XML 문서와 데이터베이스 통합을 위한 SQL의 확장

이찬구*, 정원일, 이충호, 김종훈, 배해영
인하대학교 전자계산공학과

Extension of SQL for Integrating XML Document with Database

Chan-Gu Yi, Won-Il Chung, Chung-Ho Lee, Jong-Hoon Kim, Hae-Young Bae
Dept. of Computer Science & Engineering, Inha University

요 약

XML 은 다양한 정보의 표현이 가능하고 이들 정보의 구조적 표현이 가능하여 많은 연구 분야에서 사용되고 있다. 이러한 XML 문서의 효율적인 검색과 저장 관리를 위하여 데이터베이스 시스템을 이용한 연구가 활발히 진행되고 있으나 기존의 연구들은 XML 문서에 대한 질의 수행을 위해 SQL 과 독립된 질의 언어를 정의함으로써 데이터베이스와 통합된 질의를 제공하지 못한다.

본 논문에서는 XML 문서와 데이터베이스의 통합된 질의를 위하여 XML 문서의 엘리먼트를 자료형으로 정의하는 DDL 구문의 추가와 질의어에서 이들 자료형에 대한 연산을 지원하는 DML 의 확장을 한다. 제안한 질의어는 SQL 을 이용하여 XML 문서에 대한 질의를 처리함으로써 XML 문서와 데이터베이스의 통합된 질의를 가능하게 한다.

1. 서론

인터넷에는 수 많은 양의 정보들이 전자 문서의 형태로 존재하고 있으며, 이러한 정보들의 수는 계속하여 증가하고 있다. 이들 정보를 효율적으로 저장하고 검색, 이용하기 위하여 W3C(World Wide Web Consortium)에서는 차세대 인터넷 전자 문서의 표준으로써 XML(eXtensible Markup Language)[1]을 채택했다.

XML 은 다양한 정보의 표현이 가능하고 이들 정보의 구조적인 표현이 가능하여 연구 분야에서 사용되고 있으며 이러한 XML 문서를 효율적으로 검색하고 저장하기 위한 시스템에 관한 연구가 활발히 진행되고 있다[3][4][5].

XML 문서의 저장 및 검색 시스템에 관한 연구는 XML 문서의 효율적인 검색을 위해서 구조적인 특징을 반영한 저장 시스템의 설계에 관한 연구와 기존의 데이터베이스 시스템을 이용하여 XML 문서를 저장하기 위한 연구로 진행되고 있다.

XML 문서가 데이터베이스 내에 통합되면 문서를 다른 데이터와 같이 질의할 수 있고 동시성, 복구 제어 등의 많은 이점을 제공할 수 있으나[8], 기존의 연구들은 SQL 에서는 제공되지 않는 XML 문서에 대한 질의 수행을 위해 SQL 과 독립된 질의 언어를 정의하고 있다. 이들 질의어는 XML 문서에 대한 효율적인 검색을 지원하기 위한 구조적 검색 기능을 제공하고 있지만 데이터베이스에 대한 질의어인 SQL 과 독립되어 있기 때문에 데이터베이스와 통합된 질의 수행에 많은 제약사항이 따른다.

본 논문에서는 XML 문서와 데이터베이스의 통합된 질의 수행을 제공하기 위하여 SQL 를 확장한다. XML 문서를 저장하고 질의하기 위하여 제안한 질의어는 XML 을 구성하고 있는 엘리먼트를 자료형으로 선언하기 위한 DDL 구문을 정의하고 이들 자료형에 대한 연산이 가능하도록 SQL 의 DML 을 확장함으로써 XML 문서와 데이터베이스의 통합된 질의를 가능하게 한다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련연구를 살펴보고 3 장에서는 제안한 SQL 의 확장에 관하여 논한다. 끝으로 4 장에서는 결론 및 향후 연구방향에 대해 논한다.

2. 관련연구

2.1 XML

XML 은 웹을 기반으로 하는 구조화된 문서를 기술하는 방법에 대한 표준이다. XML 로 기술된 데이터 객체를 XML 문서라고 한다. XML 문서는 프롤로그(Prolog)와 도큐먼트 엘리먼트(Document Element)로 구분된다. 프롤로그에는 XML 문서임을 나타내는 선언문과 문서의 문법 규칙을 나타내는 DTD 로 구성되어 있다. DTD 에서는 도큐먼트 엘리먼트에 나타나는 각각의 엘리먼트와 해당 엘리먼트에 대한 속성, 엔티티 정보등에 대한 정의를 하며 각각의 엘리먼트의 논리적인 구조를 정의한다. XML 에 있어서 DTD 는 생략이 가능한 것으로 이러한 XML 문서를 well-formed XML Document, DTD 의 구조를 모든 도큐먼트 엘리먼트의 모든 엘리먼트가 만족하는 문서를 valid XML Document 라고 한다. 도큐먼트 엘리먼트에는 XML 문서의 실제 정보를 담고 있는 것으로 태그(Tag)로 둘러싸인 엘리먼트들이 중첩되어 나타난다.

2.2 T/RDBMS 와 SQL/MM Part 2

T/RDBMS 는 SGML 을 저장하기 위하여 관계형 데이터베이스 시스템과 문서 정보 시스템을 이용한 멀티데이터베이스 시스템이다. T/RDBMS 는 문서의 메타 정보와 일반 데이터는 관계형 데이터베이스에 저장하며, SGML 문서 정보는 문서 정보 시스템에 저장하는 방식을 취하고 있다[4]. 이 시스템은 데이터베이스 시스템과 문서 정보 시스템의 상위 레벨에서 질의를 통합하여 처리하는 기능을 제공하고 있지만 하위 레벨의 저장 시스템의 분리로 인해 문서 정보에 관한 질의와 일반 데이터의 통합 질의를 수행하기에는 적합하지 않으며 구문상으로도 지원하지 않는다.

SQL/MM Part 2 는 SGML 과 같은 구조화된 문서를 지원하기 위한 새로운 ADT(Abstract Data Type) 를 추가함으로써 기존의 관계형 모델의 데이터베이스에서 SGML 문서의 지원을 가능하게 한다[6]. 그러나 제안된 ADT 는 ADT 와 관련된 연산들이 T/RDBMS 에서의 연산들을 계승하고 있으며 연산들이 이 자료형에 국한되어 통합 질의를 수행하기에는 적합하지 않다. 또한, SQL/MM Part2 는 SGML 과 같은 구조적 문서를 지원하기 위한 추상화된 타입의 정의를 하고 있기 때문에 XML 을 표현하기에는 제한이 따른다.

3. XML 을 지원하기 위한 SQL 의 확장

데이터베이스와 XML 문서의 통합 질의를 수행하기 위해서는 데이터베이스 시스템은 XML 문서를 내부의 자료형으로 인식할 필요가 있다. SGML 에 바탕을 둔 기존의 연구에서는 문서 자료형을 제공하여[4][6] 데이터베이스 시스템이 SGML 문서에 대한 연산을 수행할 수 있도록 하였으나 연산의 단위가 문서로 제한되어 문서자체에 대한 저장과 검색만을 제공하는 등 질의 수행 능력에 제한을 받는다는 단점이 있었다.

본 논문에서는 XML 문서의 엘리먼트를 자료형으로 정의하고 기본 자료형과 연산이 되도록 함으로써 XML 문서와 데이터베이스 간의 통합 질의를 가능하도록 한다.

3.1 SQL 의 DDL 의 확장

본 절에서는 SQL 에서 XML 문서의 엘리먼트(Element)를 자료형으로 정의하기 위한 DDL 을 확장한다. [그림 1] 은 확장된 DDL 의 BNF 형태이다. 엘리먼트 자료형은 XML 문서의 DTD 에서 정의되는 엘리먼트를 정의하기 위한 것으로 구조 정보는 엘리먼트의 연산에 사용되는 제약조건으로 엘리먼트의 구조 정보가 유지되도록 한다. 기존의 XML 문서는 네임 스페이스와 함께 엘리먼트를 생성할 수 있다.

CREATE ELEMENT 문에서는 EXTERNAL DTD 와 같은 형식은 데이터베이스 내에 제약조건을 명식 할 수 없기 때문에 지원하지 않는다.

```
CREATE ELEMENT <ELEMENT name>
[ AS { <Content Model> | <Contents> | <Select Clause> | 'xml document'
NAMESPACE <namespace> } ]
[ IN <Attribute definition list> ]
[ { NULL | VARIANT } ]

<Content Model> ::= (choice | seq) ('?' | '*' | '+')?
cp ::= (Name | choice | seq) ('?' | '*' | '+')?
choice ::= '(' S? cp ( S? '|' S? cp )* S? ')'
seq ::= '(' S? cp ( S? ',' S? cp )* S? ')'
<Contents> ::= <string>
<Select Clause> ::= <SELECT ... FROM ... WHERE Clause in SQL>

<Attribute definition list> ::= <Attribute name> <Attr-Type>
[ <Default> ]
<Attr-Type> ::= { <Enumerate> | (<ENTITY name>)+ | <string> | <xml
Extend type> }
<Enumerate> ::= '(' <string> ( '|' <string>)* ')' DEFAULT <string>
<Default> :: = { NOT NULL | NULL | CONST <string> | DEFAULT
<string> }

<xml Extend type>:: = xml:xmltype <typename> <typestring>
<typename> ::= <string>
<typestring> ::= <string>
```

[그림 1] CREATE ELEMENT

아래 [그림 2] 는 ENTITY 를 정의하기 위한 구문이다.

```
CREATE ENTITY <ENTITY name> AS <Entity Definition> CONST
<Entity Definition> ::= <string>
```

[그림 2] CREATE ENTITY

이미 정의된 엘리먼트의 구조 정보를 변경하기 위한 구문으로 ALTER ELEMENT 를 지원한다. ALTER ELEMENT 는 구조 정보만을 변경할 수 있으며 속성이나 내용에 대한 변경은 불가능하다. 이에 대한 구문은 아래 [그림 3] 과 같다.

```
ALTER ELEMENT <ELEMENT name>
[ AS { <Content Model> | <Select Clause> } ]
[ { NULL | VARIANT } ]
```

[그림 3] ALTER ELEMENT

기 생성된 엘리먼트를 제거하기 위한 확장된 구문은 아래 [그림 4] 와 같다.

```
DROP ELEMENT <ELEMENT name>
```

[그림 4] DROP ELEMENT

다음의 [그림 5] 는 XML 의 DTD 의 한 예를 보여주고 있으며, [그림 6] 은 [그림 5] 에서 나타난 XML 의 DTD 를 CREATE ELEMENT 구문을 이용한 질의다.

```
<!DOCTYPE EMAIL [
        <!ELEMENT EMAIL ( TO, FROM, CC, SUBJECT, BODY)>
        <!ATTLIST EMAIL LANGUAGE
(Western|Greek|Latin|Universal) "Western">
        <!ELEMENT TO (#PCDATA)>
        <!ELEMENT FROM (#PCDATA)>
        <!ELEMENT CC (#PCDATA)>
        <!ELEMENT SUBJECT (#PCDATA)>
        <!ELEMENT BODY (#PCDATA)>
]>
```

[그림 5] XML 의 DTD 예

```
CREATE ELEMENT TO AS string
CREATE ELEMENT FROM AS string
CREATE ELEMENT CC AS string
CREATE ELEMENT SUBJECT AS string
CREATE ELEMENT BODY AS string
CREATE ELEMENT EMAIL AS (TO,FROM,CC,SUBJECT,BODY)
    IN LANGUAGE (Western|Greek|Latin|Universal) DEFAULT Western
```

[그림 6]. XML DTD[그림 5]에 대한 질의

다음은 'XML1.xml' 이라는 문서를 바탕으로 엘리먼트를 정의하는 구문이다.

```
CREATE ELEMENT XML1 AS 'XML1.xml' NAMESPACE xml1
```

구조 정보를 가지지 않는 엘리먼트는 DTD 를 갖지 않는 XML 문서의 경우이다. 이러한 엘리먼트에 대해서는 어떠한 구조 제약조건도 존재하지 않는다. 이러한 엘리먼트는 VARINT 속성을 갖는 엘리먼트와 동일하며 다음의 3 개의 구문은 모두 동일한 의미를 갖는다.

```
<!ELEMENT TEST ANY>
```

```
CREATE ELEMENT TEST
```

```
CREATE ELEMENT TEST VARIANT
```

3.2 엘리먼트의 연산

3.2.1 기존 자료형과의 연산

엘리먼트는 UDT(User Defined Type)과 마찬가지로 SQL 에서 확장된 자료형으로 정의되었기 때문에 필드를 정의하는 자료형으로 사용할 수 있으며 엘리먼트 만으로 테이블을 정의할 수 있다.

엘리먼트는 다음의 연산 규칙을 따른다.

- 구조 정보를 갖는 엘리먼트는 동일한 구조를 갖는 엘리먼트나 릴레이션과 연산을 수행한다.
- 단말 엘리먼트는 구조 정보를 갖는 엘리먼트에 추가와 삭제가 가능하다.
- 기본 자료형은 단말 엘리먼트의 내용에 대해서만 연산이 수행된다.

3.2.2 엘리먼트의 구조 탐색

엘리먼트는 기존의 '.' 연산자만으로는 효과적으로 엘리먼트의 구조를 탐색할 수 없기 때문에 XQL[2]을 기반으로 한 구조 탐색 구문을 정의한다. [그림 7] 은 엘리먼트에 대한 구조를 탐색하는 구문을 보여주고 있다.

```
<pattern> :: = { <path> | '(' <path> ')' } [ <pattern> ]
<path> ::= [ '^' ] <elements> [ { '/' | '//' | ';' | ';;' } <pattern> ]
<elements> ::= { <element> | '@' <attribute name> [ '[' <filter> ']' ] }
<element> ::= <element name> [ { '@' <attribute name> | '[' <filter> ']'
| '?' | '??' | '*' } ]
<filter> = { <integer> | <elements> }
```

[그림 7] 엘리먼트 구조 탐색

3.2.3 엘리먼트 구조 탐색의 예

invoice 첫번째 내부 단계에 나타나는 products.

```
invoice/product
```

invoice 내부에 나타나는 products.

```
invoice//product
```

prod_name 속성이 "smoke" 인 product.

```
product[@prod_name="smoke"]
```

"Shady Grove"를 포함하는 TD 와 그것을 바로 뒤따르는 TD.

```
TD = "Shady Grove" ; TD
```

SPEAKER 를 포함하는 상위 엘리먼트.

```
^SPEAKER
```

3.3 SQL 의 DML 의 확장

3.3.1 검색 질의

[그림 8] 는 엘리먼트 자료형이 검색 질의가 참여하기 위해 확장된 SELECT 문의 BNF 형태이다.

EXPORT 구문은 질의 결과를 XML 문서로 결과를 출력하며 문자열 자료형이므로 더 이상의 엘리먼트 연산은 수행할 수 없다. 구조 정보를 갖지 않는 엘리먼트에 대한 질의어는 반드시 DYNAMIC 지시어를 사용해야 한다.

```
SELECT [ DISTINCT | ALL ]
{ Column expression [ AS name] | <ELEMENT expression> } [ ,...] | *
FROM <Table reference> [ {,<Table Reference> } ... ]
[ WHERE search condition ]
[ GROUP BY Columns [ HAVING condition ] ]
[ ORDER BY <sort specification list> ]

<Table Reference> ::=
[ ONLY ] { <Table name> | <query name> } [ [ AS ] <Correlation name>
        [ (<derived Column list>) ] ] |
<Table subquery> [ AS ] <Correlation name>
        [ (<derived Column list>) ] |
<joined Table> |
LITERAL (<query expression>) [ AS ] < Correlation name>
        [ (<derived Column list>) ] |
<ELEMENT expression> ]

<ELEMENT expression> ::=
{ [DYNAMIC] <ELEMENT Pattern> [ AS name ] [ EXPORT name ]
```

[그림 8] SELECT 구문

Author 가 'Kim' 인 Book 엘리먼트를 찾아서 XML 문서 XML1.xml 을 출력하라.

```
SELECT B? EXPORT XML1.xml
FROM Books AS B
WHERE B/Book/Author = 'Kim'
```

Books 테이블을 이용하여 Book 엘리먼트의 형태로 XML 문서를 출력하라.

```
SELECT Book? EXPORT XML2.xml
FROM BOOKS
WHERE Author = 'KIM'
```

이 예제는 결과 릴레이션을 Book 엘리먼트의 구조 정보를 이용하여 XML 결과 문서를 생성한다.

3.3.2 변경 질의

(1) INSERT

INSERT 구문은 기존의 엘리먼트에 새로운 엘리먼트를 추가하는 연산을 수행한다. [그림 9] 는 INSERT 구문을 정의하고 있다.

```
INSERT INTO <Table Name> [ <column list> ]
{ <query expression> | VALUE <column expression > } |

INSERT INTO <ELEMENT Pattern>
{ <query expression> | VALUE <column expression > }

<column expression> ::= { <column list> | <ELEMENT list> | DEFAULT }
<ELEMENT list> ::= [ ( <ELEMENT Pattern> [ {,<ELEMENT Pattern> }
... ] ) ]
<column list> ::= [ (<Column name> [ {,<Column name> } ... ] ) ]
```

[그림 9] INSERT 구문

Author 엘리먼트와 Date 엘리먼트를 Book 엘리먼트에 추가

```
INSERT INTO BOOKS/Book VALUE ( Author , Date )
```

저자가 'Kim' 인 책의 저자명(Author)와 발행일(Date)를 Book 엘리먼트에 추가

```
INSERT INTO BOOKS/BOOK
Select Author,Date from BOOK as B where B.Author = 'KIM'
```

(2) UPDATE

UPDATE 연산은 기존의 엘리먼트에 대한 내용 변경을 수행한다. UPDATE 는 기존의 엘리먼트 구조에 대한 변경은 허용하지 않으며 엘리먼트 구조의 변경을 수행하기 위해서는 ALTER ELEMENT 문을 이용하여야 한다. [그림 10] 는 UPDATE 구문을 정의하고 있다.

```
UPDATE <Table name> SET
<Column name>=scalar_expression
[ { , <Column name>=scalar_expression} ... ] |
 [ WHERE <search condition> ] |

UPDATE <ELEMENT Pattern> SET <ELEMENT Pattern> =
{ <Column name> | scalar_expression }
[ WHERE { <ELEMENT Pattern> | <search condition> }
```

[그림 10] UPDATE 구문

(3) DELETE

DELETE 문은 엘리먼트로부터 조건에 부합하는 엘리먼트를 삭제하는 연산을 수행한다. [그림 11] 은 DELETE 구문을 정의하고 있다.

```
DELETE FROM <Table name>
[ WHERE <search condition> ] |

DELETE FROM <ELEMENT Pattern>
[ WHERE <ELEMENT Pattern > ]
```

[그림 11] DELETE 구문

4. 결론

데이터베이스에 구축되어 있는 방대한 양의 정보를 인터넷 상에서 효율적으로 이용하기 위해서 데이터베이스 시스템은 XML 문서에 대한 지원이 가능해야 한다. 기존의 연구들은 XML 문서의 질의를 위하여 SQL 과는 독립된 언어를 제공하여 XML 문서와 데이터베이스 간의 통합된 질의는 제공하지 못한다.

본 논문에서는 SQL 의 DDL 을 확장하여 XML 문서의 엘리먼트를 자료형으로 정의하였으며 DML 을 확장하여 이 자료형을 바탕으로 하여 연산을 수행하는 질의어를 정의하였다. 제안된 질의어는 SQL 문으로 XML 문서에 대한 저장과 질의가 가능하여 데이터베이스와 통합된 질의가 가능하다.

향후 본 논문에서 제안된 질의어를 효율적으로 처리하기 위한 질의 처리기의 설계 및 구현에 관한 연구와 저장 관리자에 관한 연구가 요구된다.

참고문헌

[1] eXtensible Markup Language(XML) 1.0, "http://www.w3.org/TR/PR-xml-971208".

[2] XML Query Language (XQL), "http://www.w3.org/TandS/QL/QL98/pp/xql.html".

[3] A. Deutsch, M. Fernandez, D. Suciu, "Storing Semistructured Data with STORED," SIGMOD '99 Philadelphia PA.

[4] G. E. Blake, M. P. Consens, P. Kilpelainen, P.-A. Larson, T. Snider, and F. W. Tompa, "Text/Relational Database Mangement Systems: Harmonizing SQL and SGML," Proc. Application of Databases (ADB 94), Vadstena, Sweden (June 1994), Lecture Notes in Computer Science 819, Springer-Verlag, pp. 267-280.

[5] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, J. Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities," Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999.

[6] L.J.Brown, M.P.Consens, I.J.Davis, C.R.Palmer, and F.W.Tompa, `A Structured Text ADT for Object-Relational Databases,' in `Objects, Databases, and the WWW,' a special issue of Theory and Practice of Object Systems, Vol. 4, No.4 (1998) 227-244.

[7] R. Sacks-Davis, T. Arnold-Moore and J. Zobel, "Database System for Structured Documents," In International Symposium on Advanced Database Technologies and Their Integration, pp. 272-283, Naran Japan, 1994.

[8] 허명식, 손기락, "XQL 을 지원하는 XML 문서 저장 시스템," '99 가을 학술발표 논문집 제 26 권 2 호, PP39-41.