XML 데이터베이스의 능동적 검증 기법*

[0]김상균*, 전희영*, 이명철*, 이경하*, 이규철*, 이미영**, 손덕주**
*충남대학교 컴퓨터공학과 데이터베이스연구실
**한국전자통신연구원
{skkim, hyjun, mclee, bart, kclee}@ce.cnu.ac.kr
{mylee,djson}@dbserver.etri.re.kr

An Active Validation Mechanism for XML Databases

[0]Sang-Kyun Kim*,Hui-Young Jun*,Myung-Cheol Lee*, Kyong-Ha Lee*, Kyu-Chul Lee*
Duk-Joo Son**, Mi-Young Lee**
*Dept. of Computer Engineering, Chungnam National University
**Electronics and Telecommunications Research Institute

요 약

XML[1]은 문서의 논리적인 구조를 가지고 있으며 XML문서를 파싱할 때 이 구조에 맞는지 검증을 하게 된다. 이 때 대부분 파서의 경우에는 문서 단위로 검증을 하며 문서의 일부분만 검증할 수 없다. 또한 XML 문서가 변경되었을 때 이 문서가 유효(valid)한지 검사할 때에도 문서 전체를 검증해야 한다. 이렇게 되면 검증할 필요가 없는 부분도 다시 검증하게 되는 오버헤드가 발생하는데 만약 XML문서가 데이터베이스에 저장되어 있다면 문서 전체를 꺼내어서 검증하고 다시 삽입해야 한다. 본 논문에서는 이러한 문제점을 해결하기 위하여 XML문서가 변경되었을 때, 변경된 부분만 검증할 수 있는 기법을 제안한다.

1. 서론

1998년 W3C(World Wide Web Consortium) 에서 권고안으로 채택된 XML(eXtensible Markup Language)은 차세대 인터넷 전자 문서 표준으로서, 기존의 문서 포맷과는 달리 하나의 문서에서 내용 정보와 함께 문서의 구조 정보를 함께 가지고 있다. 이러한 정보는 DTD(Document Type Defin ition)에서 정의되는데, XML문서를 파싱할 때 파서는 이 정보를 이용해서 XML문서가 유효한지 검증을 하여 올바르지 않으면 어디에 오류가 있는지 알려준다.

하지만 일반적인 파서에서의 검증은 문서 단위로 이루어지며 문서의 일부분을 파싱하고 브라우징할 수는 없다. 또한 어플리케이션에서 문서를 변경하였을 때 변경된 XML문서가 유효한지를 검사하려면 전체 문서를 다시 파싱해야 하기 때문에 비효율적이다. 더구나 데이터베이스에 저장되어 있는 XML문서가 변경되었을 때, 이를 검증하기 위해서는 문서 전체를 꺼내어 파싱하고 유효성을 검사한 후 다시 저장해야만 한다.

*본 연구는 한국전자통신연구원의 " 멀티미디어 문서 모델링 도구 개발(위탁계약번호: 00-085)" 과제의 일부로 수행된 결과임

따라서 본 연구에서는 데이터베이스에 저장된 XML문서를 변경할 때 능동적으로 검증하는 방법을 설계하고 구현함으로서 보다 효율적인 XML저장 시스템을 보이고자 한다.

2. 관련연구

2.1 부분 검증 지원 시스템

현재 대부분의 XML저장 관리 시스템들은 XML문서를 변경할 때 유효한지 검사하지 않는다[2,3]. 대신 변경을 한 후 변경된 XML문서를 읽어들일 때 문서 전체에 대해서 유효성을 검사한다. 반면에 XML 파서중에서 IBM의 XML4J 파서[4]는 이와 비슷한 기능을 지원한다. XML4J는 두 가지 종류의 DOM(Document Object Model)[5] 을 가지고 있는데 하나는 표준 DOM으로 W3C에서 제안한 DOM Level 1을 따르고 다른 하나는 TX Compatibility DOM으로 표준 DOM에 서브 트리 검증 기능과 다른 여러 가지 기능이 추가된 것이다.

TX Compatibility DOM 에서는 XML 문서의 서브 트리를 검증하기 위해서 결정적 유한 오토마타(Deterministic Finite Automata)[6]을 참조하였다. 만약 검증할 서브 트리가 주어지면 트리를 순회하면서 결정적 유한 오토마타를 구성하여 자식 노드들을 검증하게 된다.

2.2 결정적 유한 오토마타

유한 오토마타는 일반화된 전이 다이어그램에 의해서 정규식을 인식하는 것으로 비결정적 유한 오토마타와 결정적 유한 오토마타가 있다. 두 가지 모두 정규 집합을 정확히 인식할 수 있지만 다음과 같은 시간 공간 사이의 차이가 있다. 결정적 유한 오토마타는 비결정적 유한 오토마타보다 크기가 크지만 빨리 인식할 수 있다.

결정적 유한 오토마타는 어떤 입력에 대해서도 각 상태로부터의 전이는 최대 하나만 가진다. 만약 결정적 유한 오토마타의 전이 함수를 나타내기 위해 전이 테이블을 사용한다면 전이 테이블에서 각 항목은 하나의 상태이다. 결과적으로 결정적 유한 오토마타에는 시작 상태로부터 문자열에 의해 표시되는 경로가 최대 하나가 있기 때문에 이 입력 문자열을 받아들이는지를 결정하는 것이 쉽다.

3. XML 문서 검증기의 설계

본 연구는 W3C에서 제안한 DOM Level 1 을 따른다. 따라서 XML 문서에 대한 변경 연산은 DOM에서 정의된 것을 지원하며, 변경에 대한 유효성 검사도 이 함수들에 대해서 수행한다.

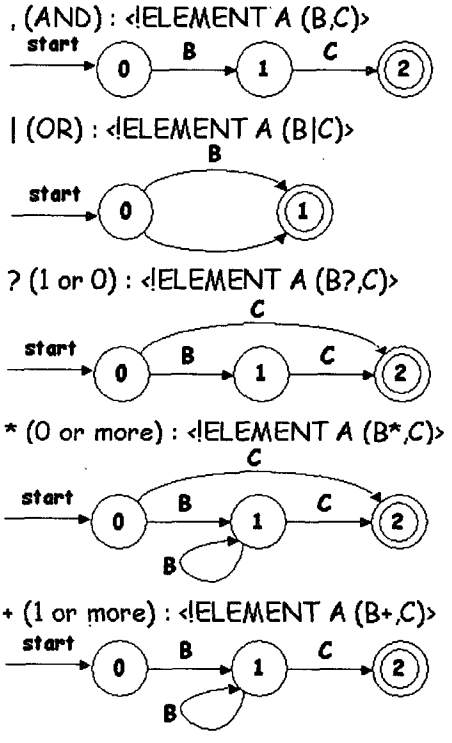
3.1 DTD 정보 저장

XML문서가 유효한지 검사하려면 DTD를 보고 판단해야 하므로 DTD를 파싱해서 저장해야 한다. DTD는 XML문서가 파싱되기 전에 파서에 의해 먼저 처리되기 때문에 XML문서를 저장할 때 파서에서 DTD정보를 추출하여 저장할 수 있다. 하지만 하나의 DTD를 가지는 XML문서들을 저장할 때마다 DTD에서 정보를 추출할 필요가 없으므로 첫 번째 XML문서를 저장할 때만 DTD를 정보를 추출한다. DOM Level 1에서는 엘리먼트와 어트리뷰트만 변경이 가능하다. 따라서 추출된 정보도 이 두가지 선언들로 구성되며 추출한 정보를 가지고 변경시 유효성 검사를 효율적으로 하기 위해서는 DTD 정보를 꺼낼 때 어떠한 형태로 추출하느냐가 중요하다. 이를 위해서 본 논문에서는 하나의 엘리먼트 선언마다 하나의 결정적 유한 오토마타를 구성한다. 이것을 이용하면 하나의 엘리먼트 선언이 하나의 정규식으로 이루어지기 때문에 변경이 일어났을 때 쉽게 유효성 검사를 할 수가 있다. 그리고 어트리뷰트 선언은 엘리먼트 이름, 어트리뷰트 이름, 어트리뷰트 값, 어트리뷰트 타입, 디폴트 타입, 디폴트값으로 나누어서 해당 컬럼에 저장한다.

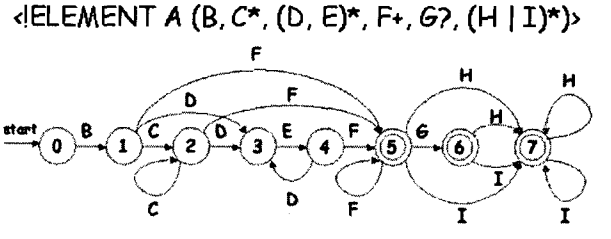
3.2 결정적 유한 오토마타 구성

엘리먼트를 선언할 때, ' ,' (and), ' |' (or), ' ?' (1 or 0), ' *' (0 or more), ' +' (1 or more) 이렇게 5가지 연산자를 이용한다.

엘리먼트 선언을 하나의 정규식으로 표현하기 위해서 위 연산자 각각의 경우마다 하나의 식으로 만들고 연산자가 나올 때마다 그것들을 조합하여 구성한다. 그 외에 괄호는 재귀적으로 처리하며, ' +' 연산자로 선언되었을 때, 엘리먼트의 삭제여부를 판단하기 위해서 마지막 상태(final state)를 가진다. [그림 1]은 각 연산자에 대한 식을 나타낸 것이고 [그림 2]는 하나의 엘리먼트 선언에 대해 이 5가지 연산자들을 조합해서 표현한 것이다.



[그림 1] 연산자에 대한 결정적 유한 오토마타 표현



[그림 2] 하나의 엘리먼트 선언에 대한 결정적 유한 오토마타 표현

[표 1]은 [그림 2]를 데이터베이스 테이블에 저장하는 예제이다. 그리고 [표 2]는 두 가지 어트리뷰트 선언을 테이블에 저장하는 예제이다. 만약 엘리먼트 선언이 "EMPTY"또는 "ANY"로 선언되어 있다면 tranName 어트리뷰트 값도 "EMPTY"와 "ANY"로 선언된다.

[표 1] Element 저장 예제

dtd_id	element Name	before State	tran Name	after State	final State
AAAAA	A	0	B	1	0
AAAAA	A	1	C	2	0
AAAAA	A	1	D	3	0
AAAAA	A	1	F	5	1
AAAAA	A	2	C	2	0
...	...	...	...	...	...

[표 2] Attribute 저장 예제

<!ATTLIST eg xml:space (default|preserve) #FIXED "preserve">

<!ATTLIST loc href CDATA #REQUIRED>

dtd_id	element Name	attr Name	attr Type	attr Value	default Type	default Value
AAAAA	eg	xml:space	Enumeration	default|preserve	FIXED	preserve
AAAAA	loc	href	CDATA		REQUIRED	

3.3 유효성 검사

[표 1] 과 [표 2] 와 같이 DTD 정보를 저장하면 변경 연산이 일어났을 때, 이 정보를 보고 유효성 검사를 할 수 있다. 엘리먼트의 경우 변경하는 노드의 바로 전 노드와 바로 다음 노드가 데이터베이스에 저장되어 있는지 검사하면 되고, 어트리뷰트의 경우에도 데이터베이스에 저장된 어트리뷰트가 있는지 질의하면 된다.

4. XML 문서 검증기의 구현

본 논문에서 제시한 방법은 DOM 을 사용자 인터페이스로 가지는 XML 문서 저장 검색 시스템[7,8]에 적용되었다. 이 시스템은 데이터베이스로 오라클 8.0.6를 사용하고, 파서는 IBM 의 XML4C 2.3.1[9] 을 사용하였다.

4.1 엘리먼트 검증

유효성 검사는 엘리먼트의 삽입, 삭제 연산에 대해 수행된다. 다음은 삽입과 삭제의 경우에 필요한 유효 조건들이다.

4.1.1 엘리먼트 삽입

1. DTD에 정의되어 있는가
2. 엘리먼트의 어트리뷰트가 "REQUIRED"로 선언되어 있는가
3. 부모 엘리먼트의 내용 모델이 "ANY"인가
4. 삽입할 곳 바로 전 엘리먼트의 뒤에 새 엘리먼트가 삽입될 수 있는가
5. 삽입할 곳 바로 다음 엘리먼트의 앞에 새 엘리먼트가 삽입될 수 있는가

4.1.2 엘리먼트 삭제

1. 부모 엘리먼트의 내용 모델이 "ANY"인가
2. 마지막 자식이면 내용 모델이 "+"인가
3. 삭제할 엘리먼트의 이전 엘리먼트 뒤에 삭제할 엘리먼트 다음 엘리먼트가 나올 수 있는가

4.2 어트리뷰트 검증

어트리뷰트는 해당 타입이 데이터베이스에 저장되어 있는지를 질의하면 된다.

4.1.1 어트리뷰트 변경

1. 디폴트 타입이 FIXED 또는 REQUIRED인가
2. 어트리뷰트 타입이 올바른가

4.2.2 어트리뷰트 삭제

1. 디폴트 타입이 FIXED 또는 REQUIRED 또는 DEFAULT인가
2. 어트리뷰트 타입이 ID 일 때 IDREF 가 존재하는가

5. 결론

본 연구에서는 데이터베이스에 저장된 XML 문서의 일부분이 변경되었을 때 능동적으로 검증하는 방법을 제안하였다. 이 기법은 문서 전체를 파싱하지 않고 변경된 부분만 검증할 수 있도록 하며, 유효한 변경 연산들만 수행하도록 한다. 따라서 일부분만 유효성을 검사하므로 검증의 오버헤드를 줄일 수 있으며, 변경 후에도 XML 문서의 유효성을 유지할 수 있다.

6. 참고 문헌

[1]. " XML(eXtensible Markup Language)" ,http://www.w3.org/TR/REC-xml
[2]. POET Content Management System, " http://www.poet.com"
[3]. eXcelon, " http://www.exceloncorp.com"
[4]. " XML For Java" , http://www.alphaworks.ibm.com
[5]. " DOM(Document Object Model) Level 1" , http://www.w3.org/TR/REC-DOM-Level-1/
[6]. Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, " Compilers principles, Techniques, and Tools"
[7]. 연제원, " XML문서의 효율적 검색 및 변경을 위한 저장관리기의 설계 및 구현" , 석사 학위 논문, 2000
[8]. 연제원, 조정수, 이강찬, 이규철, " XML문서 구조검색을 위한 저장 시스템 설계" , ' 99 한국정보과학회 봄 학술 발표 논문집(B), 26권 1호, pp.185-187, 1999
[9]. " XML For C++" , http://www.alphaworks.ibm.com