# XML 데이터 관리 시스템의 자동 생성을 위한 XMLStoDBS

박종현°, 이한수, 강지훈
충남대학교 컴퓨터과학과
{ jhpark°, hansu, jhkang }@cs.cnu.ac.kr

# The XMLStoDBS for Automatic Generation of XML Data Management System

Jong-Hyun Park°, Han-Su Lee & Ji-Hoon Kang
Dept. of Computer Science, Chungnam National University

**요 약**

XML[1]이 인터넷 상의 메시지 교환 형식으로 활발히 이용되면서, XML을 이용하는 많은 응용에서는 이를 효율적으로 관리하기 위한 방법이 요구되었다. 이러한 요구에 발 맞추어 현재, XML 데이터의 관리를 위한 많은 연구[3, 4, 5, 6, 7, 8, 9]가 진행 중에 있으나 아직까지는 어떤 방법이 XML 데이터의 관리를 위해 최적의 방법이라는 결론은 없는 실정이다. 또한, 이러한 방법들을 실제 응용에 적용하기 위해서는 시스템 관리자가 XML 문서의 특성을 파악하고, 이를 기반으로 XML 데이터의 관리를 위해서 필요한 모든 모듈들을 개별적으로 개발하고 관리해야 한다. 이와 같은 방법은 응용의 측면에서 추가적인 비용과 노력을 추가해야 하는 부담이 발생한다.

본 논문에서는 앞서 언급한 요구사항들을 해결하기 위하여 고려하여 XML Schema to Relational Database Schema (XMLStoDBS)라는 XML문서를 제안하고, 이를 이용하여 응용에서 필요한 XML 데이터 관리 시스템을 자동으로 생성하기 위한 방법을 제안한다. XMLStoDBS는 응용에서 사용하고자 하는 XML Schema또는 DTD의 정보와 실제 데이터가 저장될 데이터베이스의 정보, 이 둘간의 사상관계를 표현하는 문서이다. 우리의 XMLStoDBS는 사용자가 응용에서 사용하고자 하는 XML Schema/DTD의 입력만으로 자동으로 생성되며, 응용에서는 이를 이용하여 응용에 필요한 XML 데이터 관리 시스템을 자동으로 생성할 수 있다.

## 1. 서론

인터넷 상의 메시지 교환을 위한 표준으로 자리 잡은 XML은 현재 많은 응용에서 사용되고 있으며, 응용에서 사용하는 XML 문서의 양 또한 방대하다. 이러한 XML 데이터를 효율적으로 관리하기 위하여 많은 응용에서는 여러 가지 해법을 찾고 있으나[3, 4, 5, 6, 7, 8, 9], 아직까지는 어떤 방법이 가장 효율적인 방법이라는 결론은 없는 실정이다. 또한 이미 제안된 많은 방법들을 실제 응용에 적용하기 위해서는 어떤 방법이 응용에서 사용하는 XML 문서의 특성을 잘 반영하고 있는지를 파악하여 시스템 개발자가 직접 응용에서 XML 문서를 관리하기 위한 모듈들을 개발하여야 한다. 그러나, 이러한 방법들은 많은 노력과 비용을 필요로 하며, 개발한다 하더라도 현실적으로 효율적인 관리는 힘들다.

본 논문에서는 이러한 현실을 고려하여 XMLStoDBS라는 XML 문서를 제안한다. XMLStoDBS는 XML Schema/DTD의 정보, XML 데이터가 물리적으로 저장될 데이터베이스의 정보와 이 둘간의 사상관계를 표현하고 있는 XML 문서로써, 이를 이용하여 응용에서 XML 데이터의 관리를 위해서 필요로 하는 모든 정보를 기술할 수 있다. XMLStoDBS에 기술되는 정보들은 XML 데이터의 저장과 검색을 위해, 기존에 제안된 방법들 중 검증된 방법들을 기준으로 필요한 정보들을 기술하고 있으므로 XML 데이터를 사용하는 어떠한 응용에서도 적절하게 사용이 가능하다. 또한 XMLStoDBS는 XML 데이터 관리 시스템 자동 생성기뿐만 아니라 사람이 읽기 쉽도록 정의하였으므로 응용의 필요에 의해서 확장이 가능하다는 장점을 갖는다.

본 논문의 구성은 다음과 같다. 2절에는 XML문서의 관리를 위한 기존 연구들에 대해 기술하고 있으며, 3절에서는 XMLStoDBS에 관하여 구체적으로 설명하고 있다. 4절에서는 XMLStoDBS를 이용하는 XML 데이터 관리 시스템 자동 생성기에 대하여 기술하고 있으며, 마지막으로 4절에서 결론과 함께 향후 계획을 기술하고 있다.

## 2. 관련연구

많은 응용에서는 현재 대량의 XML 문서를 효율적으로 관리하기 위한 방법으로 여러 가지 다양한 장점을 포함하는 관계형 데이터베이스를 이용하고 있다[9]. 그러나 XML 문서의 특성상 구조적인 XML 문서를 평면 구조의 관계형 데이터베이스에서 관리하기 위해서는 많은 문제가 존재한다. 그러므로 현재 많은 연구 기관에서 이를 해결하기 위한 연구가 진행 중에 있으나[3, 4, 5, 6, 7, 8, 9], 아직까지는 어떤 방법이 가장 효율적인 방법이라는 결론은 나지 않은 상태이다.

응용의 측면에서 XML 문서 관리의 문제는 크게 저장과

* 이 연구는 BK21 충남대학교 정보통신 인력 양성 사업단과 소프트웨어 연구 센터의 지원을 받았음.

검색의 문제로 나뉜다. 그러나 결국 XML 문서의 저장과 검색의 문제는 상호 의존적인 모습을 보인다. 즉, XML 문서를 효율적으로 저장하기 위해서는 데이터의 중복 저장을 최소화하기 위해서 여러 개의 관계형 테이블에 나누어 저장해야 하지만, 검색과 문서의 재조합을 위해서는 최대한 관계형 테이블의 개수를 줄여야 한다.

현재, 국내,외에서 앞서 언급한 문제를 해결하기 위한 대표적인 방법으로는 XML 문서를 Edge 단위로 저장하는 방법[3], XPath 테이블을 사용하는 방법[8]과 Inlining 방법[6]등이 있다. Edge 방법은 XML문서에서 두 Element Node 사이를 연결하기 위한 Edge를 단위로 관계형 테이블을 정의하여 저장하는 방법으로, 가장 편리한 방법이지만 구조의 검색과 문서의 재조합을 위해서는 많은 테이블 Join이 발생한다는 단점이 존재한다. 이러한 단점을 해결하기 위한 방법으로 XPath 테이블 방법이 있다. XPath 테이블 방법은 XPath 테이블에 각 Element Node에 대한 경로를 표현하여 해당 Node에 직접 접근이 가능하도록 하는 방법으로, XML 문서의 구조 검색 시 불필요한 Join을 줄이는 방법이다. 그러나 이 방법 또한 문서의 재조합을 위해서는 Join이 필요하다는 단점을 제거하지는 못했다. Inlining 방법은 XML DTD또는 Schema를 분석하여 상위 Node에 해당하는 테이블에 Inlining하는 방법이다. 이때 몇 가지 Inlining을 위한 규칙을 정의하여 규칙에 만족할 경우 Inlining이 가능하다. 이 방법의 단점은 관리자가 DTD 또는 schema를 분석해야 한다는 단점이 존재하며, 최악의 경우 Inlining 노드가 하나도 존재하지 않는다면 Edge방법과 동일하게 적용될 수 있다.

본 논문에서 제안하고 있는 XMLStoDBS는 위에 제안된 모든 방법들을 모두 만족시킬 수 있도록 정의되어 있다. 그러나 본 논문에서는 위 방법들의 단점을 제거하고 장점만을 취합하는 방법을 사용하기 위하여 [10]에 제안된 방법을 기반으로 XMLStoDBS작성하고 사용한다.

## 4. XMLtoDBS

XMLStoDBS는 XML로 기술되며, [그림 1]과 같은 구조의 DTD를 기반으로 기술 된다.

```
1:  <!ELEMENT XMLSTODBS (ELECON|TABLECON|ATTCON)*>
2:  <!ELEMENT ELECON (ATTCON | ELECON | PARENT)*>
3:  <!ATTLIST ELECON
4:          ELENAME CDATA #REQUIRED
5:          MAPTABLE IDREF #IMPLIED
6:          MAPDBATT IDREF #IMPLIED
7:          WILDCARD CDATA #IMPLIED>
8:  <!ELEMENT ATTCON (#PCDATA)>
9:  <!ATTLIST ATTCON
10:         ATTNAME CDATA #REQUIRED
11:         ATTID IDREF #REQUIRED
12:         MAPTABLE IDREF #IMPLIED
13:         MAPFIELD IDREF #IMPLIED>
14: <!ELEMENT TABLECON (DBATT)*>
15: <!ATTLIST TABLECON
16:         TABLEID ID #REQUIRED
17:         TABLENAME CDATA #REQUIRED>
18: <!ELEMENT DBATT (#PCDATA)>
19: <!ATTLIST DBATT
20:         DBATTNAME CDATA #REQUIRED
21:         DBATTID ID #REQUIRED
22:         DBATTCTR CDATA (default|element|attribute) 'default'
23:         DBATTTYPE CDATA (varchar|int|CLOB) 'varchar'
24:         DBATTSIZE CDATA #IMPLIED
25:         DBTABLELINK IDREF #IMPLIED
26:         DBATTLINK IDREF #IMPLIED
27:         DBINDEX CDATA (true|false) 'false'>
28: <!ELEMENT PARENT (#PCDATA)>
```

[그림 1] XMLStoDBS를 위한 DTD

[그림 1]에 기술된 DTD는 XMLStoDBS를 위한 최소한의 DTD이다. XMLStoDBS는 XML 데이터 관리 시스템 자동 생성기뿐만 아니라 사용자에게 응용에서 사용되는 XML 문서의 정보와 XML 문서가 데이터베이스에 어떻게 저장되었는지를 쉽게 확인할 수 있도록 하는 기능을 제공한다. 만약 사용자가 추가적인 정보를 기술하기를 원한다면 새로운 DTD를 추가로 정의하여 확장할 수 있다. XMLStoDBS의 Element는 크게 ELECON, TABLECON, ATTCON로 구성된다. ELECON는 응용에서 사용하고자 하는 XML Schema/DTD에 정의된 Element의 정보를 기술하는 부분으로, 속성에는 데이터베이스에 사상되는 테이블과 필드의 정보가 기술된다. ELECON은 자식 Element로 ELECON, ATTCON과 PARENT Element를 취한다. ATTCON은 해당 Element의 Attribute 노드의 정보를 기술하는 부분으로 Attribute의 이름, 식별자, 데이터베이스에 사상될 테이블의 이름과 필드의 이름 등이 기술된다. PARENT Element에는 해당 Element 노드의 부모가 될 수 있는 노드를 모두 기술한다. PARENT 노드는 XQuery 엔진과 XQuery 사용자 인터페이스의 생성시 XPath 표현의 처리에 유용하게 사용된다. TABLECON은 데이터베이스의 테이블 정보를 기술하는 Element로 DBATT를 자식 Element로 이용하여 데이터베이스 테이블과 필드들을 기술한다. DBATT에는 해당 필드의 이름과 식별자를 기술하며, 해당필드가 XML Schema의 어떤 타입의 노드와 사상되는가, 해당 필드의 데이터 타입, 크기, 다른 테이블과의 연관 관계, Index여부 등을 기술 한다.

```
1: <XMLSTODBS>
2:   <ELECON ELENAME="PLAY" MAPTABLE="PLAY1" WILDCARD="ONE">
3:     <ATTCON ATTNAME="id" ATTID="PLAY_ATT3" MAPFIELD="ID"/>
4:     <ATTCON ATTNAME="abstract" ATTID="PLAY_ATT4" MAPFIELD="ABSTRACT"/>
5:     <ELECON ELENAME="SCNDESCR" ATTID="PLAY_CHILD1" MAPFIELD="SCNDESCR"
         WILDCARD="ONE">
6:       <PARENT>PLAY</PARENT>
7:     </ELECON>
8:     <ELECON ELENAME="PERSONAE" ATTID="PLAY_CHILD2" MAPFIELD="PERSONAE"
         WILDCARD="ONE">
9:       <PARENT>PLAY</PARENT>
10:    </ELECON>
11:    <PARENT>/< PARENT >
       ....
12:  </ELECON>
13: <TABLECON TABLEID="PLAY1" TABLENAME="PLAY">
14:    <DBATT DBATTNAME="ID_PLAY" DBATTID="PLAY_ATT1" DBATTCTR="default"
       DBATTTYPE="varchar" DBATTSIZE="100" DBINDEX="true"/>
15:    <DBATT DBATTNAME="ID_PATH" DBATTID="PLAY_ATT2" DBATTCTR="default"
       DBATTTYPE="varchar" DBATTSIZE="200" DBINDEX="true"/>
16:    <DBATT DBATTNAME="PLAY" DBATTID="PLAY_VALUE" DBATTCTR="content"
       DBATTTYPE="varchar" DBATTSIZE="200" DBINDEX="true"/>
17:    <DBATT DBATTNAME="id" DBATTID="PLAY_ATT3" DBATTCTR="attribute"
       DBATTTYPE="int" DBATTSIZE="200" DBINDEX="false"/>
18:    <DBATT DBATTNAME="abstract" DBATTID="PLAY_ATT4" DBATTCTR="attribute"
       DBATTTYPE="varchar" DBATTSIZE="200" DBINDEX="false"/>
19:    <DBATT DBATTNAME="SCNDESCR" DBATTID="PLAY_CHILD1"
       DBATTCTR="element" DBATTTYPE="varchar" DBATTSIZE="200" DBINDEX="false"/>
20:    <DBATT DBATTNAME="PERSONAE" DBATTID="PLAY_CHILD2"
       DBATTCTR="element" DBATTTYPE="varchar" DBATTSIZE="1" DBINDEX="false"/>
       ....
21: </TABLECON>
22: <ELECON ELENAME="PM" MAPTABLE="PM1" WILDCARD="ONEORMORE">
23:    < PARENT >PLAY</ PARENT >
24: </ELECON>
25: <TABLECON TABLEID="PM1" TABLENAME="PM">
26:    <DBATT DBATTNAME="ID_PM" DBATTID="PM_ATT1" DBATTCTR="default"
       DBATTTYPE="varchar" DBATTSIZE="100" DBINDEX="true"/>
27:    <DBATT DBATTNAME="ID_PATH" DBATTID="PM_ATT2" DBATTCTR="default"
       DBATTTYPE="varchar" DBATTSIZE="200" DBINDEX="true"/>
28: </TABLECON> ..... </XMLSTODBS>
```

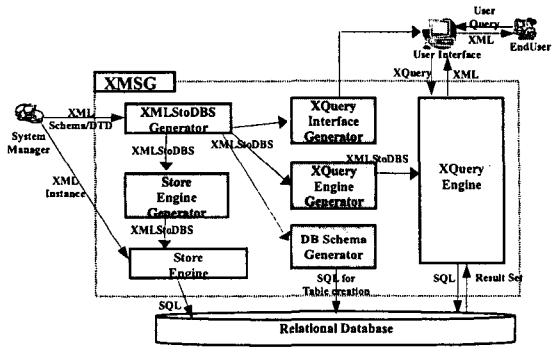
[그림 2] Shakespeare DTD 를 위한 XMLStoDBS 문서

[그림 2]는 XMLStoDBS DTD를 기반으로 Shakespeare DTD[12]를 기술한 XMLStoDBS 문서의 일부분이다. [그림 2]의 문서는 [10]에서 제안한 방법을 기반으로 작성한 문서이다. ELECON Element(선 2~12)는 Shakespeare DTD의 상위 세 개(PLAY, SCNDESCR, PERSONAE)의 Element 노드와 노드의 속성(Attribute Node)에 관한 정보를 기술한다. PLAY Element는 Shakespeare DTD의 최상위 노드로 MAPTABLE이라는 속성을 이용하여 데이터베이스 테이블과의 사상관계를 표현한다. 선 3과4는 PLAY Element의 속성을 기술하고 있으며, 선 5~10은 PLAY Element의 자식

Element중 PLAY에 Inlining되는 Element의 정보를 기술하고 있다. TABLECON Element(선 13~21)는 데이터베이스 테이블의 구조를 기술하고 있으며 이들 각각은 ELECON Element에 기술된 정보들과 사상된다. DBATT는 테이블의 필드를 기술하는 부분으로, 해당 Element의 식별자와 Path를 기술하기 위하여 기본적으로 2개의 필드(선14~15, 선 26~27)를 생성한다. 또한 테이블에 해당하는 Element가 내용을 갖는지(선16), 속성을 갖는지(선17~18), 자식 Element가 Inlining되는지(선 19~20)의 여부에 따라 추가적인 필드가 생성된다. 선 21~24는 Shakespeare DTD중 PM Element를 기술하고 있으며, 선 25~28은 PM Element와 사상되는 데이터베이스 테이블의 정보를 기술하고 있다.

## 3. XML 데이터 관리 시스템 자동 생성기

[그림 3]은 XMLStoDBS를 이용하여 응용에서 필요한 모듈을 자동으로 생성하는 XML data Management System Generator (XMSG)의 기본 구조이다. XMSG의 구성은 크게 XMLStoDBS Generator와 4개의 모듈(DB Schema Generator, Store Engine Generator, XQuery Interface Generator, XQuery Engine Generator)로 구성된다.




[그림 3] XML 데이터 관리 시스템 자동 생성기

[그림 3]에서 XMLStoDBS Generator는 시스템 관리자로부터 XML Schema를 입력 받아 다른 모듈들에서 사용해야 할 XMLStoDBS 문서를 생성하고 각 모듈로 XMLStoDBS를 제공하는 역할을 담당한다. DB Schema Generator는 XMLStoDBS를 입력으로 받아 관계형 데이터베이스 테이블을 생성한다. Store Engine Generator는 관계형 테이블의 저장 구조에 맞게 XML 문서를 저장하는 저장 엔진을 생성하는 역할을 담당한다. 저장 엔진과 XML 데이터가 저장되는 관계형 테이블의 자세한 구조는 [10]에 자세히 기술되어 있다. XQuery Interface Generator는 XMLStoDBS를 기반으로 사용자에게 XQuery를 손쉽게 생성할 수 있는 XQuery 사용자 인터페이스를 생성하는 역할을 담당한다. XQuery[2]는 W3C에서 제정중인 XML 데이터 검색을 위한 표준 질의어로 상호 운용성(interoperability)을 보장받음과 동시에 여러 검색어의 장점을 취합하고 있다. 그러나 XQuery는 일반 사용자가 사용하기는 복잡하다는 단점이 존재한다. 이러한 이유에서 본 논문에서는 XMLStoDBS를 이용하여 XQuery 사용자 인터페이스를 자동으로 생성함으로써 응용을 사용하고자 하는 일반 사용자에게보다 쉽게 원하는 XQuery를 생성할 수 있도록 한다. XQuery Engine Generator는 사용자가 질의한 XQuery를 처리하기 위한 XQuery 엔진을 자동으로 생성하는 역할을 담당한다. XQuery 엔진은 XMLStoDBS를 이용하여 사용자가 질의한 XQuery를 분석하여 실제 XML 문서가 저장된 관계형 데이터베이스에 질의하기 위한 SQL로 변환하는 기능을 수행한다. 또한 검색 결과를 취합하여 사용자가 원하는 형태의 XML문서로 재구성하는 역할을 한다. 이때 XQuery 엔진은 XMLStoDBS를 이용하여 XML 문서의 어떤 정보가 데이터베이스의 어느 부분에 저장되었는지 신속히 확인할 수 있으며, 문서의 재조합 에도 사용된다. XQuery 엔진의 자세한 구조와 검색 방법은 [11]에 자세히 기술되어 있다.

## 4. 결론

우리는 급속히 증가하는 XML 응용들을 위하여 응용에서 사용하고자 하는 XML 데이터와 이를 관리하기 위해서 필요한 정보들을 기술하기 위한 XMLStoDBS를 제안하였다. 이는 XMSG와 같은 응용에서 매우 효과적으로 사용할 수 있을 뿐만 아니라 사용자의 필요에 따라 확장이 가능하므로, 특화된 XML 문서를 사용하는 응용으로의 확장도 가능할 것으로 기대된다. 또한 본 논문에서 제안하고 있는 XMSG는 각각 개별적인 모듈로 설계되었으므로 응용에서 필요에 따라 모듈들을 선택적으로 사용할 수 있도록 하였다. 이러한 특징은 응용에서 사용하는 모듈들이 보다 유연성 있고 확장이 가능하도록 한다.

## 5. 참고문헌

[1] W3C, Extensible Markup Language (XML) Version 1.0, Recommendation, February1998.
[2] W3C, XQuery 1.0: An XML Query Language, Working Draft, November, 2002.
[3] A. Schmidt, M. Kersten, M. Windhouwer, & F. Waas, "Efficient Relational Storage and Retrieval of XML Documents," Proc. WEBDB 2000, 2000.
[4] D. Florescu & D. Kossmann, "Storing and Querying XML Data Using an RDBMS," IEEE Data Engineering Bulletin, Vol. 22 , No. 3 , 1999.
[5] I. Tatarinov, S.D.Viglas, K.Beyer, J.Shanmugasundaram, E.Shekita, & C.Zhang, "Storing and Querying Ordered XML Using a Relational Database System," Proc. ACM SIGMOD Conf., 2002.
[6] J.Shanmugasundaram, K.Tufte, G.He, C.Zhang, D.DeWitt, & J.Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities," Proc. 25th VLDB, 1999.
[7] J. Shanmugasundaram, "XPERANTO:Bridging Relational Technology and XML", IBM Research Report, 2001.
[8] M.Yoshikawa, T.Amagasa, T.Shimura, & S.Uemura: "XRel: a path-based approach to storage and retrieval of XML documents using relational databases," Proc. ACM Transactions on Internet Technology, Volume 5, 2001.
[9] S. Amer-Yahia & D. Srivastava "A mapping schema and interface for XML stores," Proc. WIDM 2002, 2002.
[10] 이용희, 박종현, 이민우, 정민옥, 강지훈 "TV-Anytime 메타데이터의 저장방법 비교 분석", 한국정보과학회 춘계학술발표논문집 제 31 권(1 호), pp.22-24, 대전KAIST, 2004년 4월.
[11] 이민우, 박종현, 이용희, 정민옥, 강지훈, "TV-Anytime기반 메타데이터 관리 시스템", 한국정보과학회 춘계학술발표논문집 제31권 (1호) pp.442-444, 대전KAIST , 2004년 4월.
[12] Shakespeare DTD by Susan Kelsch (www.kelschindexing.com/shakesDTD.html)