개별화된 노드 테이블 내의 분리된 경로를 사용한 XML 문서 관리 기법

봉하익, 황병연
가톨릭대학교 컴퓨터공학과
e-mail:{drvong, byhwang}@catholic.ac.kr

A Management Method of XML Documents Using Splited Path in Seperated Node Table

Ha-Ik Vong, Byung-Yeon Hwang
Dept. of Computer Engineering, The Catholic University of Korea

요 약

1996년 W3C에서 XML을 제안한 이래, 다량의 XML(eXtensible Markup Language) 문서들이 인터넷에 확산되고 있다. 이런 이유로, XML과 관련된 연구의 필요성이 증가하고 있는 실정이다. 많은 기관과 기업에서 XML 문서를 활용하고 있는 사례가 많으며, 이에 따라 XML 문서들을 저장, 검색, 그리고 관리하기 위한 XML 관리 시스템에 대한 연구가 활발히 진행되고 있다. 이에 본 논문에서는 XML 문서가 관계형 데이터베이스에 저장될 때 경로를 저장하되, 경로를 세 부분(루트 노드, 중간 노드들, 말단 노드)으로 나눠 저장하는 방식을 제안한다. 또한 각 노드 정보 테이블을 생성하여 검색 시간을 줄이는 방법을 제시한다. 그리고 XPath 질의들에 대한 처리 성능을 기존 방법과 비교 설명함으로써 제안한 방법의 효율성을 제시한다.

1. 서론

1996년 W3C(World Wide Web Consortium)에서 XML(eXtensible Markup Language)[1]이 제안된 이후 그 사용이 증가하였으며, 이로 인해 XML 데이터의 양도 빠른 속도로 늘어나고 있다. 이런 추세에 발맞추어 XML 문서에 대한 효율적인 관리 방법들이 연구되어 오고 있으며, 그 대상 역시 대용량이면서 구조가 복잡한 XML 문서들에 초점이 맞추어지고 있다.

이에 본 논문에서는 객체 관계형 데이터베이스를 기반으로 모델 매핑 방식을 사용한 XML 문서의 저장, 검색 및 관리 방법을 제시한다. 이는 XML 문서가 데이터베이스의 테이블 구조에 저장될 때 모든 가능한 경로들을 루트 테이블 내에 저장하되, 이를 루트 노드와 중간 노드들, 그리고 말단 노드의 세 부분으로 나누어 저장한다. 이어 하위 노드들로 시작하는 테이블을 생성하며, 이들 역시 경로를 세 부분으로 구분하여 저장시킨다. 이 때 발생할 수 있는 저장 공간의 낭비는 각 문자열 노드들에 고유 식별자를 부여하여 문자열 대신 숫자를 매칭함으로써, 저장 공간의 축소와 검색 시간의 단축을 기대한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 소개하고, 3장에서는 본 논문에서 제안하는 XML 문서의 데이터 모델 및 저장 스키마와 질의 처리에 대해서 설명한다. 4장에서는 예상 결과 및 효율을 설명하며, 5장에서는 결론 및 향후 연구과제에 대해 기술한다.

2. 관련 연구

XML 문서가 관리되기 위한 데이터베이스 디자인 스키마로는 구조 매핑 방식(Structure-mapping approach)과 모델 매핑 방식(Model-mapping approach) 2가지가 있다[2].

본 논문에서 사용하는 모델 매핑 방식은 고정된 데이터베이스 스키마가 DTD에 관한 정보 없이 XML 문서 모델의 구조를 나타낸다. 이 접근법으로 고정된 데이터베이스 스키마는 모든 XML 문서들의 구조를 저장하기 위하여 사용된다. 그 예로는 XML 문서 트리 내 에지(edge)가 관계형 튜플로써 저장되는 에지 접근법(edge approach)[3]과 관계형 테이블을 이용한 스키마 레벨 방법에 기반하면서 역 인덱스 기술을 이용하는 방법[4]이 있다. 그리고 경로와 영역(region)의 결합이라는 방식으로 XML 트리 구조를 표현한 XRel[2] 등이 있다.

XRel은 XML 문서의 트리 구조 내에서 인스턴스 내의 루트노드를 제외한 루트로부터 각 노드까지의 모든 경로들을 열거했으며, 관계형 속성들내의 경로 표현들 그 자체를 저장시켰다. 또한 모든 가능한 경로 표현들이 하나의 문자열로써 데이터베이스에 저장되기 때문에 문자열 매칭(String Matching)이라는 방식으로 처리할 수 있다. XRel에서 XML 데이터 모델을 저장하기 위한 데이터베이스 스키마는 다음과 같다.

Element table(docID, pathID, start, end, index, reindex)
Attribute table(docID, pathID, start, end, value)
Text table(docID, pathID, start, end, value)
Path table(pathID, pathExp)
Document table(docID, docExp)

각각은 엘리먼트 정보, 속성 정보, 자료 값 정보, 경로 표현식, 그리고 문서 정보를 저장하고 있다. 여기서 ID와 Exp는 각각의 고유 식별자와 표현명을 나타내며, value는 해당 자료 값을 나타낸다. start와 end는 포함관계를 통하여 XML 문서 내 해당 영역을 나타낸다. index와 reindex는 같은 경로를 갖는 노드들 사이의 순서 관계를 pre-order 방식으로 나타내기 위하여 사용된다[2].

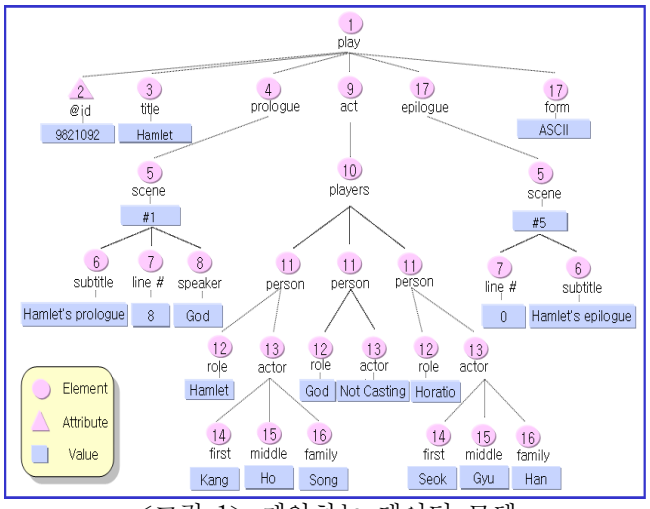
3. 제안하는 데이터 모델 및 저장 스키마와 질의 처리

XML 문서는 트리로 볼 수 있으며, 단말 노드는 자료 값 즉, 텍스트에 해당되고 내부 노드는 XML 엘리먼트에 해당된다. 그러나 일반적인 XML 문서에서는 단말 노드만이 자료 값을 갖는 형태를 벗어나 내부 노드도 자료 값을 갖는 문서가 존재하므로 내부 노드 역시 자료 값을 가진 문서를 바탕으로 한다.

제안 방식에서는 XML 질의어로 XPath (XML Path Language)를 사용한다. 데이터베이스 스키마를 위한 접근법으로는 모델 매핑 방식을 사용한다. 이는 모델 매핑 방식이 구조 매핑 방식보다 유동적인 상태의 데이터 관리에 적합하기 때문이다. 많은 복잡한 웹 에플리케이션들이 XML의 유연하면서도 동적인 사용에 기반한다. 그러나 구조 매핑 방식은 주로 제한된 수의 문서 구조나 DTD를 저장할 때 또는 이들이 고정된 상태에서 적합하기 때문에 유동적인 XML 문서 관리에는 부적합하다[2].

3.1 XML 문서의 데이터 모델

<그림 1>은 제안하는 XML 데이터 모델을 보여주고 있다. 여기서 노드에 식별자를 부여하는 방식은 pre-order 방식을 사용하되, 기존의 노드명과 동일한 노드명이 나오게 되면 새 노드 식별자를 부여하는 것이 아니라, 기존 노드명의 식별자를 사용한다.



<그림 1> 제안하는 데이터 모델

3.2 데이터베이스 저장 스키마

트리 구조를 갖고 있는 XML 문서에 대한 경로를 표현하기 위해 루트 노드로부터 말단 노드까지의 경로 내 모든 노드들을 나열하여 ***경로 및 노드 정보 테이블***에 저장시켜 질의 검색 시 사용한다.

이 때, 경로를 저장하는 조건은 모든 가능 경로를 저장하는 것이 아니라 말단 노드가 자료 값을 갖는 경로만 저장한다. 이는 사용자가 질의 검색 시 자료 값이 없으면 NULL 값을 결과로 나타내기 때문에 경로 정보를 저장하지 않음으로써 저장 공간의 축소를 야기할 수 있다.

그리고 노드명에 해당하는 노드 식별자를 할당함으로써, 문자열 대신 숫자를 저장시켜 매칭하는 방식을 사용한다. 이는 매칭할 데이터의 양을 줄여줌으로써 검색 속도를 향상시킨다.

<그림 2>는 <그림 1>의 데이터 모델을 바탕으로 한 ***노드 식별자 테이블***과 ***경로 정보 테이블***이다. ***Node 테이블***은 저장 및 검색 시 모두 사용되지만, ***Path 테이블***은 경로 정보 저장 시에만 사용한다.

Node	
NodeID	*NodeExp*
1	play
2	@id
3	title
4	prologue
5	scene
6	subtitle
7	line #
8	speaker
9	act
10	players
11	person
12	role
13	actor
14	first
15	middle
16	family
17	epilogue
18	form

Path	
PathID	*NodeOrd*
1	/1/2
2	/1/3
3	/1/4/5
4	/1/4/5/6
5	/1/4/5/7
6	/1/4/5/8
7	/1/9/10/11/12
8	/1/9/10/11/13/14
9	/1/9/10/11/13/15
10	/1/9/10/11/13/16
11	/1/9/10/11/13
12	/1/17/5
13	/1/17/5/7
14	/1/17/5/6
15	/1/18

<그림 2> 노드 식별자 테이블과 경로 테이블

그리고 데이터베이스 내의 테이블에 저장되는 경로 정보는 루트 노드(n = number of node, n=1), 중간 노드들(n≥0), 말단 노드(n=1) 세부분으로 나눠져 저장된다. 이는 질의 유형에 따라 테이블 내 특정 열만 검색함으로써 질의 처리 속도를 향상시키기 위함이 그 목적이다. 예를 들어, 깊이가 2인 질의(예 : '/play/title')에 대해 처리할 경우, <그림 3>의 ***루트 노드 경로 테이블***을 통하여 루트 노드(예 : 'play')와 말단 노드(예 : 'title'), 그리고 깊이가 '2'라는 정보만으로도 원하는 결과를 얻을 수 있다. 이는 테이블 내 중간 노드들이 저장되어 있는 열은 검색할 필요가 없음을 의미하고, 결과적으로 검색할 데이터의 양이 적어지게 된다.

<그림 3>은 ***루트 노드 경로 테이블***을 나타낸 그림이며, 이것은 <그림 2>의 ***Node 테이블***과 ***Path 테이블***을 바탕으로 생성된다. 제시한 그림에서는 이해를 돕기 위하여 테이블 내 모든 노드명을 문자열 형태로 표현하였지만, 실제로 저장될 때는 노드 식별자를 통해 숫자로 표현된다. 이는 다음에 나오는 <그림 4>와

<그림 5>의 ***하위 노드 경로 테이블***에도 동일하게 적용된다.

Root_Node Path Table				
Ele_ID	Root_Node	Mid_Nodes	End_Node	Length
1	/play	null	/@id	2
2	/play	null	/title	2
3	/play	/prologue	/scene	3
4	/play	/prologue/scene	/subtitle	4
5	/play	/prologue/scene	/line #	4
6	/play	/prologue/scene	/speaker	4
7	/play	/act/players/person	/role	5
8	/play	/act/players/person/actor	/first	6
9	/play	/act/players/person/actor	/middle	6
10	/play	/act/players/person/actor	/family	6
11	/play	/act/players/person	/actor	5
12	/play	/epilogue	/scene	3
13	/play	/epilogue/scene	/line #	4
14	/play	/epilogue/scene	/subtitle	4
15	/play	null	/form	2

<그림 3> 루트 노드 경로 테이블

또한, 제안 방식은 루트 노드로부터 말단 노드까지의 모든 가능 경로를 저장하는 테이블(Root_Node Path Table) 뿐만 아니라, 루트 노드 이외의 노드로부터 시작하는 경로 테이블(Sub_Node Path Table)도 생성하게 된다. 이 ***하위 노드 경로 테이블***에서는 해당 노드보다 낮은 깊이에 있는 노드 정보는 루트 노드를 제외하고는 저장하지 않는다. 예를 들어, 생성하고자하는 하위 노드 경로 테이블의 해당 노드의 깊이가 4인 경우, 4보다 작은 깊이의 노드의 정보는 루트 노드를 제외하고는 저장하지 않는다. 예를 들어, ***'actor'***(깊이 5) ***노드 경로 테이블***을 생성할 경우, 'act'(깊이 2), 'players'(깊이 3), 'person'(깊이 4)을 제외한 채 'play'(깊이 1), 'first'(깊이 6), 'middle'(깊이 6), 'family'(깊이 6)에 대한 데이터만 저장한다.

하위 노드에 대한 경로 테이블을 생성할 때, 해당 노드의 깊이가 2인 경로 단 하나만('/루트 노드/해당 노드') 존재할 경우에는 하위 경로 정보 테이블을 생성하지 않는다. 예를 들어, <그림 1>의 데이터 모델에서 노드 '@id', 'title', 'form'은 깊이가 2이면서, 해당 노드를 포함하고 있는 경로가 단 한 개이므로 해당 노드에 관한 ***하위 노드 경로 테이블***을 생성하지 않는다. 이는 깊이가 2인 경로 질의에 대해서는, 앞에서 설명한 바와 같이 ***루트 노드 경로 테이블***을 통하여 처리할 수 있기 때문에, 불필요한 테이블을 생성하지 않음으로써 저장 공간의 효율을 높이는 것이다.

<그림 4>는 <그림 1>의 데이터 모델을 바탕으로 생성되게 될 ***하위 노드 경로 테이블***의 예를 나타낸 그림이다.

act_Node Table				
Root_Ele_ID	Root_Node	Mid_Nodes	End_Node	Length
7	/play	/players/person	/role	5
8	/play	/players/person/actor	/first	6
9	/play	/players/person/actor	/middle	6
10	/play	/players/person/actor	/family	6
11	/play	/players/person	/actor	5

<그림 4> 하위 노드 경로 테이블 ('act' Node Table)

그리고 각 경로가 나타내는 자료 값은 ***자료 값 테이블***(Value_Table)에 저장되며, 검색 시 사용되는 경로 테이블과 참조되어 질의 처리를 수행하게 된다.

<그림 5>는 <그림 1>의 데이터 모델을 바탕으로 한 ***자료 값 테이블***이다. 이 때 PathIndex는 동일한 경로 표현식을 갖지만 자료 값이 다른 경우, 이를 구분하기 위하여 사용하는 식별자이다. 예를 들어, <그림 1>의 데이터 모델에서 '/play/act/players/person/role'의 경로는 모두 3개가 나오며, 이 경로들은 모두 다른 자료 값을 갖고 있다. 이 때, 이 3개의 경로에 고유 식별자인 PathIndex를 부여하여, 서로 다른 자료 값들을 개별적으로 저장한다.

Value_Table				
ValueID	DocID	Root_Ele_ID	PathIndex	Value
1	1	1	1	9821092
2	1	2	1	Hamlet
3	1	3	1	#1
4	1	4	1	Hamlet's prologue
5	1	5	1	8
6	1	6	1	God
7	1	7	1	Hamlet
8	1	8	1	Kang
9	1	9	1	Ho
10	1	10	1	Song
11	1	7	2	God
12	1	11	1	Not Casting
13	1	7	3	Horatio
14	1	8	2	Seok
15	1	9	2	Gyu
16	1	10	2	Han
17	1	12	2	#5
18	1	13	1	0
19	1	14	1	Hamlet's epilogue
20	1	15	1	ASCII

<그림 5> 자료 값 테이블

이와 같은 내용을 바탕으로 한 데이터베이스 스키마는 다음과 같다.

Document(DocID, DocExp)
Node(NodeID, NodeExp)
Path(PathID, PathExp)
Root_Node Path(Ele_ID, Root_Node, Mid_Nodes, End_Node, Length)
Sub_Node Path(Root_Ele_ID, Root_Node, Mid_Nodes, End_Node, Length)
Value(ValueID, DocID, Root_Ele_ID, PathIndex, Value)

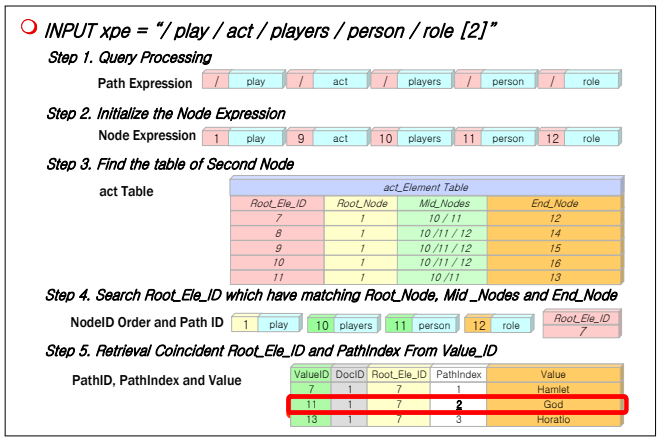
3.3 질의 처리

제안 방식은 질의 유형에 따라 질의 처리 방법이 달라진다. 질의가 일반적인 부모-자식 질의인 '/' 또는 조상-후손 관계를 나타내는 질의인 '//'로 시작되느냐에 따라 처리 방식을 달리할 수 있다.

'/'로 시작하는 일반적인 질의의 경우에는 다음과 같은 방법으로 질의를 처리한다.

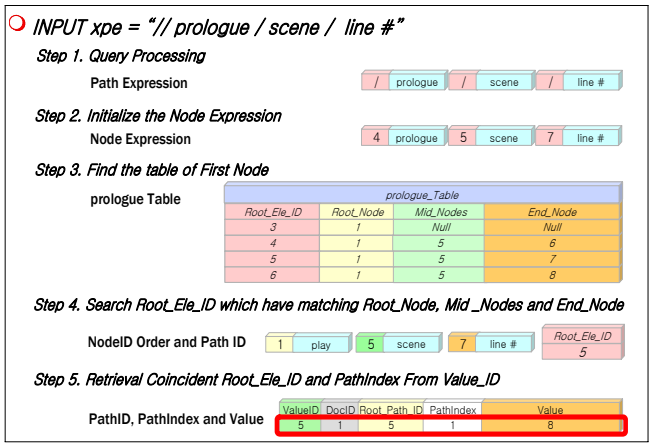
1. 입력된 질의의 첫 번째 노드명을 확인하여 해당하는 ***하위 노드 경로 테이블***을 검색한다.
2. 해당 노드 경로 테이블에서 루트 노드와 중간 노드들, 말단 노드를 매칭한다. XRel의 경우에는 모든 경로 정보를 포함하는 테이블 내에서 문자열을 매칭해야 하지만, 제안 방법은 두 번째 노드로부터 시작하는 테이블에서 일치하는 정보를 찾으면 된다.

<그림 6>은 부모-자식 질의를 처리하는 과정을 보여준다. XRel의 경우에는 검색 대상이 ***경로 테이블***의 15행이지만, 제안 방식은 ***'act' 노드 경로 테이블***의 5행만 검색하면 되므로 매칭 시간이 줄어들게 된다.



<그림 6> 부모-자식 경로 질의 검색 과정

'//'로 시작하는 일반적인 질의는 위 질의 처리 방식처럼 첫 번째 노드로 가는 것이 아니라, 입력된 질의의 첫 번째 노드명을 확인하여 해당하는 ***하위 노드 경로 테이블***을 검색한다는 점이 다르다. <그림 7>은 조상-후손 관계 질의를 처리하는 과정을 보여준다.



<그림 7> 조상-후손 경로 질의 검색 과정

4. 예상 결과 및 효율

제안 방법은 조건에 맞는 노드마다 테이블을 생성함으로써, 기존 방식에 비하여 저장 공간을 많이 차지하는 단점이 있다. 이 단점을 보완하기 위하여, 문자열 매칭 대신에 숫자 매칭을 사용하고, 자료 값을 갖지 않는 경로는 저장하지 않는 방식을 채택하였다.

제안 방식은 XRel의 문제점으로 지적되어 온 '//'로 시작하는 조상-후손 관계 질의에 대해 우수한 성능 개선을 보일 것으로 예상된다[5]. XRel은 질의문이 '//'로 시작하는 경우, 경로 정보 테이블에 있는 모든 경로들을 검색해야 한다. 그러나 제안 방식은 '//' 바로 다음에 오는 노드 테이블에서 경로를 매칭하면 된다.

예를 들어, <그림 7>에서 볼 수 있는 '//prologue/scene/line #'에 대한 질의는 ***'prologue' 노드 경로 테이블***에서 검색을 실행하게 된다. 이 때, 2가지 측면에서 제안 방식이 우수한 성능을 보이게 된다.

첫 번째는 검색할 테이블 내 행의 개수가 적어진다.

XRel은 해당 질의문을 XML 문서내의 모든 경로와 매칭해야 한다. 이는 경우에 따라 경로 테이블의 모든 행을 검색해야하는 것을 의미한다. 반면, 제안 방식은 '//' 바로 다음에 오는 노드 테이블에 있는 정보만 검색하게 된다. ***하위 노드 경로 테이블***은 해당 노드로부터 시작하는 경로만 저장하고 있으므로, 모든 경로를 저장하는 테이블에 비해 테이블 내 행의 수도 줄어들게 되며, 이로 인해 검색할 정보량이 줄어든다.

두 번째는 검색할 테이블 내의 한 행에 있는 데이터 양이 적어진다.

XRel은 이런 질의를 처리하기 위하여, ***경로 테이블*** 내에 저장되어 있는 행의 모든 문자열을 매칭한다. 그러나 '//' 질의는 '//' 질의 앞에 생략된 알 수 없는 노드 정보를 검색할 필요가 없다. 이 점에 착안하여, 제안 방식은 해당하는 ***하위 노드 경로 테이블***이 루트 노드 이외의 해당 노드보다 낮은 깊이의 경로는 저장하지 않기 때문에, 매칭해야 하는 경로의 문자열이 줄어드는 것이다.

물론, 제안 방식은 '/'로 시작하는 질의 및 '/'와 '//'가 질의 내에 공존하는 유형에도 앞에서 설명한 바와 같이, 검색 대상이 되는 테이블 내의 정보가 줄어드는 이유를 바탕으로 기존 방식에 비하여 우수한 성능을 나타낼 것으로 예상된다.

5. 결론 및 향후 연구 과제

본 논문은 관계형 데이터베이스를 기반으로 하여 질의 유형에 따라 데이터베이스의 테이블을 검색하는 XML 문서 관리 기법을 제안하고 있으며, 기존 연구의 단점을 보완하는 방법이 된다. 현재 실험이 진행 중에 있으며, 실험을 바탕으로 다른 연구들과의 성능 비교를 할 예정이다.

앞으로 테이블 생성의 다량화를 보완하여, 좀 더 간결한 관계형 데이터베이스 스키마를 연구할 계획이다.

또한, 스트림 데이터 처리 등 XML 문서에 대한 확장된 영역의 연구도 진행할 계획이다.

참고문헌

[1] "Extensible Markup Language(XML) 1.1", W3C Candidate Recommendation, 2002

[2] M. Yoshikawa and T. Amagasa, "XRel: A Path-Based Approach to Storage and Retrieval of XML Documents using Relational Databases," ACM Transactions on Internet Technology, Vol. 1, No. 1, pp. 110-141, 2001

[3] D. Florescu and D. Kossmann, "Storing and Querying XML Data Using an RDBMS," IEEE Data Engineering Bulletin, Vol. 22, No. 3, pp. 27-34, 1999

[4] 이윤호, 최일환, 김종익, 김형주, "색인된 XML 문서에서 레벨 정보를 이용한 효과적인 구조 조인 기법," 정보과학회논문지, 제32권, 제6호, pp. 641-649, 2005

[5] Felix Weigel, Klaus U. Schulz, and Holger Meuss, "Exploiting Native XML Indexing Techniques for XML Retrieval in Relational Database Systems," In Proc. of ACM, Bremen, Germany, pp. 23-30, 2005