객체관계 데이터베이스를 이용한 XML 문서 저장 모델 설계*

김훈°, 홍의경
서울시립대학교 전산통계학과 데이터베이스 연구실
{hkim92, ekhong}@venus.uos.ac.kr

Design of an XML Document Storage Model using Object-Relational Database

Hoon Kim°, Eui Kyeoung Hong
Department of Computer Science and Statistics, University of Seoul

요 약

최근 인터넷의 발전으로 인하여 정보교환을 위한 표준으로 XML에 대한 활용이 늘어나면서 이들 문서에 대한 저장 및 검색에 대한 연구가 활발히 진행되고 있다. 본 연구에서는 객체관계 데이터베이스의 모델링 기법을 이용하여 XML 문서 저장 시스템의 기본이 되는 효과적인 모델을 제시한다. 관계 데이터베이스와 객체지향 데이터베이스의 장단점을 수용한 객체관계 데이터베이스를 이용하여 XML 문서 저장 모델을 설계하면 XML 문서의 구조적인 정보를 효과적으로 표현할 수 있도록 모델 설계가 가능하다. XML 문서의 빈번한 수정이 용이하도록 분할저장 방식을 사용하였고, DTD에 관계없이 XML 문서를 저장할 수 있도록 DTD 독립적인 모델을 제시하였다.

1.서론

최근 WWW의 발전으로 인하여 인터넷의 사용과 정보의 양이 급증하고 있다. 이에 따라 인터넷 상의 정보를 효과적으로 사용하고자 하는 연구가 현재 계속 진행되고 있으며, 구조화된 문서의 전송 및 교환을 용이하도록 하는 표준이 필요하게 되었다. 구조화된 문서를 표현하기 위한 표준인 SGML이 너무나 복잡하다는 단점을 가지고 있고 인터넷 상의 정보를 표현하기 위한 HTML이 구조화된 문서를 표현하기에는 부족하다는 단점을 가지고 있음에 따라 SGML과 HTML의 단점을 보완하여 1998년 W3C(World Wide Web Consortium)에서 차세대 웹 문서의 표준으로 XML(eXtensible Markup Language)을 제안하였다[8]. 현재 인터넷 Web 문서 뿐만 아니라 전자도서관, 전자상거래, EC/EDI를 포함한 다양한 분야에서 XML을 활용하고자 폭넓은 연구를 하고 있으며, 이러한 XML 문서들을 효과적으로 저장하고 검색할 수 있는 XML 문서 저장 검색 시스템을 필요로 하고 있다. 이러한 시스템을 위하여 XML 문서의 구조정보와 내용정보를 효과적으로 검색할 수 있도록 데이터 모델을 설계하는 것이 중요하다.

본 논문에서는 객체관계 데이터베이스의 모델링 기법을 이용하여 XML 문서를 저장하고 XML 문서의 구조정보와 내용정보를 효과적으로 검색할 수 있도록 데이터 모델을 설계하고, 차후 이 모델에 따라 상용 ORDBMS를 이용하여 XML 문서 저장 검색시스템을 구현할 예정이다.

* 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음

2. 관련연구

XML 문서를 관계 데이터베이스를 이용하여 저장할 경우 관계 데이터베이스의 우수한 성능을 이용할 수 있고, 기존의 응용시스템의 데이터를 함께 사용할 수 있는 장점을 가진다. 그러나 데이터 형 선언의 제약, set-value를 지원하지 않는 등의 이유로 인하여 XML 문서의 저장을 위한 효과적인 모델을 설계할 수 없다. 또한 검색 시 다수의 테이블에 대한 고비용의 조인(join) 연산을 수행해야 하기 때문에 검색 시 효율이 떨어지는 단점이 있다[5].

객체지향 데이터베이스에서의 데이터 모델 설계는 파싱 후 생성된 객체를 바로 DB에 복합 객체(complex object) 형태로 저장이 가능하고 데이터 형 선언과 set-value 지원 등 객체지향적 특성을 이용할 수 있으므로 관계 데이터베이스에서의 모델 설계보다 구조적인 XML 문서를 위해 효과적으로 모델을 설계할 수 있다. 하지만 객체지향 데이터베이스 자체가 대용량의 데이터에 대한 질의 처리가 성숙되어 있지 않다는 문제점이 있다[1].

관계 데이터베이스와 객체지향 데이터베이스의 장단점을 수용한 객체관계 데이터베이스를 이용하여 XML 문서 저장 모델을 설계하면 XML 문서의 구조적인 정보를 효과적으로 표현할 수 있도록 모델설계가 가능하다. 또한 계층적 질의를 조인을 이용한 것보다 빠르게 처리할 수 있고 대용량의 데이터에 대한 질의를 안정성 있게 처리할 수 있을 것이다.

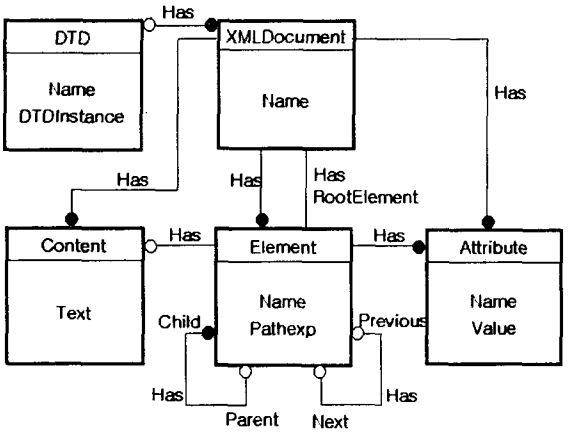
□

XML 문서의 구조정보를 데이터베이스에 저장하는 방법은 문서의 전체내용을 하나의 Blob형태로 저장하고 각 단말노드에 대하여 문서 내에서의 위치정보만을 저장하는 '가상분할' 방법과 문서를 엘리먼트 단위로 쪼개어 저장하고 검색 시 구조정보를 참조하여 해당 엘리먼트나 하위 엘리먼트들의 조합을 생성하여 처리하는 '분할저장' 방식으로 나눌 수 있다[2].

'가상분할' 방식은 문서검색 시 취합 과정이 필요 없이 위치정보만을 이용하여 구조적인 검색이 가능하므로 검색 효율이 우수한 반면 XML 문서의 내용이 수정될 경우 문서 전체의 위치를 수정해 주어야 하고 일관성을 유지해야 한다는 단점이 있다[3]. 이에 반해 '분할저장' 방식은 문서내용 수정 시 관계되는 노드들만 수정하면 되므로 가상분할 방식보다 문서내용의 갱신이 자유로우나 검색과정이 복잡하고 검색시간이 많이 걸리는 단점이 있다[9].

3. XML 문서의 모델 설계

본 절에서는 객체관계 데이터베이스의 특징을 사용하여 XML 문서를 저장하기 위한 모델을 설계한다. 객체관계 데이터베이스의 특징인 complex object와 set-value를 사용하면 관계 데이터베이스에서는 표현하기가 어려웠던 엘리먼트 간의 계층적 구조를 잘 나타낼 수 있다. DTD에 관계없이 XML 문서를 저장할 것으로 가정하여 DTD 독립적인 모델을 제시하였고 XML 문서의 빈번한 수정이 있을 것으로 가정하여 문서를 엘리먼트 단위로 쪼개어 저장하는 '분할저장' 방식을 사용하였다. 그리고 XML 문서의 요소들 중 가장 기본적인 XML 문서, DTD, 엘리먼트, Content, 애트리뷰트 정보들에 관해서만 저장하는 것으로 가정한다. <그림 1>은 XML 문서를 객체관계 데이터베이스로 사상하기 위해 설계한 모델을 OMT(Object Modelling Technique) 객체도를 이용하여 표현한 것이다[4].



<그림 1> XML 문서 저장을 위한 OMT 객체도

<그림 2>에서는 제시된 OMT 객체도를 이용하여 구성된 스키마를 보이고 각각의 테이블에 관한 설명을 덧붙인다. 빈번한 질의의 빠른 수행을 위해, 그리고 XML 문서 검색 시 질의의 복잡성을 최소화하기 위해 서로 간 참조하는 속성을 최대한 많이 포함시켰다(예: Attribute 테이블과 Element 테이블간의 참조, Content 테이블과 Element 테이블간의 참조 등). 클래스들간의 참조는 reference를 속성으로 지정하고 해당 클래스에 대한 reference를 저장하는 방식으로 구현하였다. 이는 객체관계 데이터베이스를 위한 표준 질의어인 SQL-3에서 ref와 deref 함수를 사용하여 구현할 수 있다. 또한 객체관계 데이터베이스에서는 set-value를 지원하기 때문에 DTD와 XML 문서와의 관계, Parent Element와 Child Element들과의 관계, 그리고 Element와 Attribute와의 관계를 쉽게 정의할 수 있다[7]. 문서에 대한 구조적 질의를 쉽게 하기 위하여 Element 테이블에 Pathexp 속성을 추가하였다[6].

DTD 테이블

Name	DTD 문서 이름
DTDInstance	실제 DTD 문서 저장
XMLDocument	DTD를 만족하는 XML 문서 참조

XMLDocument 테이블

Name	XML 문서 이름
DTD	XML 문서가 따르는 DTD 참조
RootElement	XML 문서의 RootElement들 참조

Element 테이블

Name	Element Tag Name
Parent	Element의 Parent Element 참조
XMLDocument	Element를 포함하고 있는 XMLDocument 참조
Children	Element의 Child Element들 참조
Prev	Element의 previousSibling 참조
Next	Element의 nextSibling 참조
Content	Element의 Content 참조
Attributes	Element의 Attribute 들 참조
Pathexp	Element의 구조 정보 저장

Content 테이블

Text	Content 내용
XMLDocument	Content를 포함하고 있는 XML Document 참조
Element	Content를 포함하고 있는 Element 참조

Attribute 테이블

Name	Attribute Name
Value	Attribute Value
XMLDocument	Attribute Name, Value 쌍을 포함하는 XML Document 참조
Element	Attribute Name, Value 쌍을 포함하는 Element 참조

<그림 2> XML 문서 저장 테이블 구조

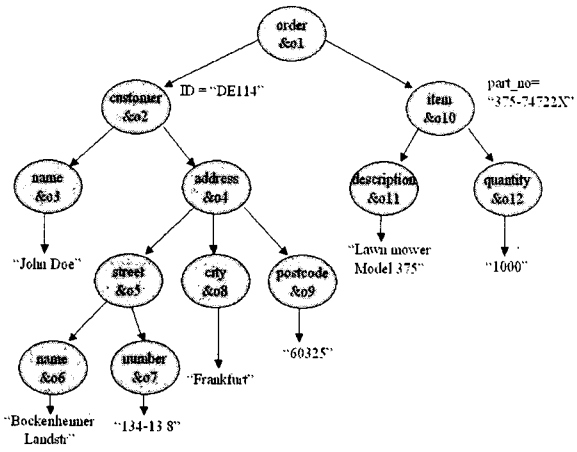
4. XML 문서 저장

3절에서 제시한 모델을 기반으로 하여 XML 문서가 어떻게 저장이 되는지 살펴보겠다.

```
<?xml version="1.0"?>
<order status="final">
  <customer ID="DE114">
    <name>John Doe</name>
    <address>
      <street>
        <name> Bockenheimer Landstr. </name>
        <number>134-13 8 </number>
      </street>
      <city>Frankfurt</city>
      <postcode>60325</postcode>
    </address>
  </customer>
  <item part_no="375_74722X">
    <description>Lawn mower model 375</description>
    <quantity>1000</quantity>
  </item>
</order>
```

<그림 3> 예제 XML 문서

<그림 3>은 예제 XML 문서이고 이 문서가 파싱되었을 때 생성된 트리 구조가 <그림 4>에 있다. 트리의 루트 노드로부터 DFS(Depth First Search) 방식으로 노드를 검색하여 요소별로 각각의 테이블에 저장한다.



<그림 4> 예제 XML 문서의 트리 구조

<그림 4>의 각각의 Element 노드에 나와 있는 숫자는 DFS 방식으로 Element들을 테이블에 저장한다는 가정 하에 생성된 OID(Object IDentifier)의 예이다. 각각의 테이블에 저장된 내용은 <그림 5>에 있다.

DTD 테이블

OID	Name	DTDContent	XMLDocument
o60	order.dtd	실제 DTD File	{o70, ..}

XMLDocument 테이블

OID	Name	DTD	RootElement
o70	order.xml	o60	o1

Element 테이블

OID	Name	Parent	XML Document	Child_list	Prev	Next	Content	Attribute List	Pathexp
o1	order	null	o70	{o2,o8}	null	null	null	{o80}	order
o2	customer	o1	o70	{o3,o4}	null	o8	null	{o81}	order.customer
o3	name	o2	o70	null	null	o4	o50	null	order.customer.name
o4	address	o2	o70	{o5,o6,o7}	o3	null	null	null	order.customer.address
o5	street	o4	o70	{o6,o7}	null	o8	null	null	order.customer.address.street
o6	name	o5	o70	null	null	o7	o51	null	order.customer.address .street.name
o7	number	o5	o70	null	o6	null	o52	null	order.customer.address .street.number
o8	city	o4	o70	null	o5	o9	o53	null	order.customer.address.city
o9	postcode	o4	o70	null	o8	null	o54	null	order.customer.address .postcode
o10	item	o1	o70	{o9,o10}	o2	null	null	{o82}	order.item
o11	description	o8	o70	null	null	o10	o55	null	order.item.description
o12	quantity	o8	o70	null	o9	null	o56	null	order.item.quantity

Content 테이블

OID	Value	XML Document	Element
o50	John Doe	o70	o3
o51	Bockenheimer Landstr.	o70	o6
o52	134-13 8	o70	o7
o53	Frankfurt	o70	o8
o54	60325	o70	o9
o55	Lawn mower model 375	o70	o11
o56	1000	o70	o12

Attribute 테이블

OID	Name	Value	XML Document	Element
o80	status	final	o70	o1
o81	ID	DE114	o70	o2
o82	part_no	375-74722X	o70	o8

<그림 5> 각 테이블의 저장 내용 예

각 객체의 판독을 쉽게 하기 위하여 임의의 OID를 지정하였고 객체 서로 간의 참조는 OID를 통해서 이루어진다는 것을 확인할 수 있다. 또한 Element 객체들의 ChildList 속성을 보면 OID의 set 값이 저장되어 있는 것을 확인할 수 있고 이를 통해 관계 데이터베이스에서 구현하기 힘들었던 엘리먼트간의 계층적 구조를 저장할 수 있다. Element 객체의 Pathexp 속성에 루트 Element로부터 해당 Element까지의 Element 이름들을 나열함으로써 차후에 XML 문서에 대하여 구조적인 질의를 할 때 이용할 수 있도록 한 것을 확인할 수 있다.

5. 결론 및 향후 연구 방향

본 연구에서는 XML 문서를 효과적으로 저장하고 내용 및 구조 정보를 검색하기 위하여 DTD 독립적인 XML 문서 저장 모델을 설계하였다. 또한 객체관계 데이터베이스의 기능들을 이용하여 기존의 관계 데이터베이스에서 구현하기에 어려움이 있었던 부분을 해결할 수 있었다.

향후 과제로는 설계된 모델을 실제 ORDBMS 시스템에 적용시켜 볼 것이고 시스템에서의 저장 및 검색 성능을 측정해 볼 것이다. 또한 내용 및 구조정보 검색을 위한 ORDBMS 질의를 연구할 계획이며 XML 문서의 다른 요소들을 포함할 수 있도록 기존에 설계한 모델을 지속적으로 개선할 계획이다.

6. 참고 문헌

[1] D. Florescu and D. Kossmann, "Storing and Querying XML Data using an RDBMS," IEEE Data Engineering Bulletin 22(3), pp.27-34, 1999.

[2] P. Francois, "Generalized SGML Repositories: Requirements and Modeling," Computer Standard & Interface, 1996.

[3] S. Malaika, "Using XML in Relational Database Applications," 15th Int'l Conf. on Data Engineering, Sydney, Australia, p.167, 1999.

[4] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, Object-Oriented Modeling and Design, Prentice-Hall, 1991.

[5] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt, and J. F. Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities," Proc. of 25th Int'l Conf. on VLDB, Edinburgh, Scotland, UK, pp.302-314, 1999.

[6] T. Shimura, M. Yoshikawa, and S. Uemura, "Storage and Retrieval of XML Documents Using Object-Relational Databases," DEXA99, pp.206-217, 1999.

[7] M. Stonebraker and P. Brown, Object-Relational DBMSs: The Next Great Wave, Morgan Kaufmann Publishers, Inc., 1996.

[8] W3C, Extensible Markup Language(XML) 1.0, http://www.w3.org/TR/1998/REC-xml19980210.html, Feb. 1998.

[9] 연제원, 장동준, 김용훈, 이강찬, 이규철, "효율적인 검색지원 SGML 저장관리기의 설계 및 구현," '99 동계 데이터베이스 학술대회 논문집, pp.136-143, 1999.