RDBMS를 이용한 XML DTD 데이터베이스와 확장 SQL의 설계

오준환 °, 이병욱 °
°경원대학교 대학원 전자계산학과

Design of Extended SQL and XML DTD Database using RDBMS

Jun-Hwan Oh °, Byung-Wook Lee °
°Dept of Computer Science, Kyungwon University

요 약

최근 XML문서를 저장 및 검색하기 위한 연구가 활발히 진행되고 있다. 하지만 기존의 연구는 주로 XML문서 저장을 위한 연구들이었다. 즉 XML문서를 정의해주는 DTD 문서의 저장에 관한 연구는 상대적으로 적었다. 하지만 DTD문서를 효율적으로 저장하고 관리하여 재활용함으로써 XML문서를 효율적으로 관리할 수 있다

본 논문에서는 DTD를 RDBMS에 저장하기 위한 스키마를 제안하고, 저장 방식은 정보의 중복 저장을 막고 DTD의 모든 내용을 수용할 수 있도록 설계하였다. 또 제안하는 데이터모델에 적용하여 SQL의 DDL을 확장하였다. 제안한 DTD 데이터베이스로 인해 DTD의 재활용과 관리를 할 수 있게되었고 SQL의 확장으로 제안 시스템의 사용을 용의 하게 하였다.

1. 서론

최근 인터넷의 발전으로 정보의 표현이 HTML 만으로는 부족하여 새로운 표현 방법이 1997년 W3C(World Wide Web Consortium)에서 제시되었다. 제시된 구조 정보표현 방법은 HTML(Hyper Text Markup Language)의 편리성과 SGML(Standardized Generalized Markup Language)의 확장성을 조합한 형태를 가지고 있는 XML(eXtensible Markup Language)이다[1][2][4]. XML을 이용한 정보표현이 많아짐으로 인해 구조적 문서인 XML 문서를 데이터베이스에 저장 관리하는 연구가 이루어지고 있다. 객체지향 데이터베이스로의 저장과 관계형 데이터베이스로의 저장으로 나누어진다. 저장방식으로의 분류는 가상 분할 방식과 분할 방식으로 나누어진다.

위에서 알아본 저장에 관한 연구는 XML 실제 문서 저장에 관한 연구이고 DTD 저장에 관한 부분은 아주 간소하여 DTD를 분할하지 않고 저장하거나 엘리먼트와 에트리뷰트를 나누어서 저장하는 방식이다[3][5][6]. 또 저장하는 방식만을 제안했을 뿐 사용자들의 위해 SQL를 확장하지 않아 실제 시스템을 사용할 때 많은 어려움을 주었다.

본 논문에서는 XML문서 저장의 연구 중에 DTD의 저장 방법을 개선하여 DTD의 정보를 모두 담을 수 있는 데이터모델을 설계하고 SQL을 확장하여 설계한 데이터모델을 사용하기 쉽게 하여 DTD 데이터베이스 시스템을 설계한다. 제2장에서는 DTD 저장을 위한 저장 스키마를 제안하고 제3장에서는 SQL의 확장을 제안하고 제4장에서는 결론 및 향후 연구 방향을 제시한다.

2. DTD 저장을 위한 스키마

제안한 데이터 모델은 [5]에서 제안한 모델을 사용하였고 XLINK에 대한 지원은 사용하지 않았다. 제안된 테이블은 모두 8개이다

데이터 모델 스키마의 테이블은 tbDTD_info, tbnonMixELEMENT, tbMixELEMENT, tbSUB_ELEMENT,

tbELEMENT_content, tbATTLIST, tbATT_enumeration, tbENTITY로 구성되어 있다. 다음 그림은 위에서 지정한 테이블의 구성을 나타낸다.

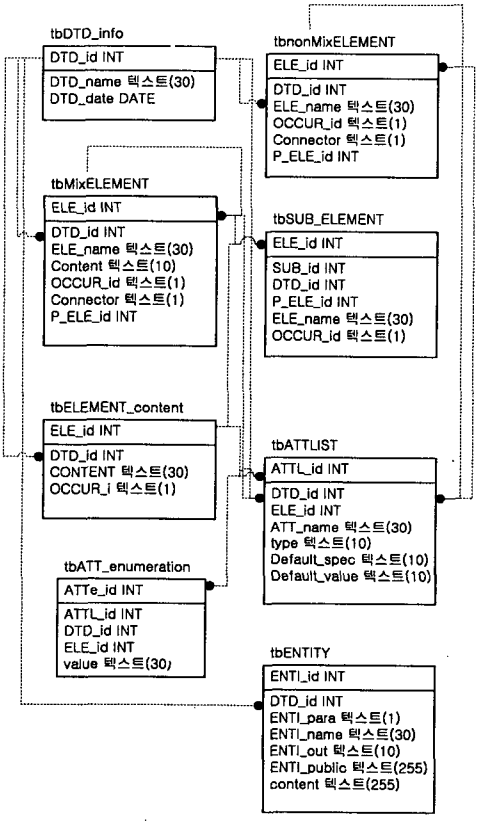

그림 2.1 DTD 데이터 모델의 ERD

3. SQL의 확장

기존의 XML문서를 위한 SQL의 확장은 SQL의 문법과 약간 다르고 키워드를 중복해서 사용하는 단점이 있다[7]. 본 논문에서는 기존의 확장을 보완하여 제안하는 데이터 모델에 적용할 수 있게 하였다.

3.1 SQL의 DDL의 확장

DDL의 확장 문은 CREATE을 확장한 CREATE DTD과 ALTER, DROP이다. 다음 그림은 확장된 CREATE 문의 BNF를 나타낸다. CREATE문중 DTD에 해당하는 부분만을 표현한 것이다.

```
CREATE DTD <DTD name> '('
{ <Element expression> | <Entity expression> } | <Attlist expressi
on> [,...] } | *
')'

<Element expression> ::= <ELEMENT name> [ AS ELEMENT { <
NonMix Model> | <Mix Model> | <Content> }

<NonMix Model> ::= { <choice> | <seq> } { '?' | '*' | '+' }?
<choice> ::= '(' <space>? <cp> ( <space>? '|' <space>? <cp> )*
<space>? ')'
<seq> ::= '(' <space>? <cp> ( <space>? ',' <space>? <cp> )* <s
pace>? ')'

<Mix Model> ::= { <Mix choice> | <Mixseq> } { '?' | '*' | '+' }?
<Mix choice> ::= '(' <space>? <Content> ( <space>? '|' <space>?
<cp> )* <space>? ')'
<Mix seq> ::= '(' <space>? <Content> ( <space>? ',' <space>? <
cp> )* <space>? ')'

<cp> ::= ( <Element name> | <choice> | <seq> ( '?' | '+' | '*' )?
<Content> ::= '#PCDATA'

<Attlist expression> ::= <ATTLIST name> <Attist Type> [ <default
> ]
<Attlist Type> ::= <Enumerate Type> | ( <ENTITY name> )+ | <
Cdata> | <xml Extend type> | <Tokenized type>
<default> ::= { NOT NULL | NULL | DEFAULT <string> }
<Enumerate Type> ::= <Enumerate> | <Notation>
<Enumerate> ::= '(' <string> ( '|' <string>)* ')' DEFAULT <string>
<Notation> ::= 'NOTATION' <space> '(' <string> ( '|' <string> )* ')'
DEFAULT <string>
<Cdata> ::= 'CDATA'
<Tokenized type> ::= 'ID' | 'IDREF' | ' IDREFS' | 'ENTITY' | 'ENT
ITIES' | 'NMTOKEN' | 'NMTOKENS'

<Entity expression> ::= <ENTITY name> '(' <Entity Definition> ')'
AS ENTITY

<Entity Definition> ::= <stinrg>
<ENTITY name> ::= <stinrg>
<Element name> ::= <stinrg>
<ATTLIST name> ::= <stinrg>
<stinrg> ::= String
```

그림 3.1 CREATE DTD의 BNF

위에서 정의한 CREATE DTD의 BNF는 #REQUIRES, #IMPLIED, #FIXED를 NOT NULL, NULL, DEFAULT로 변경하여 기존의 SQL에서의 의

미와 비슷하게 하였다.

다음 그림 3.2는 DTD를 삭제 할 때 사용하는 DROP DTD에 대한 BNF이다. DTD를 삭제 할 때 사용하는 것으로써 DTD 자체를 삭제할 때만 사용되어 지고 엘리먼트, 에트리뷰트, 엔티티를 삭제 할 때는 사용할 수 없다. 이들을 삭제 할 때는 ALTER DTD를 이용한다.

```
DROP DTD <DTD name>
<DTD name> ::= String
```

그림 3.2 DROP DTD의 BNF

다음 그림 3.3은 DTD의 내용을 변경하기 할 때 사용하는 ALTER DTD의 BNF이다. CREATE DTD에서 사용하는 엘리먼트, 엔티티, 어트리뷰트의 BNF를 그대로 사용을 하고 뒤에 각각의 변경 내용(ADD, DROP, UPDATE)을 표시해준다.

```
'ATLER DTD' <DTD name> { <New DTD name> | '('
<ELEMENT> | <ATTLIST> | <ENTITY>
')' }

<ELEMENT> ::= ( <ELEMENT name> 'DROP' ) | ( <Element
expression> 'UPDATE' ) | ( <Element expression> 'ADD' )
<ATTLIST> ::= ( <ATTLIST name> 'DROP' ) | ( <Attlist
expression> 'UPDATE' ) | ( <Attlist expression> 'ADD' )
<ENTITY> ::= ( <ENTITY name> 'DROP' ) | ( <Entity expression>
'UPDATE' ) | ( <Entity expression> 'ADD' )
```

그림 3.3 ALTER DTD의 BNF

위의 3가지 DDL를 확장함에 있어서 문법은 기존의 CREATE, DROP, ALTER와 유사하게 제안하여 기존의 SQL을 사용하던 사용자들도 아주 쉽게 적용할 수 있게 하였다. 또 XML의 BNF를 참고하여 제안하였기 때문에 XML 사용자도 적용하기 쉽게 하였다.

3.2 확장 SQL과 스키마 적용 방법

제안하고 있는 확장 SQL을 제안하는 스키마에 적용을 하려면 확장된 SQL을 기존의 SQL로 번역을 해주어야한다.

CREATE DTD의 경우는 INSERT 문을 이용하여 각각에 해당하는 스키마에 인스턴스 값들을 저장한다. DROP의 경우는 DELETE문을 이용하여 해당하는 인스턴스 값들을 지운다. 또 ALTER의 경우는 UPDATE 문은 이용하여 인스턴스의 변경 내용을 변경한다.

4. 결론 및 향후 연구 방향

문서를 정의하는 DTD를 저장관리 하면 DTD를 재활용할 수 있다. XML문서를 만드는 것보다 문서 정의부인 DTD를 만드는 것은 어려운 작업이다. 이렇게 재활용을 한다면 DTD를 만드는데 시간을 낭비할 필요가 없어지게 된다.

본 논문에서는 DTD를 RDBMS에 저장하기 위하여 스키마를 제안하였다. 이 스키마는 DTD를 분할해서 저장하여 수정 삭제를 쉽게 하였다 제안 스키마는 DTD의 정보를 손실 없이 저장할 수 있다. 또 이 스키마를 사용하기 위해 SQL를 확장하였다. 확장된 SQL은 DTD를 생성하고 삭제하고 수정하고 추가할 때 사용이용의 하게 하였다. 확장된 질의는 기존의 SQL 문법과 유사하게 확장하여 기존의 SQL사용자들도 쉽게 접근할 수 있게 하였다.

제안한 DTD 시스템을 이용한 XML 문서 저장 시스템과 그에 적용되는 SQL을 확장하면 기존의 RDBMS를 사용하였던 사람들도 XML 저장시스템을 쉽게 접근할 수 있을 것이다.

[참고 문헌]

[1] ISO 8879, "Standard Generalized Markup Language(SGML)", Generva Sritzerland, 1986

[2] eXtensible Markup Language(XML),"http://www.w3.org/TR/PR-xml-971208," 1997

[3] Susan Malaika, "Using XML in Relational Database Application," ICDE99, pp167, 1999

[4] Didier martin, 외12명, "Professional XML," Wrox Press, 2000

[5] 오준환, 이병욱, "XLink를 지원하는 XML DTD 데이터 모델 설계," 인터넷정보학회춘계학술대회, 2000

[6] 빈진영, "구조 기반 검색을 지원하는 XML DTD 데이터베이스의 설계 및 구현," 단국대학교 석사논문, 1999

[7] 이찬구, "XML 문서와 데이터베이스 통합을 위한 SQL의 확장," 정보과학회춘계학술대회 논문지, 2000

[8] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, "Compilers - Principles, Techniques and Tools," 1988

[9] Serge Abiteboul, Peter Buneman, Dan Suciu, "Data On the Web from Relations to Semistructuer Data and XML," Moran Kaufmann Publishers, 2000