XML Schema에 대한 유효성를 보장하는 효율적인 XML Data 갱신 기법

이지현[O] 정진완
한국과학기술원
{hyunlee, chungcw}@islab.kaist.ac.kr

An Efficient Updating Method conforming to XML Schema for XML data

Jihyun Lee[O] Chin-Wan Chung
Division of Computer Science, Department of Electrical Engineering & Computer Science
Korea Advanced Institute of Science and Technology

요 약

XML은 인터넷 상에서의 표준 데이터 형식으로 XML 데이터에 대한 구조적 제약 조건들은 DTD나 XML Schema에 정의한다. XML Schema에 유효한 XML 문서는 XML Schema에 정의된 모든 구조적 제약 조건들을 만족해야 하며, 갱신이 발생한 후에도 대응되는 XML Schema에 대해 유효함이 보장되어야 한다. 본 연구는 DBMS에 저장되어 있는 XML Schema에 대해 유효한 XML 데이터를 갱신 할 때 필요한 효율적인 유효 검증 기법을 제안한다. 이 유효 검증 기법에서는 유효 검증 범위를 갱신과 관련된 영역에 한정시키고 갱신 이전에 유효 검증을 수행하여 유효 여부를 판단한다. 또한 본 연구에서는 유효 검증을 위한 유효 검증 제약 조건들과 XML Schema문서의 특성을 분석하여 유효 검증 시 필요한 스키마 정보만을 효율적으로 추출할 수 있는 XML Schema 저장 방법 및 그에 따른 스키마 정보 추출 방법을 제안하고 실험을 통해 그 성능을 보인다.

1 서 론

XML (eXtensible Markup Language)[1] 이 웹 (World Wide Web) 데이터 표현 및 교환을 위한 표준으로 제안된 후, 많은 XML 데이터들이 DBMS에 저장되어 왔다. 그리고 XML 문서 상의 구조적 제약 조건들은 DTD와 XML Schema[2]와 같은 스키마에 정의된다. 이 중 XML Schema는 표준 XML 스키마로서 풍부한 데이터 타입과 상속을 지원하는 등 DTD에 비해 기능적으로 우월한 특징을 가지고 있다. XML Schema에 대해 유효한 XML 문서란, 대응되는 XML Schema에 정의된 모든 구조적 제약 조건들을 만족하는 문서를 의미하며 이러한 XML Schema에 대해 유효한 문서는 갱신 (update) 이 된 후에도 XML Schema에 대해 유효함이 보장되어야 한다. 따라서 XML 데이터에 대한 갱신을 지원하는 시스템들은 XML Schema에 대한 유효 검증 메커니즘을 포함한 갱신 방법을 제공해야 한다. 따라서 본 연구에서는 XML 데이터 갱신에 대하여 효율적으로 XML Schema 유효성을 검증하는 방법을 제안하고자 한다.

우선 XML 문서 상의 특정 요소와 그에 대응되는 XML Schema 상의 스키마 컴포넌트 간의 효율적인 매핑 방법을 제안한다. 그리고 XML Schema에 유효한 XML 문서가 만족해야 할 유효 제약 조건을 요소 타입 별로 분석 정리 하였다. 그리고 이를 기반으로 하여 효율적으로 XML Schema에 대한 유효를 보장하는 갱신 방법과 이 방법을 지원하기 위해 유효 검증 시 스키마를 효율적으로 추출하기 위한 스키마 저장 방법을 제안한다. 마지막으로 스키마 저장 방법에 따른 성능 차를 실험 결과를 통해 보인다.

2 관련연구

기존의 XML 저장 시스템에서 제공하는 갱신 및 유효 검증에 대해서 간단히 살펴보고자 한다. 우선 파일 시스템에 XML 문서를 저장한 경우 갱신을 위해서는 문서 전체를 메모리에서 파싱하여, 갱신 후 재저장 해야 하므로 갱신 성능 자체가 좋지 않으며 스키마를 동반한 경우 스키마에 대한 유효 검증을 위해서는 갱신 후 XML 문서 전체에 대한 유효 검증이 불가피하다. 객체지향 데이터베이스 시스템을 기반으로 한 대표적인 XML 저장 시스템인 eXcelon[3]에서는 스키마로 DTD만을 지원하고 있으며 새로운 XML 문서가 저장 될 때, 이 문서에 대한 유효 검증은 사용자 옵션으로 제공하고 있으나 갱신에 대한 유효 검증 메커니즘은 제공하고 있지 않다. 관계형 및 객체 관계형 데이터베이스 시스템을 기반으로 한 대표적인 XML 저장 시스템인 Oracle XML DB[4]에서는 XML Schema로부터 관계형 스키마를 생성하여 데이터를 저장하기 때문에 새로운 문서를 저장할 때나 삽입 갱신이 발생 했을 때에는 XML Schema에 정의되지 않은 엘리먼트들이나 에트리뷰트들을 검출해내는 등의 간단한 유효 검증은 제공하지만 XML Schema에 정의된 모든 제약 조건들을 체크하기 위해서는 갱신 후 문서 전체에 대하여 유효 검증을 수행해야 하며 이는 굉장한 오버헤드를 초래한다.

3 XML 데이터와 XML Schema 모델

3.1 XML 데이터

XML 문서는 일반적으로 트리 형태로 표현되며 트리 상의 모든 노드들은 이름을 가진 엘리먼트 (element), 에트리뷰트 (attribute), 그리고 값을 갖는 노드들로 구성된다. 엘리먼트 간에는 순서가 존재하며 각 엘리먼트는 자식 엘리먼트들과 에트리뷰트들을 갖는다. 그리고 트리에서 말단 노드들은 다양한 타입의 값 노드들이다. 본 연구에서는 문제의 간소화를 위해 모든 값의 타입을 스트링 (string) 타입으로 간주한다.

3.2 XML Schema

XML Schema는 XML 문서상의 엘리먼트, 에트리뷰트, 그리고 그들의 타입 및 구조가 정의된 XML 문서이다. 타입 및 구조 정의를 위해 XML Schema에서는 한정된 종류의 엘리먼트들과 에트리뷰트를 사용한다. 이들 중 본 연구에서는 schema, import, element, complexType, simpleContent, complexContent, extension, choice, sequence, attribute 엘리먼트들과 그와 관련된 에트리뷰트들 - targetNamespace, namespace, maxOccurs, minOccurs, name, ref, type, base - 을 이용하였으며 이들만으로도 XML Schema의 주요 기능들은 대부분 표현할 수 있다.

3.3 XML 데이터와 XML Schema 컴포넌트 간의 매핑

우선 본 연구에서는 XML 문서 상의 특정 노드에 대한 스키마 선언에 대응되는 부분을 스키마 컴포넌트라 하겠다. 즉, 노드가 엘리먼트라면 스키마 상에서 그 엘리먼트에 대한 선언 부분이 그 엘리먼트에 대한 스키마 컴포넌트이다.

XML 문서 상에서는 동일한 이름을 가진 엘리먼트이더라도 그 엘리먼트가 선언된 상위 타입에 따라 다른 타입을 가질 수 있다. 즉, 특정 노드의 스키마 컴포넌트를 찾기 위해서는 그 노드의 선언을 포함하고 있는 타입을 알아야 하는데 이는 부모 노드의 타입이다. 그리고 그 부모 노드의 타입 역시 그의 부모 노드의 타입을 알아야 찾아낼 수 있다. 다시 말하면 궁극적으로 특정 노드의 스키마 컴포넌트는 그 노드의 루트로부터의 노드 경로를 통해 매핑 된다.

4 XML 데이터 갱신

4.1 XML Schema 유효 제약 조건

XML Schema에는 XML 문서 상의 요소들의 종류, 구조, 타입 정보들과 같은 제약 조건들이 정의, 선언되어 있으며 XML 문서 상의 모든 요소들이 정의된 제약 조건들을 만족 시킬 때, 그 문서를 XML Schema에 대한 유효한 XML 문서라고 한다. XML 트리에서 특정 엘리먼트에 새로운 엘리먼트나 에트리뷰트가 삽입 혹은 삭제되면, 엘리먼트가 가지는 서브 트리가 변형되며 변형된 결과가 XML Schema에 선언된 그 엘리먼트의 유효 제약 조건들을 만족하는지 검증해야 한다. XML Schema에 대해 유효한 XML 문서가 갖추어야 할 유효 제약 조건들에는 XML 문서 상의 각 엘리먼트들이 만족해야 할 엘리먼트 유효 제약 조건, 특정 엘리먼트의 서브 트리가 만족해야 할 조건인 엘리먼트 타입 유효 제약 조건, 특정 엘리먼트의 서브 트리에서 엘리먼트 그룹이 만족해야 할 조건인 그룹 유효 제약 조건, XML 문서 상의 에트리뷰트 들이 만족해야 할 에트리뷰트 유효 제약 조건, ID 에트리뷰트가 만족해야 할 조건인 ID 타입 에트리뷰트 유효 제약 조건이 포함된다.

4.2 갱신 이전의 갱신 부분 유효 검증 방법

본 연구에서 제안하는 효율적인 XML Schema 유효 검증을 제공하는 XML 데이터 갱신 과정을 설명하도록 하겠다. 그림 1은 갱신 처리 과정을 개괄적으로 보여준다.

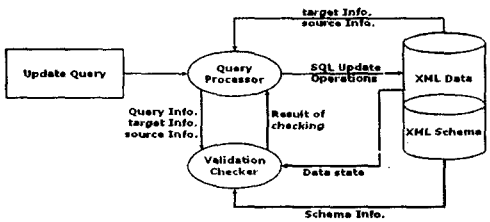

그림 1. XML 데이터 갱신 메커니즘

우선 갱신 질의가 들어오면 Query Processor는 질의를 분석하고 데이터베이스로부터 갱신이 적용될 타겟 노드 정보를 추출 해 온 후, Validation Checker를 호출하며 갱신 질의 정보와 타겟 노드 정보를 넘긴다. Validation Checker는 갱신 질의 종류에 따른 XML Schema 유효 제약 조건과 타겟 노드에 따라 갱신과 관련된 데이터 상태 정보 및 스키마 정보를 데이터베이스로부터 가져와 갱신 이후의 XML Schema 유효 여부를 예측하고 그 결과를 Query Processor로 반환한다. 그리고 예측 결과가 유효라면 Query Processor는 적절한 SQL 갱신 질의를 생성하여 데이터베이스 상의 XML 데이터를 갱신한다. 이 방법은 갱신 후 갱신된 데이터 전체에 대하여 XML Schema 유효 검증을 수행하는 것이 아니라 갱신이 발생하기 이전에 갱신의 종류와 갱신과 관련된 정보를 근거로 갱신될 일부 영역에 대해서만 유효 제약 조건들 만을 체크하는 " 갱신 이전의 갱신 부분 유효 검증" 을 수행함으로써 불필요하고 중복되는 유효 검증 범위를 없앴다.

4.3 갱신 별 유효 검증

4.3.1 삽입 갱신

우선 삽입되는 노드 n이 엘리먼트인 경우, 부모 노드가 될 엘리먼트 p의 서브 트리에서 엘리먼트 그룹이 갱신되므로 XML Schema 유효 제약조건 중 그룹 유효 제약 조건만 체크하면 된다. 즉, 삽입되는 노드 n이 스키마상에서 p의 자식 노드로 선언되어 있는지, 삽입 후 n의 카디널리티가 스키마에 선언된 maxOccurance를 초과하지 않는지, 그리고 스키마에 선언된 n이 속하는 엘리먼트 그룹의 종류가 sequence인지 choice인지에 따라, 그리고 그룹 내에서의 엘리먼트간 순서를 고려하여 형제 노드가 될 엘리먼트들 사이에 n이 삽입될 수 있는지를 체크하여 n의 삽입이 스키마에 유효한지를 예측한다. 삽입되는 n이 에트리뷰트인 경우에는 에트리뷰트 유효 제약 조건을 체크해야 한다. 즉, n이 스키마에 선언되어 있는지, 입력된 에트리뷰트의 값이 스키마에 선언된 타입을 만족시키는지, 그리고 이미 동일한 이름을 가진 에트리뷰트가 부모 노드가 될 p의 에트리뷰트로 저장되어 있지는 않는지를 체크한다. 또한 스키마 상에 n이 ID 타입으로 선언되어 있다면 동일한 값을 갖는 ID의 유무를 체크한다. 이러한 유효 검증 절차에 따라 n의 삽입이 스키마에 유효한지를 예측하며 유효하다고 판단되면 실제 데이터베이스를 갱신한다.

4.3.2 삭제 갱신

삭제되는 노드 n이 엘리먼트이면, n이 속한 엘리먼트 그룹에 변화가 생기므로 그룹 유효 제약 조건을 체크해야 한다. 즉, n이 삭제된 후, n의 카디널리티가 스키마에 선언된 minOccurance 미만이 되지 않는지를 체크한다. 삭제되는 노드 n이 에트리뷰트인 경우에는 n에 대한 에트리뷰트 유효 제약 조건을 체크한다. 즉, 스키마 상에서 n의 use 타입을 고려하여 반드시 문서상에 있어야 하는 (required) 에트리뷰트인지 확인한다. 그리고 이러한 유효 검증 절차에 따라 n의 삽입이 스키마에 유효한지를 예측하며 유효하다면 실제 데이터베이스를 갱신한다.

4.3.2 그 밖의 갱신

본 연구에서는 삽입, 삭제 갱신 이외에도 엘리먼트 및 에트리뷰트 이동 (move), 엘리먼트 및 에트리뷰트 새 이름 할당 (rename), 엘리먼트, 에트리뷰트 및 텍스트 값 수정 (modify) 갱신들에 대한 유효 검증 메커니즘과 데이터베이스 갱신 프로세스를 설계, 구현, 실험하였다. 대체로 큰 맥락은 앞서 설명한 삽입, 삭제 갱신과 유사하므로 지면 관계상 자세한 설명은 생략한다.

5 XML Schema 저장 방법

스키마를 추출하는 방법은 스키마 저장 방법에 따라 결정된다. 본 연구에서는 스키마를 데이터와 함께 객체 관계형 데이터베이스 시스템에 저장하였고 이를 위한 XML Schema 저장 방법을 제안하고자 한다.

5.1 네임스페이스 별 저장 방법

XML Schema 상에 선언된 스키마 컴포넌트들은 특정 네임스페이스에 종속된다. 즉, 그들의 선언을 포함한 XML Schema의 타겟 네임스페이스 영역이 그들의 유일성을 보장 받는 영역이 된다. 따라서 네임스페이스 별 저장 방법은 XML Schema을 타겟 네임스페이스 별로 저장하되 스키마 전체를 하나의 스트링으로 보고 하나의 스키마 테이블에 저장하게 된다.

3.3 절에서 언급한 바와 같이 XML 문서 상의 특정 노드에 대응되는 스키마 컴포넌트는 노드의 노드 경로를 통해 매핑 된다. 따라서 특정 노드의 스키마 정보를 추출 하고자 할 때에는 노드 경로 상의 노드들이 정의된 스키마들을 차례로 메모리로 가져와서 파스 트리를 만들고 이 파스 트리를 검색함으로써 해당 스키마 컴포넌트를 추출하게 된다. 따라서 이 방법은 XML Schema의 크기가 크고, 참조하는 XML Schema의 수가 많으면 관련된 모든 XML Schema를 모두 메모리로 올려 파스 트리를 유지 해야 하므로 불필요한 내용까지 모두 메모리에 유지 시켜야 한다. 그러므로 메모리 효율성이 낮고, 불필요한 검색영역이 크다는 단점이 있다.

5.2 타입 별 저장 방법

XML Schema는 일종의 XML 문서이므로 기존에 연구되어온 XML 데이터 저장 방법에 따라 저장할 수 있다. 그러나 XML Schema는 문법과 의미가 미리 정해져 있는 한정된 엘리먼트, 에트리뷰트들을 가지기 때문에 저장 효율성을 높이고 동시에 효율적으로 특정 노드의 스키마 정보를 추출하기 위해서는 순수한 XML 데이터 저장 방법으로는 한계가 있다. 따라서 본 연구에서는 타입 별 저장 방법을 고안하였다. 이 방법은 XML 데이터 저장 방법 중 하나인 Binary approach[5]에서처럼 기본적으로는 엘리먼트의 종류에 따라 테이블을 생성하되 각 엘리먼트 종류에 종속되어 관련된 일부 엘리먼트들과 에트리뷰트들을 해당 엘리먼트 테이블에 인라인닝 시키며 스키마 문서 파싱 과정에서 생성한 효율적인 유효 검증을 위해 필요한 정보를 추출하여 별도의 테이블 에트리뷰트에 저장하였다.

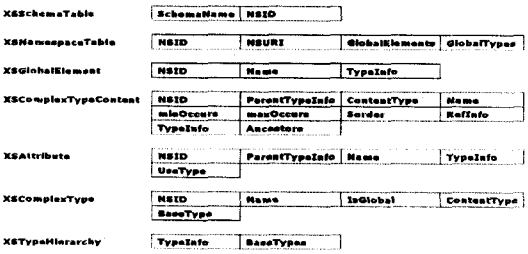

그림 2. 타입 별 저장 방법의 스키마

그림 2는 타입 별 저장 방법의 저장 스키마 전체를 보여준다. 기본적으로는 앞서 설명한 바와 같이 스키마 상의 element, complexType, attribute 엘리먼트 별로 테이블 (XSGloblaElement, XSComplexTypeContent, XSComplexType, XSAttribute) 을 만들었고 각 엘리먼트에 종속되는 에트리뷰트들과 엘리먼트들은 인라이닝 시켰다. 그리고 여러 XML Schema 문서가 하나의 네임스페이스로 선언 될 수 있기 때문에 XML Schema 문서 정보를 저장하기 위해 XSSchemaTable을 두었고 네임스페이스 정보들은 XSNamespaceTable에 저장하였다. 또한 XML Schema에서는 상속을 지원하므로 타입의 상속 관계를 효율적으로 관리하기 위해 별도의 XSTypeHierarchy 테이블을 두었다.

이 방법으로 스키마를 저장한 경우, 특정 노드의 스키마 정보를 추출하고자 할 때에는 Path 테이블에 노드 경로 정보와 함께 노드 경로에 대응되는 노드의 선언을 포함하는 상위 타입 정보를 저장해 둠으로써 이 타입 정보와 노드의 이름을 키 값으로 필요한 스키마 정보 만을 필요한 순간에 추출 할 수 있다.

6 실험 및 결과

앞서 설명한 두 가지 스키마 저장 방법에 따른 갱신에 대한 스키마 유효 검증 성능 차를 비교하기 위한 실험을 수행하였다. 실험에 사용된 스키마는 OASIS산하의 UBL TC에서 구매발주 비즈니스 문서에 대한 표준 XML Schema로 제시한 Op70[6]을 이용하였고, 이를 기반으로 하여 작성된 XML 문서를 갱신되는 XML 데이터로 이용하였다. Op70은 기본적으로 Core Component Types, Re-usable Component Library Schema, 그리고 이 두 Schema를 참조하는 비즈니스 문서를 위한 Schema[1] 세 가지로 구성된다. 그리고 스키마 크기와 복잡도에 따른 성능 변화를 보이기 위한 비교 대상으로 xsd/UBL_Library_Op70_Order Schema와 관련된 컴포넌트들만을 포함한 simple_order.xsd를 만들어 실험에 이용하였다. 실험 결과는 그림 3과 같다.

[1] 본 연구에서는 주문서와 관련된 schema인 xsd/UBL_Library_Op70_Order schema를 이용하였다.

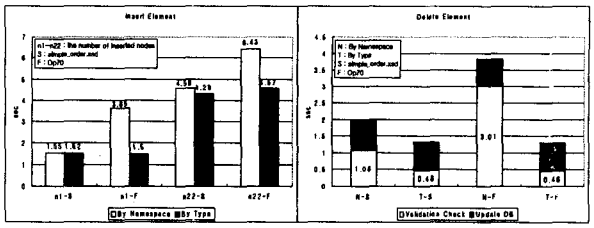

그림 3. 스키마 저장 방법에 따른 삽입, 삭제 갱신 실험 결과

우선 엘리먼트 삽입 (Insert Element) 의 경우, 삽입되는 서브트리 상의 노드의 수를 1개에서 22개까지 증가시켜 실험을 수행하였다. 갱신에 총 소요된 시간은 유효 검증을 위해 스키마를 추출하는 시간, 필요한 데이터 추출시간, 유효 검증 시간, 그리고 데이터베이스 갱신 시간을 포함한다. 스키마 저장 방법에 따라 차이가 나는 부분은 스키마 추출 시간이며 타입 별 저장 방법에 따라 스키마를 저장하였을 때 네임스페이스 별 저장 방법에 따라 저장하였을 때 보다 더 적은 시간이 소요되므로 스키마 추출 시간이 더 빨랐음을 알 수 있다. 엘리먼트 삭제 (Delete Element) 의 경우에도 차이가 나는 부분은 스키마 추출 시간이며 역시 타입 별 저장 방법에 따라 스키마를 저장 하였을 때 스키마 추출 성능이 더 좋았음을 알 수 있다. 그리고 두 그래프에서 크기와 복잡도가 작은 스키마인 simple_order.xsd (S) 와 큰 스키마인 Op70 (F) 을 저장한 경우를 비교했을 때, 타입 별 저장 방법에서는 변화가 거의 없으나 네임스페이스 별 저장 방법에서는 Op70의 경우 스키마 추출 시간이 크게 증가하는 것을 알 수 있다. 이는 스키마 크기가 커지고, 참조하는 스키마 수가 많아지면 타입 별 저장 방법의 성능이 훨씬 더 좋음을 알 수 있다. 다른 갱신들에 대해서도 실험을 수행하였고, 그 결과는 앞서 보인 삽입, 삭제 갱신과 거의 비슷한 결과를 보였다.

7 결론

본 연구에서는 XML Schema에 대해 효율적으로 유효성을 보장하는 XML 데이터 갱신 기법에 대한 연구를 수행하였다. 노드 경로를 통한 노드와 XML Schema 컴포넌트간의 매핑 방법을 제안하였고, 유효 보장을 위해 검증해야 할 요소 별 유효 제약 조건을 분석하였다. 그리고 이를 기반으로 갱신 이전 유효 검증을 제공하는 XML 데이터 갱신 메커니즘을 설계하였다. 또한 스키마 저장 방법으로 나이브한 방법인 네임스페이스 별 저장 방법과 더불어 유효 검증을 위해 필요한 스키마 정보를 효율적으로 추출하기 위한 타입 별 저장 방법을 제안하고 Path 테이블을 이용해 효율적으로 스키마를 추출하는 방법을 소개하였다. 마지막으로 실험을 통해 본 연구에서 제안한 타입 별 저장 방법이 유효 검증을 위한 효율적인 스키마 추출을 제공하는 스키마 저장 방법임을 보였다.

8 참고문헌

[1] Extensible Markup Language(XML) Specification, W3C Recommendation, 6 October 2000, http://www.w3.org/TR/REC_xml

[2] XML Schema, W3C Recommendation, 2 May 2001, http://www.w3.org/XML/Schema

[3] eXcelon Corporation. Updating XML data. eXcelon 1.1 User Guide Chapter 6, 2001

[4] Oracle XML DB, Oracle9i XML Database Developer' s Guide - Oracle XML DB Release 2 (9.2), October, 2002

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

[6] Universal Business Language-Library Content - Op70 Public Review, http://oasis-open.org/committees/ubl/lcsc/Op70/