XML Schema 갱신 시의 유효성 유지에 관한 연구

탁성수° 최윤진 이언배
한국방송통신대학교 평생대학원 정보과학과
graynote°@freechal.com yj93071@hanmail.net lub@mail.knou.ac.kr

The Study on Preserving XML Schema Validation during Updating Process

Sungsu Tark° Yunjin Choi Eunbae Lee
Dept. of Computer Science, Korea National Open University

요 약

XML 문서를 저장할 때 XML 데이터의 구조적인 검증을 위해서 DTD나 XML Schema를 같이 저장하는 방식이 사용된다. 그러나 스키마의 갱신이 발생할 경우는 기존 저장방식에서는 그 자체에 유효성을 검증하기 위한 메커니즘을 가지고 있지 않기에 파일로 추출하여 수정후 XML Parser로 검증하고 다시 저장하는 비효율적인 과정을 거쳐야 하였다. 또한 스키마의 변경에 따라 이와 연계된 XML 데이터도 유효성을 보장할 수 없는 문제점이 발생하였다. 본 논문에서는 DBMS에 저장되어 있는 XML Schema를 갱신 할 때 파일로 추출할 필요없이 SQL문에 의해 갱신이 되는 그 자체로 유효성을 보장할 수 있고, 이와 연결된 XML 데이터의 유효성 검증이 같이 이루어질 수 있는 효율적인 갱신기법을 설계한다. XML Schema 검증은 관계형 데이터베이스의 여러 제약 조건들에 의해서 이루어지고, XML 데이터는 스키마 갱신 내용에 따라 검증작업 필요성을 먼저 확인 후에 각 스키마 타입에 따라 유효성을 확인하는 갱신이 이루어지도록 설계되어 불필요한 작업에 의한 오버헤드를 사전에 방지하였다.

1. 서 론

XML 문서의 활용도가 높아짐에 따라 질의 및 검색을 위해서 데이터베이스에 저장하는 방식이 많이 일반화되고 있다[1][2]. 또한 근래에는 XML 데이터의 변경이 이루어지는 경우에 같이 저장되어 있는 DTD 및 XML Schema에 의한 유효성을 검증하는 방법에 대한 연구가 차츰 진행되고 있다[3][4]. 하지만 이러한 시스템에서는 DTD 및 XML Schema 자체의 변경이 있을 경우는 자체 검증수단이 없어서 다시 XML Schema 파일로 추출하여 XML Parser로 재 검증을 하지 않는 이상 유효성을 보장할 수 없고, 또한 기존에 저장되어 있는 XML 데이터에 대해서도 갱신된 스키마에 대해서 유효성의 지속여부를 확인할 수 없는 한계점을 가지고 있다.

본 논문에서는 XML Schema 갱신 시 데이터베이스에 SQL문에 의해서 갱신되는 것 자체로 유효성이 보장되는 저장스키마를 설계하고, 스키마 변경 작업 시 기존에 저장되어 있는 XML 데이터의 검증작업 필요성에 대해 먼저 확인절차를 거친 후, 필요한 경우로 판정된 경우에만 유효성 검증절차를 진행하는 효율적인 갱신방법을 제안한다.

본 서론에 이어 2장에서는 DTD 및 XML Schema 저장 및 XML 데이터 유효성 검증에 관련된 연구를 소개하고, 3장에서는 본 논문에서 제안하는 XML Schema 저장 모델 및 스키마 타입별 검증절차의 갱신기법을 제시하고, 상용 데이터베이스인 SQL Server 2000에 적용한 사례를 보여준다. 4장에서는 본 논문의 결론과 향후 연구과제에 대해 언급한다.

2. 관련 연구

XML 데이터의 검증을 위해서 DTD나 XML Schema를 같이 저장하는 방식이 많이 연구되고 있다. 그 중 가장 단순한 방식은 네임스페이스에 따른 저장으로 XML Schema의 targetNamespace 별로 저장하되 스키마 전체를 하나의 스트링으로 저장하는 것이다[3]. 이러한 저장 방식에서의 XML Schema의 갱신은 스키마와 XML 데이터 전체 내용을 파일로 추출하여 검증 절차 후 다시 저장해야 하는 단점이 있다.

XML 저장방식인 Edge approach[1]를 이용한 방법은 XML Schema를 element, complexType, attribute 와 같은 엘리먼트들과 name, type 과 같은 애트리뷰트들을 각각 엘리먼트 테이블과 애트리뷰트 테이블에 저장하는 것이다. 하지만 이러한 저장상태에서는 XML Schema의 갱신 시 엘리먼트와 애트리뷰트 테이블 전체를 읽어서 조합하는 작업을 거쳐서 XML Schema 파일로 생성하여 검증하고, XML 데이터도 비슷한 과정을 거쳐야 하는 문제점이 있다.

XML Schema의 추출시의 효율성을 고려한 최근 연구에서는 엘리먼트 이름별로 테이블을 생성하는 기본 Binary approach[1]에서 각각의 엘리먼트 별로 관련된 일부 엘리먼트들과, 애트리뷰트들을 해당 엘리먼트 테이블에 인라인닝하고, XML 데이터 갱신에 대한 유효 검증 시 사용 될 스키마 정보를 Pathtable기법을 통해 추출하는 방법을 사용한다[3]. 그렇지만 이 연구에서는 XML 데이터의 유효성이 유지되는 갱신에 대한 것일 뿐, XML Schema 자체의 변경에 대해서는 자체적으로 검증방법을 가지지 못하므로 유효성을 보장할 수 없고, 스키마 갱신에 따른 XML 데이터 검증에 대한 절차를 가지고 있지 않다.

데이터베이스의 Unique, Check, Not Null, 참조키 등의 여러 제약조건을 이용한 설계방식은 기존에 XML 데이터의 의미론적 제약조건을 유지하기 위한 스키마 의존적인 모델 분야에서 제시된 바 있다[5].

3. XML Schema 저장 및 갱신시의 유효성 유지

본 논문에서는 RDBMS에 저장되어 있는 스키마의 수정 필요시에 유효성 검증을 위해서 파일로 다시 추출할 필요없이 SQL문에 의해 갱신이 되는 그 자체로 유효성을 보장하고, 이와 관련된 XML 데이터에 대해서도 유효성 검증이 같이 이루어지는 스키마 저장모델을 설계한다. XML Schema의 검증과정은 슈퍼타입·서브타입의 구조과 데이터베이스의 여러 제약조건에 의해서 SQL문의 수행으로 이루어진다. 그리고 XML 데이터 유효성 검증 과정은 갱신될 스키마의 범위성이 이전보다 좁아진 경우에만 데이터를 추출하여 변경되므로 불필요한 작업으로 인한

부하를 사전에 방지하였다.

3.1 유효성이 유지되는 XML Schema 저장 모델

스키마 작성규칙에 따르면 전역 엘리먼트는 빈도 횟수를 나타내는 minOccurs 속성과 maxOccurs 속성을 사용할 수 없고, 로컬 엘리먼트에서는 기본값으로 '1'을 가진다. 로컬 엘리먼트에서의 참조는 전역 엘리먼트만이 가능하다. (그림 1)은 이러한 규칙이 있는 Element를 관계형 데이터베이스의 제약조건을 이용하여 설계한 모델이다.

Element 슈퍼타입 테이블

컬 럼	Key	Null	Default	용 도
EId	PK	N		엘리먼트 Id
EName		N		엘리먼트 이름
SCType		N		심플, 콤플렉스 타입구분
GLType		N		전역, 지역 구분
nillable		Y		nillable 여부
SId	FK	Y		SimpleType의 Id
CId	FK	Y		ComplexType의 Id

Global Element 서브타입 테이블

컬 럼	Key	Null	Default	용 도
EId	FK	N		엘리먼트 Id

Local Element 서브타입 테이블

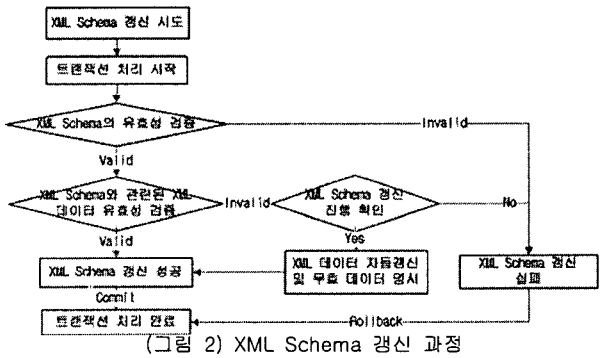
컬 럼	Key	Null	Default	용 도
EId	FK	N		엘리먼트 Id
minOccurs		Y	1	최소 발생횟수
maxOccurs		Y	1	최대 발생횟수
Sord		N		엘리먼트 저장 순서
ref	FK	Y		전역 엘리먼트의 참조 Id
PCId	FK	N		상위 ComplexType의 Id

(그림 1) XML Schema의 Element 테이블 설계

다른 구성요소에 대해서도 Check나 Trigger등의 제약조건을 더 사용한 것을 제외하고는 유사하기에 여기서는 지면관계상 생략한다.

3.2 XML Schema 데이터 갱신

XML Schema의 갱신이 이루어지면, 그 XML Schema와 연결되어 있는 XML 데이터에 대해서도 여전히 유효하다는 것이 보장되어야 한다.



(그림 2) XML Schema 갱신 과정

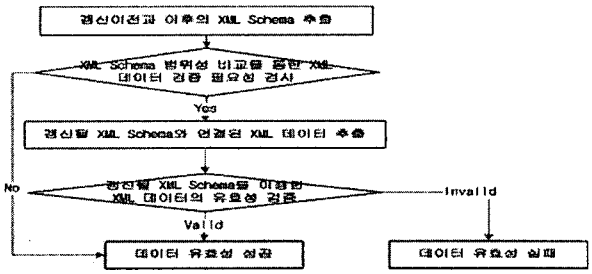
(그림 2)은 이러한 XML Schema의 갱신 절차로서 스키마와 연결된 XML 데이터의 유효성 검증 이후에 사용자 확인에 의해서 완료될 수 있도록 하나의 트랜잭션으로 이루어진다.

(1) XML Schema의 유효성 검사

스키마의 유효성은 공통속성과 고유속성을 구별하기 위한 슈퍼타입·서브타입의 구조과 데이터베이스의 참조 키, Not Null, Default, Check(Rule), Trigger의 여러 제약조건으로 설계된 모델에 XML Schema가 저장되어 있으므로 SQL문에 갱신되면 그 자체로 유효하게 변경되었음을 보장한다.

(2) XML Schema와 관련된 XML데이터의 유효성 검사

아래 (그림3)는 (그림2)의 XML Schema 갱신 절차 중 'XML Schema와 관련된 XML 데이터 유효성 검증'의 상세단계를 보여주고 있다.



(그림 3) XML Data 유효성 검증 절차

XML 데이터의 유효성 검증은 (그림4)의 '범위성에 의한 XML 데이터 검증 필요성 확인 알고리즘'을 이용하여 갱신 이전과 이후의 XML Schema와의 범위성을 확인하는 데, 이전보다 좁아진 경우에는 확인이 필요하다.

```
// XS(E1) : 갱신이전 엘리먼트, XS(E2) : 이후 엘리먼트
// XS(A1) : 갱신이전 애트리뷰트, XS(A2) : 이후 애트리뷰트
// XS(S1) : 갱신이전 심플타입, XS(S2) : 이후 심플타입

Input : XS.object /* XML Schema 갱신 대상 */
Output : CheckFlag /* True : 데이터확인 필요, False : 데이터확인 불필요 */

CheckFlag = False
If (Action = 'Update') Then
      Switch XS.object (
      Case Element :
            If (XS(E1).minOccurs < XS(E2).minOccurs OR XS(E1).maxOccurs >
            XS(E2).maxOccurs )
            Then CheckFlag = True

      Case Attribute :
            If (XS(A1).use > XS(A2).use ) Then  CheckFlag = True (Required < Optional)

      Case SimpleType :
            If (XS(S1).minExclusive(minInclusive, minLength) < XS(S2).
            minExclusive(minInclusive, minLength) OR XS(S1). maxExclusive(maxInclusive,
            maxLength) > XS(S2). maxExclusive(maxInclusive, maxLength)
            Then CheckFlag = True
            If (XS(S1).totalDigits(length) <> XS(S2).totalDigits(length) )
            Then CheckFlag = True
            If (XS(S1).fractionDigits > XS(S2).fractionDigits)  Then CheckFlag = True
            If !(XS(S1).enumeration ⊂ XS(S2).enumeration)  Then CheckFlag = True
      )
Else (Action='Insert' or Action='Delete') CheckFlag = True
```

(그림4) 범위성에 의한 XML 데이터 검증 필요성 확인 알고리즘

그리고 필요한 경우로 판정되면 <표1>의 절차대로 갱신될 XML Schema와 관련하여 검증이 필요한 XML 데이터를 추출하게 된다.

<표 1> XML Schema와 XML 데이터의 매핑 절차

단계	절 차
1	갱신된 엘리먼트(EId) 및 애트리뷰트(AId)의 아이디와 속해있는 컴플렉스 타입 아이디(PCId)를 가져온다.
2	부모 컴플렉스 타입 아이디(PCId)가 컴플렉스 타입의 아이디(CId)인 엘리먼트(EId)및 애트리뷰트(AId) 아이디를 가져온다.
3	가져온 엘리먼트(EId)와 애트리뷰트(AId) 아이디가 XML 데이터의 스키마 엘리먼트(SEId)나 애트리뷰트(SAId) 인 데이터 추출한다.

마지막으로 추출된 XML 데이터를 순환하면서 (그림 5)의 'XML 데이터의 유효성 검증 알고리즘'을 이용하여 검사한다.

```
// XDe(XS(E)) : XML Schema 엘리먼트와 관련된 XML 엘리먼트
// XDa(XS(A)) : XML Schema 애트리뷰트와 관련된 XML 애트리뷰트
// XDae(XS(S)) : XML Schema 심플타입을 데이터 타입으로 가지는 XML 엘리먼트 또는 애트리뷰트

Input : XS.object /* XML Schema 대상 */
Output : Boolean /* True : 유효성 유지, False : 유효성 상실 */

Switch XS.object
      Case Element :
            If (XS(E).minOccurs <= XDe(XS(E)).count <= XS(E).maxOccurs )
              Then XDe(XS(E)).valid = True Else XDe(XS(E)).valid = False

      Case Attribute :
            If (XS(A).use = 'optional' ) Then XS(A).valid = True
            ElseIf (XS(A).use = 'required' and Exist(Xda(XS(A))) ) Then XS(A).valid = True
            Else XS(A).valid = False

      Case SimpleType :
            If (XS(S).minExclusive < XDae(XS(S)).number < XS(S).maxExclusive )
              Then XDae(XS(S)).valid = True  Else XDae(XS(S)) .valid = False
            If (XS(S).minInclusive(minLength) <= XDae(XS(S)).number(length) <=
              XS(S).maxInclusive(maxLength) )
              Then XDae(XS(S)).valid = True  Else XDae(XS(S2)).valid = False
            If (XS(S).totalDigits(Length) == XDae(XS(S)).totalDigits(Length) )
              Then XDae(XS(S)).value = True Else XDae(XS(S)).value = False
            If (XS(S).fractionDigits > = XDae(XS(S)).fractionDigits)
              Then XDae(XS(S)).value = True Else XDae(XS(S)).value = False
            If (XS(S).enumeration ∋ XDae(XS(S)) )
              Then XDae(XS(S)).value = True Else XDae(XS(S)).value = False
```

(그림 5) XML 데이터의 유효성 검증 알고리즘

(3) XML Schema 갱신 진행 확인

XML Schema와 XML 데이터와의 범위성 검증을 통해서 일부의 데이터라도 'Invalid XML'이 된다면 사용자 확인을 통해 계속 진행시킬지의 여부를 확인해야 한다. 그 결정에 따라 XML Schema의 변경작업이 성공(Commit)과 취소(Rollback)가 결정된다. 만일 스키마 갱신으로 인해 무효한 데이터가 발생한다면 다음단계에서 이러한 데이터의 상태를 변경해 주어야 한다.

(4) XML 데이터 자동갱신 및 무효 데이터 명시

스키마 갱신으로 확인되면 시스템에서 자동으로 XML 데이터 갱신이 가능한 부분이 있는지 확인해야 한다. 예를 들어 새로 Attribute가 추가되었는데, use 속성값이 optional이고 default 값이 지정되어 있다면 그 값으로 XML 데이터 갱신이 가능하다. 나머지 데이터는 Invalid 상태 값으로 변경되어 데이터 수정을 요구하게 된다.

4. 스키마 갱신 및 데이터 유효성 실험 및 평가

제안된 모델을 이용하여 저장된 XML Schema를 갱신하고 더불어 연결된 XML 데이터의 유효성을 검증하는 실험을 하였다. 실험대상으로는 XML Schema Part 0: Primer Second Edition[6]에서 제공한 The Purchase Order XML과 XML Schema을 택했다. 실험에 사용된 SQL 쿼리문은 XML Schema를 DOM에서 제공하는 메소드를 통해서 갱신할 때 그 내용을 토대로 생성된 것이다.

```
Update XLocalElement Set Ref = 7 Where EId = 5
```

```
서버: 메시지 547, 수준 16, 상태 1, 줄 1
UPDATE 문이 COLUMN FOREIGN KEY 제약 조건 'FK__XLocalEleme__Ref'과(와)
충돌되었습니다. 충돌은 'XSchema' 데이터베이스, 'XGlobalElement', column 'EId'
테이블에서 발생했습니다. 문이 종료되었습니다.
```

(그림 6) 참조 무결성 위배 실험 결과

먼저 (그림6)과 같이 로컬엘리먼트를 참조하는 변경실험을 하였는데 이는 XML Schema 작성규칙에 어긋하기 때문에 설계된 저장모델에서도 참조제약조건에 의해서 충돌이 발생하였다. 그리고 갱신될 XML Schema에 의해 더 이상 유효하지 않는 XML 데이터들의 상태가 자동 변경되는지 알아보기 위해서 'shipDate' 엘리먼트의 'minOccurs' 값을 '0'에서 '1'로 변경하여 필수 값으로 지정하는 갱신을 하였다.

```
select e.EId, le.PCId from XSchema..XElement e
inner join XSchema..XLocalelement le on e.EId = le.EId
where e.EName = 'shipDate'

select EId from XSChema..XElement
where CId = 4
```

	EId	PCId
1	17	4

	EId
1	12

(그림 7) XML 데이터 추출 과정

```
declare @EId int
select @EId = MIN(EId) from XData..XElement where SEId = 12

while @EId is not null
Begin

        if not exists (select * from XData..XElement
        where ParentId = @EId and SEId = 17)
        begin
                update XData..XElement set Valid = 0
                where EId = @EId
        end

        select @EId = MIN(EId) from XData..XElement
        where SEId = 12 and EId > @EId
End
```

```
(1개 행 적용됨)
```

(그림 8) XML 데이터 검증 과정

(그림7)는 'shipDate'의 변경과 관련된 부모 컴플렉스 타입과 이와 연결된 XML 데이터의 추출과정을 보여준다. (그림8)에서는 추출된 XML 데이터를 바탕으로 순환 검증과정을 보여주고, (그림9)은 스키마 변경에 의해서 무효한 데이터의 Valid가 '0'으로 변하는 최종결과를 보여준다.

```
select * from XData..XElement
where Valid = 0
```

	EId	EName	NS	Valid	PathId	SOrder	ParentId	SEId
1	15	item	1	0	16	1	15	12

(그림 9) 유효하지 않는 XML 데이터

이와 같은 실험 결과를 통한 바와 같이 XML Schema 가 갱신 후에도 유효한 문서이고, 이로 인해 유효성이 상실된 (Invalid) XML 데이터의 상태가 동시에 변경된 것을 확인 할 수 있다. 그러므로 본 논문에서 설계된 스키마 모델은 XML Schema와 XML 데이터의 갱신이 데이터베이스에 저장된 상태에서 이루어지므로 가장 적은 비용으로 유효성을 보장받을 수 있는 저장 및 갱신기법이다.

5. 결 론

기존 저장 방식에서는 데이터베이스에 저장된 XML Schema에 대해서 자체 유효성을 보장할 수 없는 방법이 없다. 그래서 스키마 갱신 필요성이 발생할 경우에 기존의 저장방식으로는 다시 파일로 추출하여 검증하는 비효율적인 절차를 거쳐야 한다. 또한 연결된 XML 데이터에 대해서도 더 이상 유효성을 보장할 수 없는 문제점을 가지고 있었다. 본 논문에서 설계한 XML Schema 갱신모델은 공통속성과 개별속성 구별을 위한 슈퍼타입·서브타입을 이용한 모델링과, 참조 키, Not Null, Default, Check(Rule), Trigger 의 제약 조건을 이용하여 다른 응용프로그램 도움 없이 스키마 유효성을 검증할 수 있다. 또한 XML 데이터 검증절차에서 XML Schema 갱신 내용에 따라 '범위성에 의한 XML 데이터 검증 필요성 확인 알고리즘'과 'XML 데이터의 유효성 검증 알고리즘'을 제시하여 필요한 경우에만 검증 후 변경되므로 불필요한 작업이 발생하지 않는 장점이 있다. 이로 인해 언제나 유효한 XML Schema임을 보장할 수 있고, XML 데이터의 상태가 즉시 확인되어 유효하지 않는 데이터가 존재하는 위험성을 제거하였다. 향후 과제는 DOM을 이용한 XML Schema 변경 시 제안한 시스템에 적용될 수 있는 SQL 갱신문 자동 시스템을 구현하는 것이다.

<참고문헌>

[1] Daniela Florescu, Danald Kossmann "Storing and Querying XML Data using an RDMBMS", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 1999

[2] Jayavel Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang, David DeWitt, Jeffrey Naughton, "Relational Databases for Querying XML Documents: Limitation and Opportunities", Proceedings of the 25th VLDB, 1999

[3] 이지현(2004), "XML Schema에 대한 유효성을 보장하는 ORDBMS에 저장된 XML 데이터 갱신 기법", 석사 학위 논문, 한국과학기술원

[4] 김상균(2001), "XML 데이터베이스 변경 연산의 부분 즉시 검증 메커니즘", 공학석사 학위 논문, 충북대학교

[5] Lee, D., Chu, W. W., Oct. 2000b "Constraints-preserving Transformation from XML Document Type Definition to Relation Schema", In: Int'l Conf. on Conceptual Modeling (ER). Springer, LNCS 1920, Salt Lacke City, UT.

[6] XML Schema Part 1 : Structures Second Edition, W3C, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028