPointer ::= AbsTerm '.' OtherTerms | AbsTerm
        | OtherTerms
OtherTerms ::= OtherTerm | OtherTerm '.' OtherTerm
OtherTerm ::= RelTerm | StringTerm
AbsTerm ::= 'root()' | 'origin()' | IdLoc
IdLoc ::= 'id(' Name ')'
RelTerm ::= Keyword? Arguments
KeyWord ::= 'child' | 'descendant' | 'ancestor'
        | 'preceding' | 'following' | 'psibling'
        | 'fsibling'
Arguments ::= '(' InstanceOrAll (',' NodeType
        (',' Attr ',' Val)*)? ')'
InstanceOrAll ::= 'all' | Instance
Instance ::= ('+' | '-')? [1-9] Digit*
NodeType ::= Name | '#element' | '#pi' | '#comment'
        | '#text' | '#cdata' | '#all'
Attr ::= '*' | Name
Val ::= '#IMPLIED' | '*' | Name | SkipLit
StringTerm ::= 'string(' InstanceOrAll '.' SkipLit ')'
```

[그림 4] XPQL 의 BNF

본 XPQL 은 다양한 구조 정보 검색을 지원하고 있는데, XPointer 의 Syntax 를 그대로 사용함으로 문서 내의 엘리먼트 단위의 세세한 부분까지도 검색의 대상으로 지정할 수 있게 되었다. 또한, 검색의 결과도 문서의 특정 부분들을 지정할 수 있게 되었다.

XPQL 은 다양한 정보 검색을 지원하게 되는데 크게 내용 정보 검색과 구조 정보 검색, 속성 정보 검색, 그리고 내용이나 구조, 속성이 복합된 검색이 있게 된다. 다음은 XPQL 에서 제공해주게 되는 다양한 검색의 예들이다.

내용 검색

- '가난한 홍부'라는 구가 있는 엘리먼트를 찾아라.
  XPQL: string(ALL, '가난한 홍부')
- 'play.xml' 이라는 문서에서 "가난한 홍부"라는 구가 있는 엘리먼트를 찾아라.
  XPQL: play.xml#string(ALL, "가난한 홍부")

구조 검색

- ancestor : <TITLE>의 부모 엘리먼트를 찾아라.
  XPQL: TITLE.ancestor(1, #element)
- child : <SPEECH>의 자식 엘리먼트 중 7 번째 엘리먼트를 찾아라.
  XPQL : SPEECH.child(7, #element)
- descendant : 5 번째 <SCENE>을 찾아라.
  XPQL : descendant(5, SCENE)
- preceding : <ACT>보다 두번째 앞에 나오는 엘리먼트를 찾아라.
  XPQL : ACT.preceding(2, #element)
- following : <ACT>보다 2 번째로 뒤에 나오는 엘리먼트를 찾아라.
  XPQL : ACT.following(2, #element)
- psibling : <ACT>보다 이전에 나오는 형제 엘리먼트들을 찾아라.
  XPQL : ACT.psibling(ALL, #element)
- fsibling : <ACT> 이후에 나오는 형제 엘리먼트들을 찾아라.
  XPQL : ACT.fsibling(ALL, #element)
- 키워드가 복합된 질의 : <ACT>의 첫번째 자식 엘리먼트의 두번째 자손을 찾아라.
  XPQL : ACT.child(1, #element).descendant(2, #element)

속성 검색

- '신분'이라는 속성의 값이 "양반"인 엘리먼트를 찾아라.
  XPQL : descendant(ALL, #element, 신분, "양반")
- 속성 'SEX'가 "MALE"이고 속성 '신분'이 "농부"인 <PERSON>을 찾아라.
  XPQL : descendant(ALL, PERSON, GEN, "MALE", 신분, "농부")

복합 검색

- 내용 + 구조: 첫번째 <SCENE>에 '수출'이라는 구가 있는 엘리먼트을 찾아라.
  XPQL: descendant(1, SCENE).string(ALL, '수출')
- 내용 + 속성: 신분이 "양반"인 <PERSON> 중에서 "홍부"라는 구를 가지는 엘리먼트를 찾아라.
  XPQL: descendant(ALL, PERSON, 신분, "양반").
  string(ALL, "홍부")
- 속성 + 구조: 신분이 양반인 <PERSON>의 부모 엘리먼트를 찾아라.
  XPQL: descendant(ALL, PERSON, 신분, "양반").
  ancestor(1, #element)

XPQL 은 XPointer 의 Syntax 를 따름으로 새로운 형태의 질의 언어를 배울 필요 없이 XML 표준에서 제공하는 Syntax 만을 가지고 XML 문서 검색에 필요한 대부분의 정보 검색을 지원하고 있다. 하지만, XPQL 에서도 링크에 대한 질의는 제공되지 않기 때문에 본 논문에서는 새로운 링크 질의 언어를 설계하였다.

### 5.3 XLQL 의 구조

XLQL 은 XML 문서의 링크 정보에 대한 질의 언어로서 본 논문에서 새로운 키워드를 만들어 설계하고 있다.

XLQL 은 링크 검색을 위해 LINK 라는 키워드를 새롭게 사용하였다. 그 외의 XLQL 의 구조는 사용의 유용성을 위해 XPointer 의 syntax 와 유사하도록 설계되어져 있다.

```
XLQL ::= TargetTerm Arguments*
TargetTerm ::= LINK
Arguments ::= "(" Interval ("," Attr "," val)* ")"
Interval ::= InstanceOrAll ("~" InstanceOrAll)?
InstanceOrAll ::= * | / | instance
Instance ::= (+ | -)? [0-9] Digit*
Attr ::= Name
Val ::= NAME | SkipLit
```

[그림 5] XLQL 의 BNF

XLQL 에서는 링크 검색의 범위를 정할 수 있도록 *(forward all), /(backward all) 또는 특정 깊이 또는 구간을 지칭할 수 있도록 하였고, 링크의 어트리뷰트 값에 대한 질의를 가능하도록 하였다.

다음은 XLQL 로 표현할 수 있는 질의들의 다양한 예들이다.

Simple Link Query

- 첫 번째 section 이 링크하고 있는 부분을 찾아라.
  XQL: descendant(1, section).LINK(1)
- 첫 번째 section 을 링크하고 있는 부분을 찾아라.
  XQL: descendant(1, section).LINK(-1)

Transitive Link Query

- 두 번째 P 로부터 링크되어지는 모든 부분을 찾아라.
  XQL: descendant(2, P).LINK(*)
- 두 번째 P 를 참조할 수 있는 모든 부분들 중 3 단계 이하로 링크된 부분을 찾아라.
  XQL: descendant(2, P).LINK(-1 ~ -3)

Link Set Query

- a, b, c 가 동시에 링크하고 있는 것을 찾아라.
  XQL: descendant(1, a).LINK(1) ∪ descendant(1, b).LINK(1) ∪ descendant(1, c).LINK(1)
- [A], [B], [C] 문서가 동시에 링크하는 부분을 찾아라.
  XQL: A#descendant(all,*).LINK(1) ∪ B#descendant (all, *) ∪ C#descendant(all, *).LINK(1)

Link Attribute Query

- XML:LINK : Simple Link 를 찾아라.
  XQL: LINK(1, xml:link, simple)
- INLINE : 'inline' 링크를 찾아라.
  XQL: LINK(1, inline, TRUE)
- ROLE : role 이 'analysis'인 링크를 모두 찾아라.
  XQL: LINK(1, role, analysis)
- TITLE : 링크의 Title 이 "인간관계"인 엘리먼트를 찾아라.
  XQL: LINK(1, title, 인간관계)
- Attribute complex query : 'inline' 링크들 중에서 Title 이 '인간관계'이고, 링크 대상의 Role 이 'reference'인 것을 찾아라.
  XQL: LINK(1, inline, TRUE, title, 인간관계, content-role, reference)

Complex Query

- 두 번째 section 을 링크하면서 깊이가 3 이하이고, 'inline' 링크이며 Title 이 '인간관계'인 것을 찾아라.
  XQL: descendant(2, section).LINK(-1 ~ -3, inline, TRUE, Title, 인간관계)

Link Attribute Query 에 들어가게 되는 속성으로는 위에서 나오는 것 외에 Content-Title, Content-role 에 대해 질의가 가능하나, Actuate 나 Show 는 시스템 의존적인 속성들이기 때문에 질의대상에서 제한다.

이러한 링크에 대한 질의는 기존에 내용이나 엘리먼트의 구조에 대한 질의만 되던 검색에 보다 다양한 구조 검색 기능이 제공할 수 있게 되고, 사용자의 요구로 인한 브라우징으로 해당 문서를 찾아가던 방법에서 링크 검색으로 미리 원하는 링크 정보와 관련 문서들을 손쉽게 찾을 수가 있게 된다. 또한 문서의 편집이 있을 경우에도 링크를 받고 있는 문서가 변경되었을 경우에도 링크 검색을 통해 링크에 대한 정보 변경이나 내용 변경을 보다 손쉽게 할 수 있게 된다.

### 5.4 확장 XQL

앞에서 정의되어진 XPQL 과 XLQL 로 만들어진 XQL 은 여러 문서나 여러 DTD 의 문서에 대한 고려가 되어 있지 않다. 또한, 질의 결과로 얻게 되는 것이 모두 엘리먼트로만 되어 있다. 따라서, 여기서는 앞에서 정의된 XQL 을 확장하여 여러 문서나 여러 DTD 의 문서에 대해 검색을 할 수 있고, 결과도 엘리먼트 뿐만 아니라, 문서 전체에 대해서도 지정할 수 있도록 되어진 XQL 을 제시하고자 한다.

```
[SELECT]
  SELECT Expression
[FROM]
  FROM Expression
[WHERE]
  WHERE Expression
```

[그림 6] 확장 XQL 의 형태

본 확장 XQL 은 [SELECT], [FROM], [WHERE] 와 같은 세가지의 Clause 로 구분되어 지게 된다.

[SELECT] clause

검색의 결과로 받고자 하는 부분을 정의하는 것으로 문서 전체를 결과로 얻기 위해 #document 를, 해당 엘리먼트를 얻기 위한 #element 를, 또한 다시 새로운 부분의 엘리먼트를 얻기 위해 XQL 을 사용할 수 있다.

다음은 play DTD 의 hamlet 이라는 문서를 찾으라는 질의이다.

[select] #document
[from] play:hamlet

또한, 다음의 예는 play DTD 의 hamlet 문서의 루트의 첫번째 엘리먼트를 찾으라는 예제이다.

[select] child(1, #element)
[from] play:hamlet

[FROM] clause

검색의 범위를 정의하는 부분으로 DTD 와 문서의 쌍으로 이루어진다. 만약 모두를 지칭하는 경우에는 "*"를 이용한다.

다음의 검색 범위는 play DTD 의 모든 문서이다.

[from] play:*

또한, 변수를 사용하여 where 절의 조건으로 사용되어 질 수 있다.

[from] play:hamlet $1, talent:vivian $2
[where] $1#child(1, Title) & $2#child(1, Body)

[WHERE] clause

이곳은 실제적인 문서의 검색 질의어가 들어가는 곳이다. Where 절은 XQL 형태의 질의가 들어가는 structure 부분과 string 키워드가 여러 문서에서의 내용 검색에서 기능이 약하기 때문에 색인된 내용 정보에 대한 검색을 위한 키워드 부분으로 구성되어 진다.

다음은 모든 Title 중에서 Venis 라는 키워드를 포함하는 것을 찾으라는 where 절의 예이다.

[where] structure: descendant(all, Title)
keyword: venis

또한, 본 where 절에서는 서로 다른 조건들에 대해 & (and)와 | (or)로 연결을 할 수 있도록 하였다.

[where] structure: descendant(all, title) keyword: venis
& structure: descendant(all, body) keyword: river

확장 XQL 의 다양한 질의의 예

위의 세개의 절을 이용하여 다양한 형태의 질의를 할 수가 있다. 다음은 확장 XQL 의 질의들의 예이다.

- play DTD 중에서 <title>에 "Venis"와 "king"을 포함하고 <body>에 "river"를 포함하고 있는 문서의 act 의 첫번째 scene 를 찾아라.

[select] descendant(all, act).child(1,scene)
[from] play:*
[where] structure: descendant(all, title) keyword : Venis & king & structure: descendant(all, body) keyword : merchant

- play DTD 중에서 <PERSONA>엘리먼트의 'GEN' 속성이 'MALE'이며, <TITLE>엘리먼트에 'merchant'를 포함하는 문서를 찾아라.

[select] #document
[from] play:*
[where] structure: TITLE keyword: merchant &&
structure: descendant(ALL, PERSONA, GEN, MALE)

- 'play DTD중에서 aaa.xml' 문서에서 '김사장'이라는 <PERSONA>와 'bbb.xml'의 '여직원'이라는 <PERSONA>를 동시에 링크하고 있는 부분을 찾아라.

[select] #element
[from] play:aaa.xml $1, play:bbb.xml $2
[where] structure:$1#descendant(ALL, ERSONA).
string(1, "김사장").LINK(-1) & $2#descendant
(ALL,PERSONA).string(1, "여직원").LINK(-1)

## 6. 결론

앞으로는 웹을 기반으로 하여 XML 문서들이 다양하게 출현하게 될 전망이다. 따라서, 이러한 XML 문서를 저장하고 검색하기 위한 다양한 응용들이 나오게 될 것이다. 이러한 응용들에서 중요한 요소가 XML 문서가 지닌 다양한 문서 정보에 대해서 충분히 검색할 수 있는 검색 질의어이다.

본 논문에서는 XML 문서에 대해서 다양한 구조 정보 검색을 지원하는 검색 질의 언어로 XQL 을 제안하였다. XQL 은 XML 스펙의 XPointer 를 기반으로 하여 사용자에게 쉽게 사용할 수 있도록 하고 있고, 기존의 다른 질의에서는 제공이 되지 않고 있던 링크 정보 검색을 지원하기 위하여 새로운 검색 질의 언어를 추가하였다. XQL 은 거의 대부분의 XML 문서 정보 검색 기능을 지니고 있기 때문에 XML 문서 검색기의 가장 적절한 검색 질의 언어라고 할 수 있다.

## 참고문헌


[1]. 류은숙, "구조화된 멀티미디어 문서 모델링에 관한 연구", 박사학위 논문, 충남대학교, 1998.

[2]. Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0", REC-xml-19980210, http://www.w3.org/TR, 1998.

[3]. 이강찬, 이규철, "XML 의 개요", EDI/EC Magazine 가을(15 호), 1998.

[4]. Eve Maler, Steve DeRose, "XML Linking Language (XLink)", WD-xlink-19980303, http://www.w3.org /TR, 1998.

[5]. Eve Maler, Steve DeRose, "XML Pointer Language (XPointer)", WD-xptr-19980303, http://www.w3.org /TR , 1998.

[6]. James Clark, Stephen Deach, Adobe, "Extensible Stylesheet Language (XSL)", WD-xsl-19980818, http://www.w3.org/TR, 1998.

[7]. 김용훈, 연제원, 장동준, 조정수, 이강찬, 이규철, 나중찬, "바다-III 를 기반으로 한 멀티미디어 문서 모델링", 제 12 회 산학연 멀티미디어 기술 학술대회 논문집, 1998.