XML 스키마의 의미적 통합

강혜란°, 신동훈, 이경호

연세대학교 컴퓨터과학과

hrkang@icl.yonsei.ac.kr°, dhshin@icl.yonsei.ac.kr, khlee@cs.yonsei.ac.kr

Semantic Integration of XML Schema

Haeran Kang°, Donghoon Shin, Kyong-Ho Lee

Computer Science Department, Yonsei University

요 약

다양한 분야에서 XML이 정보 교환의 표준으로 사용됨에 따라 XML 문서 검색을 위한 XML 스키마 통합의 중요성이 증대되고 있다. 본 논문에서는 스키마의 의미를 보다 정확하게 반영한 스키마 통합 방법을 제안한다. 제안된 방법은 공통 구조 추출, 스키마 통합, 그리고 최적화의 세 단계로 이루어진다.

1. 서 론

XML(eXtensible Markup Language)은 인터넷을 비롯한 다양한 분야에서 정보 표현 및 교환을 위한 표준으로 널리 사용되고 있다. 인터넷 상에 분포되어 있는 XML 문서를 검색하기 위해서는 XML 스키마를 통해 문서의 구조와 의미를 파악하여야 한다. 그러나 XML 스키마는 제작자에 따라 같은 의미의 스키마도 다른 형태로 형성될 수 있는 다형성을 갖고 있어 정보를 얻는데 어려움이 있다. 그러므로 유사한 스키마들을 하나로 통합하는 스키마 통합의 필요성이 점차 강조되고 있다. 통합 스키마를 추출하는데 있어서 고려해야 할 기준으로는 완전성(completeness), 간결함(conciseness), 생성 시간, 자동화 여부 등이 있다.

한편 기존에 통합 스키마 추출을 위한 여러 방법이 제안되었다[1, 2, 3]. 기존의 통합 스키마 추출을 위한 방법들 중 Meo 등 [1]은 통합된 XML 스키마를 추출하기 위해 두 노드의 유의어 관계를 구할 때 두 노드 이름의 유의어 여부와 함께 두 노드와 일정한 구조적 및 의미적 관계를 가진 다른 노드들의 이름의 유의어 여부도 고려한다. Passi 등 [2]은 XML 스키마 통합을 위해 엘리먼트의 의미적 관계를 6가지 타입으로 분류하며 비 단말 엘리먼트를 엘리먼트간의 의미적 관계 타입에 따라 나눈다. 그리고 통합 방법을 all, 순서, 선택 연산자와 같은 연산자별로 나누어 제시하고 있다. Hakimpour 등 [3]은 서로 다른 커뮤니티에 속한 XML 스키마 통합을 위해 커뮤니티 온톨로지를 통합한다. Mello 등 [4]은 XML DTD 통합을 위한 단말 노드간 통합에서 단말 노드의 이름, 데이터 타입은 물론 enumeration 통합 방법도 제시하고 있다.

스키마 통합을 위한 기존 연구들은 두 노드와 일정한 관계를 가진 노드들을 추출하여 두 노드의 의미적 관계를 구할 때 의미적 관계를 정확히 구하지 못하였다. 또한 서로 다른 연산자 하의 유의어 노드들의 통합방법을 충분히 제시하지 못했고, 서로 다른 연산자 하의 부분집합 관계 노드들의 통합방법은 전혀 찾을 수가 없었다. 그리고 기존 최적화 알고리즘만으로는 통합된 스키마를 충분히 효율적으로 만들지 못했다.

2. 스키마 통합

제안된 알고리즘은 공통구조 추출, 스키마 트리 통합, 최적화의 세 단계로 구성된다.

※이 연구는 정보통신부(정보통신연구진흥원)에서 지원하는 2005년도 IT기초기술연구지원사업의 연구결과임.

2.1 공통구조 추출

본 절에서는 통합에 필요한 공통구조를 추출하는 과정을 기술한다.

2.1.1 Context 추출

본 절에서는 스키마간 특성을 구하는 첫 번째 단계로 각 노드들의 Context를 추출한다. 본 논문에서는 노드 N_a의 Context를 N_a와 일정 수준 이상의 구조적 그리고 의미적 연관 관계를 갖는 노드라고 정의한다. 그런데 Context를 구하기 위해서는 부모노드에 대한 서술적 의미 타입을 구하는 것이 필요하다. 그러므로 우선 부모노드에 대한 서술적 의미 타입에 대해 논하겠다.

부모노드에 대한 서술적 의미 타입은 <그림 1>의 예와 같이 부모 노드에 대해 자식노드가 의미상 부분집합인 경우, 서술적 의미 타입을 부분집합 형이라고 정의한다. 또한 부모 노드에 대해 자식노드가 의미상 부분집합이 아닌 경우, 이 부모노드에 대한 서술적 의미 타입을 비 부분집합 형이라고 정의한다.

또한 자식 노드가 아닌 경우에도 한 노드가 다른 노드의 의미상 부분집합인 경우 두 노드는 부분집합 관계에 있으며 그렇지 않은 경우 비 부분집합 관계에 있다고 정의한다.

부모노드에 대한 서술적 의미 타입에 따라 Context의 종류를 다음과 같이 비 하위 노드 Context, 단말 노드 Context, 의미적 하위 노드 Context의 세 가지로 분류한다. 우선 노드 N_a에 대해, 비 하위 노드 Context는 뿌리노드와 N_a 사이의 경로 상에 존재하는 모든 노드와 이들이 참조한 노드들의 집합이다. 그리고 단말 노드 Context는 N_a 노드를 조상노드로 갖는 모든 단말 노드들의 집합이다. 마지막으로 의미적 하위 노드 Context는 N_a 노

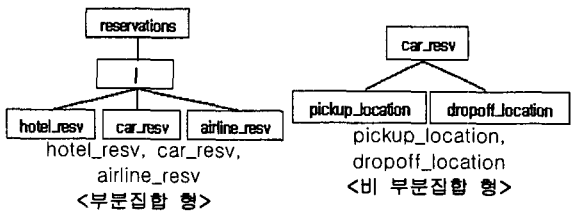

그림 1. 부모노드에 대한 서술적 의미 타입 예.

드로부터 단말노드까지 경로 상에 존재하는 노드들 중 부모노드에 대한 서술적 의미 타입이 부분집합 형인 노드가 존재하는 최소 깊이보다 작거나 같은 깊이에 존재하는 부분집합 형 노드들

(부분집합 형 의미적 하위 노드 Context)과 비 부분집합 형(비 부분집합 형 의미적 하위 노드 Context)인 노드들의 집합이다.

2.1.2 유의어와 부분집합 관계 구하기

본 절에서는 공통구조를 구하는 두 번째 단계로써, 두 노드의 유의어와 부분집합 관계를 구하는 방법을 기술한다. 이러한 두 노드의 의미적 관계를 구하기 위해서 우선 소스 및 타겟 스키마 트리의 뿌리노드로부터 단말노드까지의 모든 경로 $SPath_i$, $0 \le i \le n$와 $TPath_j$, $0 \le j \le m$를 추출한다. 그 다음에, 추출된 각 두 경로 $SPath_i$와 $TPath_j$에 대해, $SPath_i$상의 모든 노드들과 $TPath_j$상의 모든 노드들 간의 유의어 관계를 <그림 3>의 함수와 상수를 사용한 <그림 2>의 조건을 사용해 구한다.

<그림 2>에서 두 노드의 Context를 비교할 때, 의미적 하위 노드 Context들을 부분집합 형과 비 부분집합 형으로 분리해서 비교하는 이유는, 두 노드가 실제적으로 유의어 의미를 갖고 있을 경우에도, 부분 집합 형 하위 노드 Context와 비 부분집합 형 하위 노드 Context 사이에 유의어가 존재할 가능성은 매우 적기 때문이다.

임의의 두 노드 N_a와 N_b가 유의어 관계를 가질 조건

N_a와 N_b가 WordNet[5] 상에서 유의어 관계이고, 두 노드의 전체 Context가 유의어 관계에 있는지를 말해주는 함수 $SynN(N_a, N_b)$ 값이 ST 이상이면 두 노드 사이에는 유의어 관계가 성립한다.

$$SynN(x, y) = \sum_{CT \subseteq nsd,\, sdp,\, sdnp} SynC(x, y, CT) \times W_{CT}$$

$$SynC(x, y, CT) = \frac{2|Syn(C_{CT}(x), C_{CT}(y)) + \frac{1}{4} Hyper(C_{CT}(x), C_{CT}(y))|}{|C_{CT}(x)| + |C_{CT}(y)|}$$

임의의 두 노드 N_a와 N_b가 부분집합 관계를 가질 조건

N_a의 이름이 N_b의 이름의 부분집합 관계어이고,
$HyperN(N_a, N_b) \ge HT$ 를 만족하면 두 노드 사이에는 부분집합 관계가 성립한다.

$$HyperN(x, y) = \sum_{CT \subseteq nsd,\, sdp,\, sdnp} HyperC(x, y, CT) \times W_{CT}$$

$$HyperC(x, y, CT) = \frac{2|Syn(C_{CT}(x), C_{CT}(y)) + Hyper(C_{CT}(x), C_{CT}(y))|}{|C_{CT}(x)| + |C_{CT}(y)|}$$

그림 2. 임의의 두 노드가 의미적 관계를 가질 조건.

Syn(x, y) : x의 노드들 중 이름이 y에 속하는 어떤 노드의 이름에 대해 WordNet 상에서 유의어인 것들의 갯수
Hyper(x, y) : x의 노드들 중 이름이 y에 속하는 어떤 노드의 이름에 대해 부분집합형인 것들의 갯수
CT(Context Type) :
  nsd(비 하위노드 Context와 단말 노드 Context),
  sdp(의미적 하위노드 Context 중 부분집합 형 Context),
  sdnp(의미적 하위노드 Context 중 비 부분집합 형 Context)
$C_{CT}(x)$: Context Type이 CT인 노드 x의 Context 집합
W_{CT}: 각 Context 종류별 가중치

그림 3. 두 노드의 의미적 관계 계산을 위한 함수 및 상수.

2.1.3 공통구조 추출

본 절에서는 2.1.2에서 추출된 노드들 간의 유의어 관계를 기반으로, 공통의 노드 및 관계를 추출하여 공통구조를 추출한다. 공통구조 추출은 경로 간 BSV 계산과 GCS 계산의 두 단계를 거친다. BSV(Biggest Similarity Value)는 일반적인 두 대응 경로 사이의 최대 유사도를 말하며 각 경로의 노드 사이의 일대일 매칭을 통하여 추출한다. 여기에서 최대 유사도라는 것은 전체 노드 수에 대한 유의어 관계에 있는 노드들과 상위어 상수를 곱한 상위어 관계에 있는 노드들의 비율이다. GCS(Greatest Common Subset)는 <그림 3>의 함수들을 이용한 <그림 4>의 알고리즘을 통해 구한다.

Algorithm FindGCS
입 력 : 소스 스키마 트리 T_s와 타겟 스키마 트리 T_t
출 력 : T_s 와 T_t 사이의 GCS
1. T_s의 각 경로 $SPath_i$와 T_t의 각 경로 $TPath_j$간의 SV(Similarity Value)를 계산
2. 각 경로 $SPath_i$와 $TPath_j$를 정점으로 하며 각 경로간 SV의 크기를 간선의 가중치로 갖는 가중치 이분 그래프 $K_{n,m}$를 생성
3. $K_{n,m}$에서 최대 이분 매칭을 찾음. 이때의 BSV의 합이 GCS(Greatest Common Subset), 즉 공통구조

그림 4. GCS 계산 알고리즘.

2.2 통합

본 절에서는 공통 구조 추출 단계에서 계산된 공통구조 및 매칭관계를 이용하여 스키마 트리를 통합한다. 우선, 전 단계에서 구해진 공통구조에서 통합 대상이 되는 공통구조는, 공통구조 중에서 최대 이분 매칭 관계에 속한 두 경로의 노드들 중 뿌리 노드에서 부터 유의어 관계 또는 부분집합 관계로 연속적으로 연결된 노드들과 그 사이의 연산자이다. 우선 최대 이분 매칭에 속한 각 두 경로를 통합 하고 나서, 최대 이분 매칭에 속하지 않는 경로를 통합한다. 최대 이분 매칭에 속하는 각 두 경로를 통합 할 때는 앞에서 구해진 통합 대상이 되는 공통구조의 연산자와 노드를 통합 하고 나서 통합 대상이 아닌 부분을 통합 스키마에 반영한다.

2.2.1 연산자 통합

통합 대상이 되는 공통구조에 있는 노드들을 통합할 때, 첫 번째 단계로써 연산자를 통합한다. 현재 노드와 부모 노드 사이에 있는 연산자를 현재 노드가 갖고 있는 연산자라고 정의한다. 그리고 현재 노드는 그 연산자의 노드라고 정의한다. 연산자 통합 방법은 우선 그 연산자의 노드가 유의어 관계 노드이냐 부분집합 관계 노드이냐에 따라 달라진다.

유의어 관계 노드가 갖는 연산자의 통합 방법은 현재 통합 대상인 두 노드가 갖고 있는 연산자의 개수에 따라 달라진다. 한 노드는 연산자를 갖고 있지 않고 다른 노드는 연산자를 하나 가지고 있는 경우의 통합 방법이 있고, 두 노드 모두 연산자를 하나 가지고 있는 경우의 통합 방법이 있고, 두 노드 중 한 노드라도 연산자를 두개 이상 가지고 있는 경우의 통합 방법이 있다.

한 노드는 연산자를 하나 가지고 있고, 다른 노드는 연산자를 갖고 있지 않은 경우 통합 방법은 다음과 같다. 존재하는 하나의 연산자가 all일 경우 연산자를 갖고 있지 않은 노드의 빈도수가 1보다 작거나 같으면, 통합된 연산자는 all이 된다. 또한 하나의 연산자가 순서(또는 선택)일 경우 연산자를 갖고 있지 않은 노드가 자신의 부모 노드의 유일한 자식 노드일 경우 통합된 연산자는 순서(또는 선택) 연산자가 된다.

두 노드 모두 연산자를 하나씩 가지고 있는 경우, 통합 방법은 다음과 같이 세 가지로 나뉜다. 현재 통합 대상이 되는 한 연산자의 모든 노드가 다른 연산자의 모든 노드와 유의어 관계에 있고, 두 연산자의 노드의 수가 같으며, 두 연산자도 동일할 경우에는 연산자는 기존의 두 연산자와 동일한 하나의 연산자로 통합 된다.

위의 과정에서 통합되지 않았으나 현재 통합 대상이 되는 한

연산자의 모든 노드가 다른 연산자의 모든 노드와 유의어 관계에 있을 경우의 통합 방법은 다음과 같다. 두 연산자의 노드들 중 노드의 수가 많거나 같은 한 연산자를 A, 노드의 수가 적거나 같은 다른 연산자를 B라고 하자. A가 all이고 B가 선택(또는 순서) 연산자일 경우 B의 자식 노드들이 element 밖에 존재하지 않고 B의 최대 빈도수가 1보다 작거나 같을 때 통합된 연산자는 all이 된다. 그리고 A가 순서 연산자이고 B가 선택 연산자이면 통합된 연산자는 순서 연산자가 된다.

또한, 위의 두 과정에서 통합되지 않았을 경우의 통합 방법은 다음과 같다. 두 연산자가 모두 선택(또는 all) 연산자이면 통합된 연산자도 선택(또는 all) 연산자가 된다. 한 연산자는 선택 연산자이고 다른 연산자는 all(또는 순서) 연산자인 경우, 하나의 선택 연산자는 기존의 선택 연산자의 노드들과 all(또는 순서) 연산자를 하부 구조로 갖고 기존의 all(또는 순서) 연산자의 노드들은 통합된 all(또는 순서) 연산자의 노드들이 된다. 그리고 기존의 두 연산자의 노드 중 유의어 관계에 있는 노드는 all(또는 순서) 연산자의 노드로 통합된다. 두 노드 모두 연산자를 하나씩 가지고 있으나 위 세 가지 통합 방법에 해당되지 않을 경우 연산자와 그 하부구조는 통합되지 않는다.

부분 집합 관계 노드가 갖는 연산자의 통합 방법은 유의어 관계 노드의 통합 방법과 같은 방식으로 나뉜다. 차이점은 한 연산자의 모든 노드가 다른 연산자의 모든 노드와 유의어 관계에 있는지 여부에 따라 통합 방법이 나뉘지 않는다는 점이다.

예를 들어, 한 노드는 all을 하나 가지고 있고, 다른 노드는 연산자를 갖고 있지 않은 경우, 통합된 연산자는 두 개의 all이 되어, 첫 번째 all의 노드는 의미상 부분집합을 포함하는 노드가 되고 그 노드는 하부 구조로 두 번째 all을 갖고 그 all의 노드는 의미상 부분집합에 포함되는 노드가 된다.

2.2.2 유의어 노드 통합

유의어 노드들을 통합할 경우, 두 노드가 각각 단말 노드인지 비 단말 노드인지에 따라 세 가지 경우로 나누어지고 이에 따른 통합 방법은 다음과 같다.

우선 단말 노드 간 통합 방법에서 단말 노드는 하위 구조를 포함하지 않는 엘리먼트나 애트리뷰트를 나타낸다. 두 노드는 노드 사이의 이름, 데이터 타입, 빈도 지시자의 충돌을 해결하여 통합 노드를 생성한다. 엘리먼트와 애트리뷰트 사이의 통합일 경우 엘리먼트로서 통합 개념을 형성한다.

- 이름 변환 규칙 : 두 이름 중 동일 클러스터 내에서 빈도수가 높은 것을 통합 노드의 이름으로 선택한다.
- 데이터 타입 변환 규칙 :

데이터 타입은 두 타입 중 정보 손실 없이 변환을 통해 다른 타입을 포함할 수 있는 타입을 선택한다. 변환이 불가능할 경우, 통합 노드는 데이터 타입으로서 문자열(string)을 갖는다.

- 빈도수 속성 변환 규칙 :

통합되는 노드는 두 빈도수 쌍 중 최소값 중의 작은 값과 최대값 중 큰 값을 빈도수 속성으로 갖는다.

단말 노드와 비 단말 노드를 통합할 경우, 통합 노드는 색인 내용 모델의 비 단말 노드로 생성된다. 새로운 노드는 비 단말 노드의 하위 구조를 포함하며 단말 노드간의 통합과 같은 방법으로 노드 사이의 이름 및 빈도수 충돌을 해결한다.

비 단말 노드와 비 단말 노드를 통합할 경우, 통합 노드는 비 단말 노드로 생성된다. 새로운 노드는 비 단말 노드의 하위 구조를 포함하며 단말 노드간의 통합과 같은 방법으로 노드 사이의 이름 및 빈도수 충돌을 해결한다.

2.2.3 부분집합 관계 노드 통합

부분집합 관계 노드의 통합할 경우 부분집합을 포함하는 노드를 부모 노드로 하고 부분집합에 포함되는 노드를 자식 노드로 통합한다. 부모 노드는 비 단말 노드가 되고 부모노드의 이름, 빈도수와 자식 노드의 타입, 이름 그리고 빈도수는 기존 스키마에서 대응되는 노드와 같아진다.

2.2.4 통합 대상이 아닌 부분 반영

전 단계에서 현재 통합되는 두 경로의 통합 대상이 되는 공통 경로의 통합을 끝낸 다음, 현 단계에서는 그 두 경로의 통합 대상이 아닌 부분을 통합 스키마에 반영한다. 전 단계에서 통합된 마지막 노드들이 유의어 관계에 있었다면 그 마지막 노드들의 자식 노드가 갖는 연산자를 유의어 관계 노드가 갖는 연산자의 통합 방법을 이용해서 통합한다. 그리고 통합 스키마 내의 그 연산자의 하부 구조는 그 연산자에 대응 되는 기존 스키마의 연산자의 하부 구조와 동일하게 반영한다.

전 단계에서 통합된 마지막 노드들이 부분집합 관계에 있었다면 통합된 마지막 노드의 하부 구조는 그 노드에 대응 되는 기존 스키마의 노드의 하부 구조와 동일하게 반영한다.

2.2.5 최대 이분 매칭에 속하지 않는 경로 통합

최대 이분 매칭에 속하지 않는 경로를 통합할 때는 우선 기존 스키마 내에서 현재 경로와 뿌리 노드에서 부터 연속적으로 연결된 공통의 노드 수가 최장인 최대 이분 매칭에 속하는 경로를 찾는다. 그리고 최대 이분 매칭에 속하지 않는 경로의 그 마지막 공통 노드의 하부 구조를, 최대 이분 매칭에 속하는 경로의 그 마지막 공통 노드에 대응되는 통합 스키마의 노드의 하부 구조와 선택 연산자로 연결한다.

2.3 최적화

통합된 스키마에 있는 노드들 중 하위 구조가 일정 수 이상의 노드를 포함하고, 다른 노드의 하위 구조와 동일한 노드의 비율이 역시 일정 수 이상이면 그 하위 구조를 전역 타입으로 선언하여 대응되는 노드들이 참조하게 하여 스키마를 간결하게 하는 객체 지향적 개념을 활용한 최적화 알고리즘을 사용한다.

3. 결론

제안된 알고리즘은 기존 연구에서 제시하지 못한 서로 다른 연산자의 통합 방법을 제시하였다. 또한, 두 노드의 의미를 보다 충분히 반영할 수 있도록 Context를 설정하였으며, 이러한 Context를 기반으로 하여 보다 의미적으로 정확한 유의어와 부분집합 관계를 구하는 방법을 제시하였다. 그리고 객체 지향적 개념을 활용한 최적화 알고리즘을 통해 보다 간결한 스키마를 추출할 수 있었다. 또한, 이 모든 과정을 사용자의 개입 없이 수행함으로서 알고리즘의 시간 복잡도를 크게 줄였다.

참고 문헌

[1] Pasquale De Meo, Giovanni Quattrone, Giorgio Terracina, Domenico Ursino, "Integration of XML Schemas at various "severity" levels,"Information Systems, In Press, Corrected Proof, Available online 31 January 2005.

[2] K. Passi, S. Madria, Bipin S., S. Bhowmich, M. Mohania, "A Model for XML Schema Integration," Proc. 3rd ECWEB, pp. 193-202, 2002.

[3] Farshad Hakimpour, and Andreas Geppert, "Resolving semantic heterogeneity in schema integration," Proc. Int'l Conf. Formal Ontology in Information Systems, pp. 297-308, 2001.

[4] Ronaldo dos Santos Mello, Silvana Castano, and Carlos A. Heuser, "A Method for the Unification of XML Schemata," Information & Software Technology, pp. 241-249, 2002.

[5] George A. Miller, "WordNet: A Lexical Database for English," Communications of the ACM, Vol. 38, No. 11, pp. 39-41, 1995.