제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

X-treeDiff를 이용한 XML 문서의 버전 관리 시스템 프로트타입 개발

김성준*, 김동아*, 이석균*
*단국대학교 정보컴퓨터학부
e-mail:{sungjoon, dakim70, sklee}@dankook.ac.kr

A Prototype Implementation of XML Document Version Management System Using X-treeDiff

Sung-Joon Kim*, Dong-Ah Kim*, Suk-Kyoon Lee*
*Dept of Information & Computer Science, Dankook University

요 약

현재 많은 정보시스템들이 웹을 기반으로 다양한 전자 문서들을 제공하고 있다. 이러한 환경 하에서 지속적인 갱신이 이루어지는 문서들을 관리하는 응용분야에서는 이들 문서들에 대한 효율적 관리 기법이 요구되고 있다. 본 논문에서는 최근 제안된 X-treeDiff를 통해 계산된 편집 스크립트를 기반으로 한 XML 문서들에 대한 버전 관리 시스템을 제안하고 이에 대한 프로토타입의 구현을 보인다. 제안된 버전 관리 시스템은 CVS와 같은 대부분의 텍스트 기반 시스템과는 달리 트리 데이터 구조의 문서를 위한 시스템으로 XML과 같은 트리구조 문서 관리에 효과적이다.

1. 서론

현재 많은 정보시스템들이 웹을 기반으로 구축·활용되고 있으며 다양한 전자문서들이 이를 통해 제공되고 있다. 이러한 환경 하에서 특허문서관리, 소프트웨어 설계, GML 등과 같은 XML 응용분야에서는 기존 문서에 대한 효율적 관리 기법이 요구되고 있다. 최근 기존 XML 문서에 대한 효율적 관리, 즉 효율적 버전관리에 대한 많은 연구들이 이루어지고 있다[1,2,3]. 이러한 연구들은 크게 XML 문서를 구성하는 엘리먼트들을 나누어 저장하는 분할저장 시스템에서의 버전 관리 기법[1,2]과 하나의 문서와 버전간의 차이만을 저장하고 이를 통해 버전을 관리하는 기법[3]으로 나누어볼 수 있다.

최근 제안된 X-treeDiff[4]는 웹 문서의 변화를 효율적으로 탐지하는 알고리즘으로 현재 웹 문서의 해킹 여부를 탐지하는 상용 시스템 WIDS(Web Intrusion System)에서 사용되고 있다. X-treeDiff는 문서의 노드 수를 n이라 할 때 시간 비용이 O(n)이며 기존의 연구 결과들[3,6] 보다도 뛰어난 성능을 보인다[4].

본 논문에서는 X-treeDiff를 기반으로 각 XML문서의 버전들의 차이를 저장하고 이를 통해 문서들의 버전 관리를 수행하는 버전 관리 시스템을 제안하고 그 프로토타입을 구현한다. 이러한 방식은 CVS[5]와 같은 대개의 텍스트 기반의 버전관리 시스템들에서 사용되고 있다. 한편 본 논문에서 제안하는 버전 관리 시스템에서는 대부분의 연구 결과들과는 달리 이동 연산을 포함하고 있어 생성된 편집 스크립트가 간결하며 훨씬 직관적이다. 또한 텍스트 기반의 변화탐지 알고리즘이 아니라 트리 데이터에 대한 변화탐지 알고리즘인 X-treeDiff[4]를 사용하고 있어 효율적인 버전 관리가 가능하다.

본 논문의 구성은 다음과 같다. 2절에서는 X-treeDiff 알고리즘을 소개하고 3절에서는 편집 스크립트의 생성 방법에 대해, 4절에서는 프로트타입 시스템 구현에 대해, 끝으로 5절에서는 결론과 향후 연구 과제에 대해 설명한다.

2. X-tree 모델과 X-treeDiff 알고리즘

본 절에서는 변화 탐지를 위한 트리 모델인

X-tree[4]와 이를 이용한 X-treeDiff 알고리즘에 대해 간략히 설명한다.

X-tree는 XML 문서의 버전간 변화를 효율적으로 탐지하기 위한 트리 모델로, X-tree의 각 노드는 XML 문서의 엘리먼트와 텍스트에 해당된다. 또한 각각의 노드는 트리 구조를 유지하기 위한 필드들 이외에 변화 탐지를 위한 필드들과 대응 및 대응 유형을 저장하기 위한 필드들을 갖고 있다. 변화 탐지를 위한 필드들로 nMD와 tMD가 있으며 nMD는 노드 자신에 대한 정보를 MD4(Message Digest 4) 알고리즘으로 해싱한 값을 ASCII 문자로 변환한 32바이트 문자열이다. tMD는 노드 자신의 nMD와 자식 노드들의 tMD의 결합을 MD4의 입력으로 해서 얻은 32바이트 문자열이다. 각 노드의 tMD는 정의에 따라 각 노드를 루트로 하는 트리에 속한 노드들의 정보뿐만 아니라 트리 구조에 대한 정보까지도 갖게 되어 트리 간의 대응 관계를 쉽게 찾을 수 있다. 한편, 대응 및 대응 유형을 저장하기 위한 필드로는 nPtr과 Op 필드가 있으며, nPtr 필드에는 대응하는 노드의 포인터를 저장하고, Op 필드에는 대응 유형을 저장하게 된다.

대응이란 XML 문서의 버전 s의 X-tree T_s에서 삭제되지 않은 각 노드에 해당하는 다음 버전 t의 X-tree T_t의 노드를 말한다. 이때 적용되는 대응 유형으로는 NOP, MOV 그리고 UPD가 있으며 이들은 각각 해당 노드에 변화 없음, 해당 노드가 다른 부모의 자식 노드로 이동, 해당 노드의 속성 혹은 텍스트 값의 변경을 의미한다. 한편, X-tree T_s와 T_t에서 삽입 혹은 삭제된 노드의 대응 유형에는 NULL 값이 할당된다. 본 논문 전체에 걸쳐 s와 t는 버전을 나타내는 일련번호를 뜻하며 이후 특별한 언급이 없는 한 s<t로 가정한다.

X-treeDiff 알고리즘은 XML 문서의 두 버전간 변화를 탐지하기 위해 두 버전에 해당하는 X-tree T_s와 T_t의 트리간 혹은 노드간의 대응을 tMD와 nMD를 이용하여 찾고, 이들 대응 관계로부터 편집 스크립트를 생성한다. **편집스크립트**란 두 버전 T_s와 T_t의 차이를 서술하는 것으로 $diff(T_s, T_t)$ 혹은 ε로 표현한다. 만약 s<t라면 $diff(T_s, T_t)$를 순방향 편집스크립트라 하고, s>t일 경우 $diff(T_s, T_t)$를 역방향 편집스크립트라 한다.

한편 본 연구에서의 편집 스크립트 ε는 대부분의 기존 연구들[3,6]과 같이 버전 T_s에서 버전 T_t를 생성해낼 수 있는 일련의 트리 **편집연산**들로 구성되며, 편집연산들은 위에서 설명한 대응 형태에서 얻어진다. X-treeDiff에서 지원하는 편집연산은 삽입, 삭제, 이동, 갱신연산으로 다음과 같이 정의된다.

· **삽입연산** 트리 T를 노드 N의 i번째 자식 노드로 삽입한다. 삽입연산은 T_t에서 대응되지 않은 노드들로부터 추출된다. 다음은 *Title* 엘리먼트를 nID가 .Actors[0].New[0].Actor[1].Movies[0]인 노드의 두 번째 자식 노드로 삽입하는 연산을 XML로 표현한 예를 보이고 있다.

```
<i pth=".Actors[0].New[0].Actor[1].Movies[0]"
   pos="2" opord="1">
        <Title>movie30<Title>
</i>
```

nID란 노드 N에 대한 식별자(identifier)로 축약하지 않은 XPath와 유사하게 정의된다. 이는 노드에 대한 식별자를 별도로 관리하는 다른 연구들[3]과는 구별되는 것으로 생성된 편집스크립트의 가독성(readability)과 노드의 식별자 관리에 드는 추가비용을 없애는 장점을 가지고 있다.

· **삭제연산** 노드 N을 루트 노드로 하는 트리를 삭제한다. 삭제연산은 T_s에서 대응되지 않은 노드들로부터 추출된다. 다음은 nID가 .Actors[0].Old[0].Actor[2].Movies[0].Title[2]인 노드를 삭제하는 연산의 예를 보이고 있다.

```
<d pth=".Actors[0].Old[0].Actor[2].Movies[0].Title[2]"/>
```

· **이동연산** 노드 N을 루트 노드로 하는 트리를 노드 M의 i번째 자식 노드로 이동을 나타내며 이는 X-tree의 대응 유형 MOV로부터 추출된다. 다음은 노드 .Actors[0].New[0].Actor[1]이 노드 .Actors[0].Old[0]의 세 번째 자식 노드로 이동하는 연산의 예를 보이고 있다.

```
<m spth=".Actors[0].New[0].Actor[1]"
   tpth=".Actors[0].Old[0]"
   pos="3" opord="2"/>
```

· **갱신연산** 노드 N의 텍스트의 내용을 *Text Value*로의 갱신을 의미하며 이는 X-tree의 대응 유형 UPD로부터 추출된다. 다음은 노드 .Actors[0].Old[0].Actor[0].Movies[0].Title[0]의 텍스트 내용이 movie23으로 갱신되는 예를 보이고 있다.

```
<u pth=".Actors[0].Old[0].Actor[0].Movies[0].Title[0]"
   tv="movie23"/>
```

3. 편집스크립트 생성

본 절에서는 X-treeDiff 알고리즘을 통해 X-tree T_s와 T_t간에 대응 관계가 생성되어 있다고 가정하고 이들 대응 관계로부터 편집스크립트를 생성하는 알고리즘을 설명한다.

편집스크립트 $diff(T_s, T_t)$를 구성하는 일련의 편집연산 $e_1, e_2, ..., e_k$들을 T_s에 차례로 적용하면 T_t를 생성하게되며, 이를 $\varepsilon(T_s) = T_t$로 표현한다. 편집스크립트 생성은 T_s에 차례로 적용하여 T_t를 생성할 수 있는 일련의 편집연산 $e_1, e_2, ..., e_k$를 추출해 내는 것으로 적용할 편집연산의 순서는 매우 중요한 의미를 갖게 된다[3].

X-treeDiff에서 편집스크립트 생성은 다음의 순서로 이루어지며, 추출된 편집연산들은 추출된 순서에 의해 적용된다.

T_s로부터 삭제와 이동연산의 추출 T_s의 각 노드를 *깊이우선탐색(DFS)의 역순*으로 방문하면서 대응 유형이 NULL인 노드로부터는 삭제연산을, 대응 유형이 MOV인 노드로부터는 이동연산을 추출하여 방문한 순서에 의해 편집연산을 생성한다. T_s로부터 얻어진 이동연산에는 이동 대상 노드의 nID만이 할당된다.

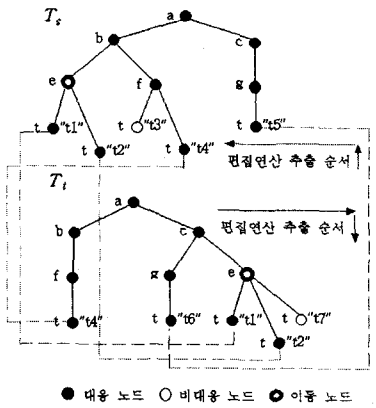

그림 1 X-treeDiff에서의 대응 예

T_t로부터 삭입 및 갱신연산의 추출과 이동연산의 추가 정보 할당 T_t의 각 노드를 *깊이우선탐색(DFS)*순으로 방문하면서 대응 유형이 NULL인 노드로부터는 삽입연산을, 대응 유형이 MOV인 노드로부터는 대응 정보를 이용하여 T_s에서 생성된 이동연산을 찾고, T_t의 정보를 통해 어디로 이동할 것인가에 대한 값 즉, 부모 노드의 nID와 형제간 순서를 이동연산에 할당한다. 이와 동시에 마지막으로 적용될 갱신연산을 추출하며 갱신연산끼리의 순서는 중요한 의미를 가지고 있지 않다.

그림 1은 X-tree T_s와 T_t 간의 대응 관계를 보이고 있다. 그림 1에서 레이블이 같은 노드들끼리 서로 대응하고 있으며, 노드 t의 경우 점선으로 그 대응 관계를 표시였다. 즉, T_s의 노드 .a[0].b[0].e[0]은 T_t의 노드 .a[0].c[0]의 2번째 자식 노드로 이동하였음을 볼 수 있으며, T_s의 노드 .a[0].c[0].g[0].t[0]의 텍스트 값 "t5"가 T_t에서는 "t6"로 바뀌었음을 알 수 있다. 그림 1에서 보이고 있는 X-treeDiff의 대응을 이용하여 구현된 프로토타입 시스템에서 실제로 생성한 편집스크립트는 그림 2와 같다.

```
<es fn="Smurf:/sample.xml" dc="08/31/03 16:25:23">
  <d pth=".a[0].b[0].f[0].t[0]"/>
  <m spth=".a[0].b[0].e[0]" tpth=".a[0].c[0]"
     pos="2" opord="1"/>
  <i pth=".a[0].c[0].e[0]" pos="3" opord="2">
    <t>t7</t>
  </i>
  <u pth=".a[0].c[0].g[0].t[0]" tv="t6"/>
</es>
```

그림 2 실제 편집스크립트 예

편집스크립트 연산자 편집스크립트의 정의에 따라 최종 버전 문서에 대한 X-tree를 T_n라하고, 이전 버전 T_{n-1}을 생성하기 위한 역방향 편집스크립트를 $\varepsilon^{-1}_{n,n-1}$이라 하면, $\varepsilon^{-1}_{n,n-1}(T_n) = T_{n-1}$이 성립한다. 이를 확장 적용하면 최초 버전 0은 $\varepsilon^{-1}_{0,1} \cdot \varepsilon^{-1}_{1,2} \cdots \cdot \varepsilon^{-1}_{n-1,n}(T_n)$이 됨을 쉽게 알 수 있다. 이때, 편집스크립트 연산자 $\cdot$은 편집스크립트의 순차적 적용을 의미한다. 따라서 버전 관리 시스템 구현에서 특정 시점 t에서의 문서는 $\varepsilon^{-1}_{t,t-1} \cdot \varepsilon^{-1}_{t-1,t-2} \cdots \cdot \varepsilon^{-1}_{n-1,n}(T_n)$을 통해 이루어진다.

4. 버전 관리 시스템 프로토타입 구현

XML 문서의 버전 관리 시스템의 프로토타입 구현은 VC++로 이루어졌으며, XML 문서의 저장은 eXcelon XML DB를 사용하였으며 구현 시스템의 구조는 그림 3과 같다.

그림 3의 Version Manager의 입력에서 볼 수 있듯이 구현된 시스템은 CVS[5]와 같이 최종 버전의 문서와 역방향 편집스크립트를 저장소에 저장하고 있다.

그림 3에 제시된 시스템은 Scheduler에 등록되어 있는 웹 문서를 특정 주기를 갖고 Crawler를 통해 얻어온다. 얻어온 문서를 저장소에 저장되어 있는 최종 문서와 X-treeDiff와 비교하여 변화가 발생하였을 경우, 얻어온 문서를 저장소에 저장하고, 편집 스크립트를 추가로 저장소에 저장한다.

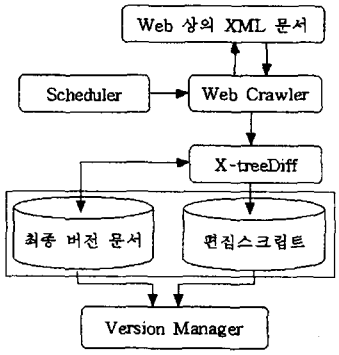

그림 3 구현 시스템의 구조

사용자가 Version Manager에 특정 시점의 문서를 요구하면, 요구한 시점의 문서를 얻기 위해 그 문서의 최종 버전과 적용시켜야할 편집스크립트들을 저장소로부터 얻어온 후 차례로 역방향 편집스크립트를 적용하여 요구한 문서를 생성·반환한다.

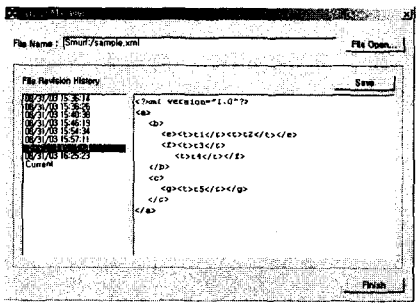

그림 4 Version Manager 실행 화면

그림 4는 구현된 Version Manager의 실행 화면을 보이고 있다. 그림 4에서는 현재 버전의 전전 버전 내용을 보여 주고 있으며, 이는 두 번의 역방향 편집스크립트를 적용시켜 얻어진다.

그림 5는 eXcelon XML DB에 저장되어 있는 각 시점별 편집스크립트들이 저장되어 있는 화면을 보이고 있다.

5. 결론 및 향후 연구 과제

X-treeDiff는 XML 문서의 두 문서간의 차이를 서술하는 편집스크립트를 작성한다. 본 논문에서는 X-treeDiff 알고리즘에서 편집스크립트를 생성하는 과정과 X-treeDiff를 통해 생성된 편집스크립트를 이용하여 문서의 버전을 관리하는 버전 관리 시스템의 프로토타입의 구현을 보였다.

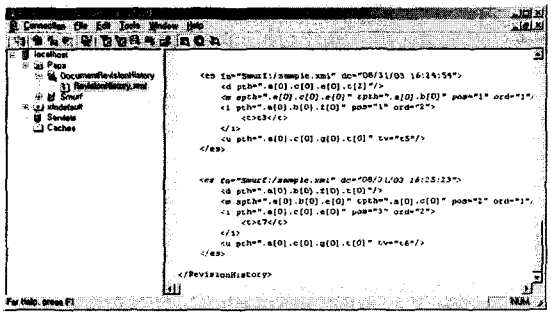

그림 5 eXcelon에 저장되어 있는 편집스크립트들

XML 문서 버전 관리 시스템의 프로토타입의 구현은 X-treeDiff 알고리즘의 검증과 텍스트 기반의 CVS와 같은 트리 구조 데이터 기반의 문서에 대한 버전 관리 시스템 개발에 대한 기반을 마련하였다는 중요한 의미를 갖는다. 현재 XML 문서 버전 관리 시스템에 요구되는 동시성 제어에 대한 연구와 편집 스크립트에 대한 질의에 대한 연구가 진행중이다.

참고문헌

[1] S.-Y. Chien, V. J. Tsotras and C. Zaniolo, "Copy-based versus edit-based version management schemes for structured documents," RIDE 2001, Proceedings, pp.95-102, 2001.

[2] 손충범, 오경근, 유재수, "버전을 지원하는 XML 저장관리 시스템 설계 및 구현," 정보처리학회 논문지 D 제10-D권 제1호, pp.13-22, 2003.

[3] A. Marian, S. Abiteboul, G. Cobena and L. Mignet, "Change-Centric Management of Versions in an XML Warehouse," Proceedings of the 27th VLDB Conference, pp.581-590, 2001.

[4] D. A. Kim and S. K. Lee, "Efficient Change Detection in Tree-Structured Data," HSI Conference, pp.675-681, 2003.

[5] CVS, Concurrent Versions System. http://www.cvshome.org.

[6] Y. Wang, D. J. DeWitt, J. Y. Cai, "X-Diff : An Effective Change Detection Algorithm for XML Documents," *19th ICDE*, 2003.