XML 의 RDB 로의 맵핑을 위한 효율적인 스키마 변환

김태희*, 김선경**
*대구대학교 컴퓨터정보공학과 박사과정
** 대구대학교 정보통신공학부 교수

The Efficient Schema Conversion to map the XML Document into the RDB

Tae-hee Kim*, Sun-kyung Kim**
*Dept. of Computer and Information, Dae-gu University
*School of Computer and Communication, Dae-gu University

요 약

웹상의 문서는 데이터 중심의 표준 언어인 XML 문서로 표현된다. XML 데이터를 범용적이고 우수한 성능의 관계형 데이터베이스와 연동하여 이용하기 위해서는 두 구조 사이의 맵핑 과정이 필요하다. 계층적 구조의 XML 문서와 데이터베이스의 평면적인 구조의 차이를 고려한 구조 맵핑을 위해서 검색 시스템에 적합한 가상 분할 방식으로 엘리먼트를 분석하여 관계 스키마를 정의한 후 XML 문서를 저장한다. 이를 위해 스키마는 DTD 에 독립적인 형태로 변환하고, 엘리먼트의 추가와 삭제, 검색의 효율성을 위해 노드간의 위치 정보와 함께 독립적인 ID를 부여하여 구조적 검색을 수행할 수 있게 하였다.

1. 서론

웹상의 문서에 대한 정보의 공유와 데이터 교환에 대한 요구가 급증하여 이에 대처하기 위한 방안으로 웹상의 문서는 데이터 중심의 표준 언어인 XML 문서로 표현되고 있다. XML 은 내용을 표현하기 보다는 문서의 구조나 정보를 표현하기에 적합한 언어로, 사용자가 내용과 관련된 엘리먼트를 직접 만들어서 사용할 수 있어 데이터 자체가 데이터베이스 역할을 할 수 있으나 대용량의 데이터를 안정적으로 다루기 위해서는 별도의 저장 시스템이 요구된다. XML 전용 데이터베이스를 사용할 수 있으나, 기존의 데이터베이스 시스템의 범용성과 기존의 데이터베이스 시스템에서 지원하고 있는 제반 기술들을 사용할 수가 없다. 현실적으로 범용성을 가진 관계 데이터베이스는 질의 처리 면에서 다른 저장 시스템에 비해 우수 하고 XML 데이터와 기존의 관계 데이터가 동시에 저장 가능하여 XML 을 관계데이터베이스 시스템에 저장하기 위한 연구가 계속되고 있다. 엄격한 스키마 구조를 가진 기존의 관계 데이터베이스

시스템을 이용하기 위해서는 두 구조 사이의 맵핑 과정이 필요하다.

XML 문서의 구조 정보를 저장하여 관계 데이터베이스의 테이블 형태로 구성하여 저장하는데 있어 XML 의 계층적 구조와 데이터베이스의 평면적인 구조의 차이에 대한 처리과정이 필요하다.

XML 문서를 관계 데이터베이스에 저장하는 방법은 모델은 모델 맵핑과 구조 맵핑 방법이 있으나[5] 이 논문에서는 구조 맵핑의 방법으로 접근한다.

구조 맵핑 방법은 구조 정보를 나타내는 엘리먼트를 분석하여 관계 스키마를 정의하여 XML 문서를 저장한다. 이를 위해 스키마는 DTD 에 독립적인 형태로 변환하고, 엘리먼트의 추가와 삭제, 검색의 효율성을 위해 노드간의 위치 정보와 함께 독립적인 ID 를 부여한 테이블을 작성하여 효율적인 구조적 검색을 지원하게 한다.

질의 처리 과정에서는 관계 데이터베이스에서는 XML 질의 언어가 지원되지 않으므로 XML 질의 언어인 Xquery 질의를 SQL 질의로의 변환이 필요하며 이러한 과정을 통해 SQL 질의 결과는 XML 형태로 얻을 수 있다.

2. 관련 연구

XML 문서의 구조 정보를 관계 데이터베이스에 저장하는 방법으로 분할 저장 기법과 가상 분할 저장 기법을 사용할 수 있다.[5]

분할 저장 기법은 XML 문서를 엘리먼트의 단위로 나누어 저장하여 검색 시 구조 정보를 해당 엘리먼트나 하위 엘리먼트의 조합을 생성하여 처리한다. 엘리먼트의 추가, 수정, 삭제 시 관계되는 엘리먼트만 수정하여 편집과 관리 면에서 쉽고 노드를 공유할 수 있는 장점이 있지만, 검색 시에 각 엘리먼트의 내용을 조합하여 결과를 구성해야 하므로 검색 시간이 많이 걸리고 DTD 구조 정보가 변하면 관련된 데이터베이스 테이블을 모두 재구성해야 하는 단점이 있다.

가상 분할 기법으로 XML 문서를 관계 데이터베이스에 저장할 때는 각 단말 노드에 대하여 실제 문서에서 가지는 시작 오프셋과 종료 오프셋을 저장하는 방식으로, 전체 문서를 저장하는 분할 방식과는 달리 통합 과정이 필요하지 않아 노드 검색 시 위치 정보를 이용한 구조 검색이 가능하고 검색 효율이 상대적으로 우수하다. 이러한 장점과 함께 엘리먼트의 추가, 삭제시에 다른 엘리먼트들의 위치 정보도 수정해야 하므로 이에 대한 오버헤드와 데이터베이스의 일관성 유지에 대한 문제가 있다. 따라서 주로 검색을 위주로 하는 시스템에 적합하다.

그 외 분할 저장 기법과 가상 분할 방식을 병합한 형태의 페이징 기법은 가상 분할 방법을 변형하여 문서를 여러 페이지로 나누어 저장하여 검색시 노드의 개수를 줄이고 연산시 범위가 항상 페이지 내로 제한되어 오버헤드를 줄이는 방법이다.

검색 기법으로는 구조적 특징을 고려한 내용 검색에 대한 연구가 활발하게 이루어지고 있으며 계층적 관계에 기반 한 질의와 수평적 관계에 기반 한 질의로 구분된다[3]. XML 문서를 각 노드에 식별자를 할당한 후 인덱스를 구축하여 주어진 경로 조건에 따라 상향식 혹은 하향식 트리 순회를 하여 검색 영역의 확장으로 인한 추가적인 필터링 과정이 필요하며[9], 루트로부터 시작하는 단순 경로에 대한 구조적 정보를 유지하여 트리를 직접 방문하지 않고 해당 엘리먼트만을 접근할 수 있으나, 루트에서 시작되지 않고 엘리먼트로부터 시작되는 질의 처리에 대해서는 검증된 바 없다.

3. XML 문서의 관계형 스키마로의 맵핑

XML 스키마는 DTD 에 독립적인 형태를 사용하고, 구조적 맵핑을 위해 구조 정보를 나타내는 엘리먼트를 분석하여 관계 스키마를 정의하고 분할 저장 기법을 사용하여 저장한다. 이를 위해 스키마는 DTD 에 독립적인 형태로 변환하고, 엘리먼트의 추가와 삭제, 검색의 효율성을 위해 노드간의 위치 정보와 함께 독립적인 ID 를 부여한다.

3.1 XML 문서의 구조

XML 문서를 스카마를 생성시켜 저장한다.

```
<!-- ======= Start Entity Declaration ======= -->
<!-- ======= End Entity Declaration ======= -->

<!-- ======= Start Element Declaration ======= -->
<!ELEMENT papers (paper)+>
<!ELEMENT paper (style? , title , editor+ , author+ , summary)>
<!ELEMENT editor (society , date)>
<!ELEMENT society (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT author (lname+ , fname)>
<!ELEMENT lname (#PCDATA)>
<!ELEMENT fname (#PCDATA)>
<!ELEMENT summary (keyword)+>
<!ELEMENT keyword (#PCDATA)>
<!-- ======= End Element Declaration ======= -->
```

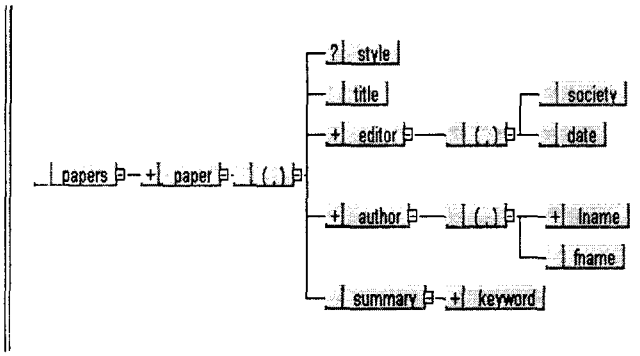


그림 1. XML 문서의 구조

엘리먼트는 EBNF(Extended Backus-Naur Form) 표기법에 따라 나타낸다.

3.2 관계 데이터베이스 스키마로의 맵핑

XML 문서를 관계데이터베이스 스키마 구조로 변환하기 위하여 ID 가 부여된 테이블은 다음과 같다. 구조적 검색 질의로의 확장을 위해 검색 경로를 엘리먼트 타입을 이용하여 최소화하기 하기 위하여 위치 정보를 내포하는 ID 를 부여한다.

UID 는 전체 문서에 대한 일반적인 내용을 나타내는 테이블이고, DTD 는 엘리먼트 정보를 담은 테이블이며, 각 엘리먼트의 목록 테이블이 EID 테이블이다. ID 의 부여는 0~9, A~Z, a~z 순으로 62 개의 문자를 사용하며 ASCII 코드의 순서를 따른다. 총 3844 개의 엘리먼트를 표현할 수 있다.

docid	docname	description
1	paper.xml	<\\\\\>

표 1. UID 테이블

docid	pid	type	content
1	01	1	<papers>
1	02	2	<paper style="cc
1	03	1	<title>
1	04	9	A study on xml
1	05	3	</title>
1	06	1	<editor>
1	07	1	<society>
1	08	9	IEEE
1	09	3	</society>
1	0A	1	<date>
1	0B	9	2003/3
1	0C	3	</date>
1	0D	3	</editor>
1	0E	1	<author>
1	0F	1	<fname>
1	0G	9	Junhee
1	0H	3	</fname>
1	0I	1	<lname>
1	0J	9	Lee
1	0K	3	</lname>
1	0L	1	<fname>
1	0M	9	Sunny
1	0N	3	</fname>
1	0O	1	<lname>
1	0P	9	Park
1	0Q	3	</lname>
1	0R	3	</author>
1	0S	1	<summary>
1	0T	9	This is a paper
1	0U	1	<key word>
1	0V	9	XML
1	0W	3	</key word>
1	0X	9	study.
1	0Y	3	</summary>
1	0Z	3	</paper>
1	0a	3	</papers>

표 2. DTD 테이블

docid	element	etid
1	author	001
1	paper	002
1	papers	003
1	editor	004
1	fname	005
1	keyword	006
1	lname	007
1	style	008
1	summary	009
1	title	00A

표 3. EID 테이블

NID 테이블은 노드의 경로를 나타낸다.

docid	pathexp	pathid
1	/003	1
1	/033/002	2
1	/003/002/00A	4
1	/003/002/004	5
1	/003/002/004/005	6
1	/003/002/004/007	7
1	/003/002/001	8
1	/003/002/001/007	10
1	/003/002/009	11
1	/003/002/009/006	12

표 4. NID xpdlqmf

엘리먼트의 실제 위치에 대한 정보는 EPID 테이블에 표현되어 있다.

docid	pathid	pid1	pid2	ppid1
1		01	0a	<NULL>
1	2	02	0Z	01
1	4	03	05	02
1	5	06	0D	02
1	6	07	09	06
1	7	0A	0C	06
1	8	0E	0R	02
1	9	0F	0H	0E
1	10	0I	0K	0E
1	9	0L	0N	0C
1	10	0O	0Q	0E
1	11	0S	0Y	02
1	12	0U	0W	0S

표 5. EPID 테이블

docid	pathid	attvalue	pid1	pid2
1	3	conference	02	0Z

표 6. ATID 테이블

그 외 속성을 나타내는 ATID 테이블, 내용정보를 저장하는 CTID 테이블이 생성되어야 한다.

docid	pathid	content	pid
1	4	A study on XM_	04
1	6	kkkk	08
1	7	2003.3	0B
1	9	Junhee	0G
1	10	Lee	0J
1	9	Sunny	0M
1	10	Park	0P
1	11	Thih is a paper	0T
1	12	XML	0V
1	11	study.	0X

표 7. CTID 테이블

스키마 구조에 따른 인덱스 테이블의 구조는 다음과 같다.

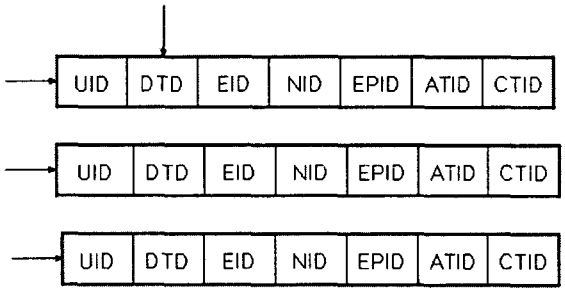


그림 2. 인덱스 테이블의 구조

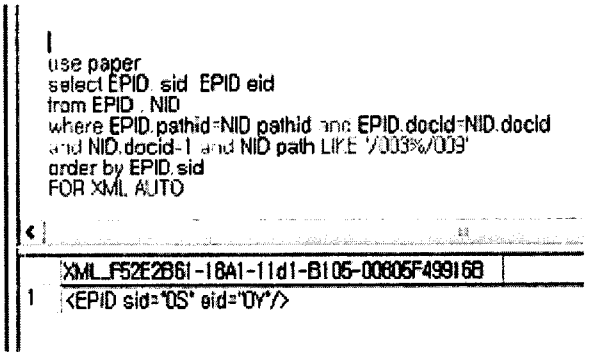
모든 엘리먼트에 대해 생긴 ID에 근거한 질의는 UID를 기준으로 한 수평 검색과 DTD를 기준으로 한 수직 검색이 가능하다.

3.3 구조적 질의

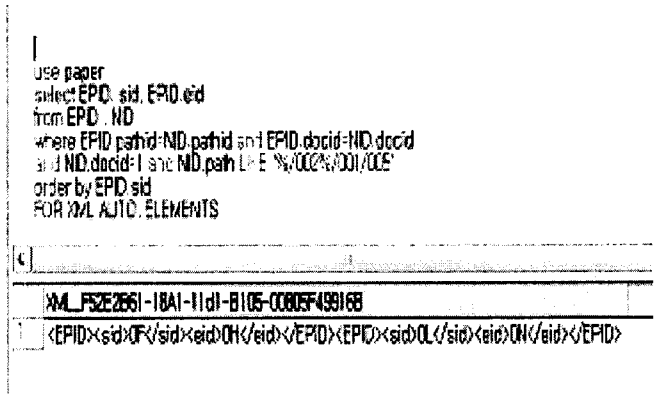
구조적 검색의 질의는 트리 구조로 표현되며 각 엘리먼트는 노드로 맵핑 되어 식별자를 할당하고 인덱스를 생성시켜 주어진 경로식에 따라 XML 문서 트리를 순회한다. 검색 조건 엘리먼트는 질의의 시작이 되는 부분이며, 검색 대상 엘리먼트는 질의의 최종 노드이며 방향 조건은 부모와 자식, 조상과 후손, 형제 등이 될 수 있다.

papers 엘리먼트의 자식 엘리먼트 중의 하나인 summary 엘리먼트를 검색하라는 Xquery 질의/papers//summary 에 대해 변환된 SQL 질의는

다음과 같이 수행된다. 검색 질의의 기준 엘리먼트는 papers 가 되고 검색 대상 엘리먼트는 summary 가 되어, papers 의 자식 엘리먼트이면서 주어진 질의와는 다른 경로상에 있는 엘리먼트의 접근은 필요하지 않게 된다.



Xquery //paper/summary/keyword 에 대한검색은 동일한 부모를 갖는 자식중 같은 엘리먼트명을 가지는 형제들의 자식은 부모 노드의 위치값 parid를 기준으로 처리된다.



4. 결론

XML 문서의 관계형 데이터베이스 스키마로의 맵핑에 있어서 구조적 특성을 고려하여 수행할 수 있는 문서 저장 구조를 제안하였다. 이를 위해 각 엘리먼트의 구조 정보와 위치 정보를 나타내는 색인 구조를 부여하여 효율적인 검색을 지원하게 되어 XML 형태의 결과를 반환받는다. 데이터의 증가로 인한 ID 부여에 대한 기법에 대한 고찰은 향후 연구과제로 남는다.

참고문헌

[1] 김훈, 한상웅, 홍의경, "XML 문서 저장 시스템 ", 데이터베이스 연구회지, 16 권 2 호, 2000.12.

[2] 박경현, 이경휴, 류근호, "DTD 가 없는 XML 데이터의 효율적인 저장 기법", 정보처리학회 논문지 제 8-D권 제 5호, 2001.10,

[3] 김성완, "XML 문서에서의 순수 구조 질의에 대한 인덱싱 및 질의 처리", 한국 정보과학회 추계학술 발표 논문집, 2002.

[4] 성린, 윤용익, "XML 문서에서의 엘리먼트 정보를 이용한 스키마 추출 방법", 정보처리학회 논문지 제 9-D권 제 3호, 2002. 6.

[5] S. Malaika, Using XML in Relational Database Applications", 15th Conf. on Data Engeneering, Sydney, Australia, p.167, 1999.

[6] P. Florescu and D. Kossman, "Storaging and Querying XML Data Using an RDBMS", IEEE Data Engineering Bulletin 22(3), pp.27-34, 1999.

[7] Dan Suciu, "Semistructured Data and XML", Proceedings of International Conference on Foundation of Data Organization, 1998.

[8] Takeyuki Shimura, Masatoshi Yoshikawa, Shunsuke Uemura, "Storage and Retrieval of XML Documents Using Object-Relational Database", DEXA99, pp. 206-217.

[9] J.McHugh et., "Indexing - Semistructured Data", Technical Report, Stanford Univ., 1998.

[10] R. Goldman and J. Widom, "DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases", VLDB 1997.