Farey Labeling : 계층적 데이터 관리를 지원하는 XML 데이터 레이블링 기법

배주호, 김학인°, 박 석
서강대학교 컴퓨터공학과
juho@sogang.ac.kr hikim@sogang.ac.kr spark@sogang.ac.kr

Farey Labeling : XML data labeling scheme for Hierarchical data management

Juho Bai, Hak In Kim°, Seog Park
Dept. of Computer Science & Engineering, Sogang University

요 약

본 논문에서는 계층적 데이터를 관계형 데이터베이스 시스템에 저장하기 위한 요구사항을 만족하는 XML 레이블링 기법으로서 Farey Sequence 를 응용한 Faray 레이블링 기법을 제안한다. 이는 일반적인 동적 삽입연산 외에 형제간 노드 사이에 새 노드를 삽입할 경우 추가적인 레이블의 사이즈 증가 없이 기존노드의 리레이블링이 없고, 말단 노드의 삭제시 레이블의 재사용이 원할하며, 부모와 자식 사이에 새 노드를 삽입하는 경우가 빈번한 계층적 데이터 관리 시에 리레이블링을 최소화 할 수 있는 방법이다. 기존 XML 레이블링 기법이 부모 자식사이에 삽입 연산을 하는 경우 하위의 모든 노드를 리레이블링 해야 하는것에 비하여 본 기법은 오직 1개의 하위 노드만을 리레이블링 하기 때문에 해당 경우의 계층적 데이터의 동적 삽입 시 하위 노드의 개수에 상관없이 일정하게 연산시간을 유지할 수 있다.

1. 서 론

XML labeling 기법은 주어진 데이터를 초기 labeling 하는 정적 labeling 방법[1]에서부터 labeling 된 구조에 새로운 노드를 삽입 할 경우 다른 노드를 재작성 없이 그대로 사용하면서 새 노드의 label을 작성하는 방향[2][3][4]으로 연구가 진행되어 왔다. 이러한 방법 중에서 전위-기반 기법은 상위의 노드 레이블에 신규 노드값을 뒤에 붙이는 방법으로 구현하여 XML labeling에 사용된다. 이 중, OrdPath[3]는 Microsoft SQL Server 2005 for execution plan optimiztion에 구현[5]되었으며, 최근에는 Microsoft SQL Server 2008에서 계층형 구조의 데이터를 저장하는 방식[6]으로 사용되고 있어서 계층형 자료구조를 데이터베이스 시스템에서 효율적으로 관리할 수 있는 방법을 제시하고 있다.

일반적으로 테이블 형태의 데이터 베이스에 계층형 데이터를 저장하기 위해서는 **그림 1** 처럼 각각의 행 데이터에 자신의 부모의 위치를 저장하여 연결하는 방식을 사용한다. 이는 여러 개의 데이터가 동일한 부모를 가지고 있는 경우나 새로운 데이터가 기존 계층구조에 삽입되더라도 쉽게 그 구조를 저장할 수 있는 방식이다. 부모/자식 방법으로 일컬어지는 이 방법을 이용하면 계층의 섹션간 쿼리가 거의 일어나지 않고 구조의 단일 지점만 쿼리를 하는 경우에는 우수한 성능을 보인다. 하지만 각각의 데이터가 부모위치의 값을 가지고 연결되어 있기 때문에 계층간의 경로가 들어가는 쿼리에 대해서는 그 경로의 깊이만큼 반복하여 데이터 리스트를 검색해야 하는 단점이 있다. [6]

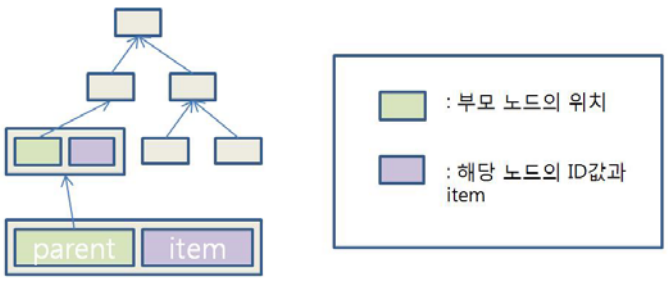


그림 1 부모/자식 방법의 계층형 데이터 저장구조.

그러나 전위기반 XML 레이블링 기법[3][7]의 레이블을 계층형 데이터로 저장하는 경우에는 각각의 레이블이 계층의 경로를 가지고 있기 때문에 단일 노드간의 비교만을 가지고 계층의 부모-자식 관계나 조부-손자관

계 또는 형제관계를 파악할 수 있어서 쿼리연산이 부모/자식 방법의 저장기법보다 매우 빠른 장점이 있다. 이러한 장점에 비하여 레이블링 기법의 경우 부모/자식 데이터 저장 기법에 비하여 동적 데이터 삽입이 제한되는 경향이 있는데, 이는 XML labeling scheme이 새 노드의 입력시 주변 노드를 재작성해야 하는 경우 이에 드는 시간이 성능에 영향을 주기 때문이다.

일반적인 계층형 데이터 구조에서 새로운 데이터가 기존구조에 추가되는 양상은 다음의 3가지 형태로 요약할 수 있다.

1) 말단 노드로 새 노드를 추가하는 경우

2) 말단 노드가 있는 상태에서 그 노드의 왼쪽이나 오른쪽 또는 말단의 양 노드 사이에 새 노드를 추가하는 경우

3) 부모노드와 자식노드 연결사이에 새로운 노드가 삽입되는 경우

예를 들어, **그림 2** 에서처럼 공정제어용 컨트롤러 센서 각각에 노드를 부여하고 각각의 센서에서 실시간으로 스트림 데이터로 센서값을 받아오는 데이터 구조를 RDBMS에 구현한다고 하자. 이를 계층형 노드형태로 도식화 하면 **그림 3** 과 같이 표현 할 수 있다. 이러한 경우 이 구조에 새로운 컨트롤러나 센서를 삽입하려 한다면 주위의 노드에 영향을 최소한으로 주는 것이 성능을 위해서 유리하다. 만약 이러한 구조에 MSSQL 2008의 hid datatype[6]을 쓰는 경우 hid가 ordpath[3]의 레이블링 기법을 기반으로 구현되므로 앞의 3가지 삽입 형태 중 첫번째와 두번째 경우 기존 레이블의 재작성 없이 노드 삽입이 가능하다. 하지만 **그림 3**의 화살표 위치에 새 노드를 삽입하는 경우에는 위에서 언급한 3가지 형태중 마지막 형태로써 그 하단의 영향을 받는 모든 부위의 노드의 레이블을 재작성 하여야 하며 이와 연결된 다른 하위-그래프의 노드 레이블 값도 변경해야 하는 문제가 있다.

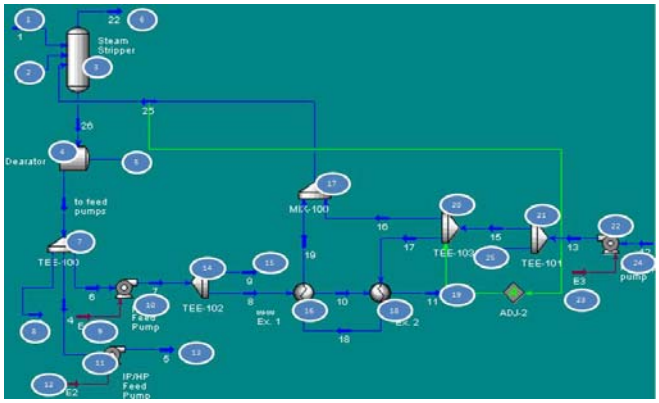


그림 2 Hysys 공정제어 흐름 설계

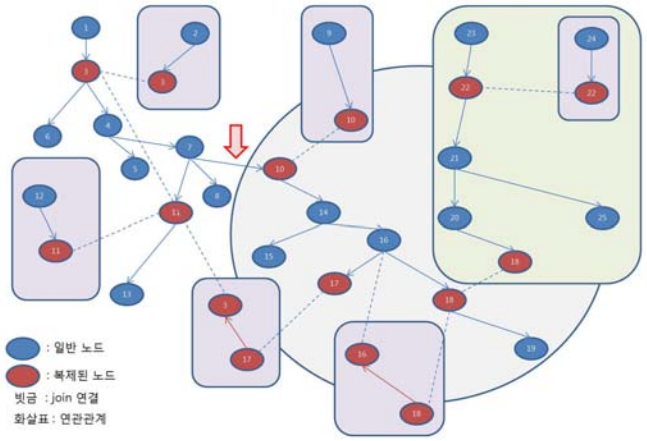


그림 3 그림.2 의 모델링 그래프

따라서 계층형 데이터를 RDBMS에 저장하는 방법[8][9]으로 XML labeling scheme을 효과적으로 사용하려면 해당 기법이 위의 3가지 형태의 삽입 방식에 대하여 최소한의 리레이블링을 전제로 구현되어야 한다. 이번 논문에서는 기존의 동적 레이블링의 장점을 포함하면서 계층적 데이터 구조 형태를 동적으로 구성하기 위하여 필수적인 삽입연산중의 하나인 부모 노드와 자식 노드 연결 사이에 새로운 노드가 삽입되는 경우에도 부모/자식 데이터 저장형태처럼 오직 1개의 추가적인 리레이블링만을 사용하는 farey 레이블링을 제안한다.

2. The Farey Labeing Scheme

먼저, Farey Sequence를 응용하여 노드 레이블링 하는 방법에 대하여 설명한다. Farey Sequence 의 정의[10][11]는 다음과 같다.

정의 Farey Sequence의 정의

Farey sequence $F(n)$ 은 $\forall j$ 에 대하여 $b_j \le n$ 인

$[0..1]$범위의 $F(n)=\left\{\frac{0}{1}<\frac{a_0}{b_0}<.....\frac{a_p}{b_p}<\frac{1}{1}\right\}$ 형태의 유리함수 수열이다. Farey Sequence가 위와 같이 정의 되었을 때, 우리는 $\frac{a_j}{b_j}<\frac{p}{q}<\frac{a_{j+1}}{b_{j+1}}$ 와 같이 $q \le n$ 인 범위내의 $\frac{p}{q}$ 형태의 유리분수를 찾을 수 없음이 성립한다. 따라서,

정리

Farey sequence $F(n)$에 대하여, $0 \le j \le p$ 에서 다음의 등식이 성립한다.:

$a_{-1}=0$, $a_{p+1}=1$, $b_{-1}=1$, $b_{p+1}=1$ 일 때,

$$\frac{a_{j-1}+a_{j+1}}{b_{j-1}+b_{j+1}}=\frac{a_j}{b_j}$$

이를 응용하여 각 노드 레이블을 깊이 부분과 형제 순서 부분 2개로 나누고 각각의 부분이 정수이므로 이를 분모와 분자의 순서쌍으로 표현하면 **그림 4** 과 같다.

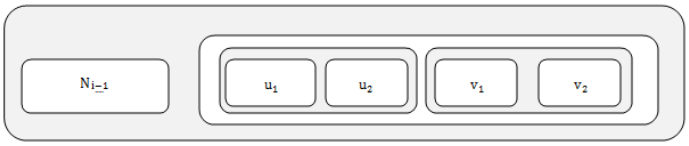


그림 4 Farey labeling Structure

그림 4 의 Ni-1은 임의의 노드 Ni 에 대하여, 선행하는 전위의 노드 레이블을 의미한다. U1 과 U2 는 해당 노드의 깊이 위치를 분모 분자 값의 순서쌍으로 나타낸 것이며, V1 과 V2 는 문서 순서값의 정수 값을 분모 분자의 순서쌍 값으로 나타낸 것이다.

2.1 초기 레이블링과 신규 노드 입력 연산

초기 레이블링의 경우 정적 레이블링 방식인 Dewey 방식을 따르되, 해당 노드값은 깊이값과 순서값을 같이 표기하고, 각각의 값은 분모를 1로하는 정수쌍으로 표현한다. 원래의 Dewey 방식의 경우 깊이값을 따로 기재하지 않아도 구분자의 개수를 가지고 그 깊이를 알 수 있는데, 각 노드별로 깊이값을 저장하게 되면 부수적인 저장공간이 필요할 수 밖에 없다. 이러한 문제를 해결하기 위하여 2bit의 데이터 타입 구분용 Flag를 노드 구조 앞에 붙여서 인코딩 시 저장 타입으로 활용한다.

2.1.1 단말 노드 삽입

XML tree 구조의 XML 노드 말단에 새 노드를 추가하는 경우, 부모 노드 레이블의 후위에 새로운 노드값을 덧붙여서 기록한다.

알고리즘 1 단말노드 삽입
Input : L_{i-1}
Output : $L_i = L_{i-1}$ + ".[(u_1 . u_2) . (v_1 . v_2)]"
Operation : $u_1 = 1$, $u_2 = L_{i-1}$ 의 깊이값 + 1, $v_1 = v_2 = 1$

2.1.2 특정 단말 노드 왼쪽 또는 오른쪽에 삽입

특정 단말 노드의 왼쪽 또는 오른쪽에 다른 형제 노드가 없는 경우 삽입연산을 하는 경우 해당 노드의 문서 순서값에 증감값을 연산하여 기록한다.

알고리즘 2 단말노드 옆에 형제 노드 삽입
Input : $L_i = L_{i-1}$ + ".[(u_1 . u_2) . (v_1 . v_2)]"
Output : $L'_i = L_{i-1}$ + ".[(u'_1 . u'_2) . (v'_1 . v'_2)]"
Operation : 단말노드 L_i의 왼쪽에 새 형제 단말노드 L'_i를 넣는 경우 :
$u'_1 = u_1$, $u'_2 = u_2$, $v'_1 = v_1$, $v'_2 = v_1 - 1$
단말노드 L_i의 오른쪽에 새 형제 단말노드 L'_i를 넣는 경우
$u'_1 = u_1$, $u'_2 = u_2$, $v'_1 = v_1$, $v'_2 = v_1 + 1$

2.1.3 형제관계인 두 노드 사이에 새 노드를 삽입

두개의 노드 사이에 새 노드를 삽입하는경우 레이블의 전위부분인 부모경로와 깊이값을 동일하게 할당하면서 문서순서를 지키기 위하여 farey sequence를 이용하

여 양 형제노드의 문서순서값을 더해 새 노드의 레이블 문서순서값을 구한다.

알고리즘 3 형제관계인 두 노드 사이에 새 노드를 삽입하기

두 노드를 S_1 , S_2 라 하고 새 노드를 S_3 라 하면 각 노드 레이블은 다음과 같다.

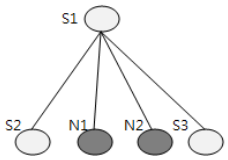
Input : $S_1 = L_{i-1} + \text{“.[(} u_1 \text{ . } u_2 \text{) . (} v_1 \text{ . } v_2 \text{)]”}$

$S_2 = L_{i-1} + \text{“.[(} u'_1 \text{ . } u'_2 \text{) . (} v'_1 \text{ . } v'_2 \text{)]”}$

Output : $S_3 = L_{i-1} + \text{“.[(} u''_1 \text{ . } u''_2 \text{) . (} v''_1 \text{ . } v''_2 \text{)]”}$

Operation : $u''_1 = u'_1 = u_1$, $u''_2 = u'_2 = u_2$,

$v''_1 = v'_1 + v_1$, $v''_2 = v'_2 + v_2$



S1 : L_{i-1}
S2 : L_{i-1} . [(1 . i) . (1 . 2)]
S3 : L_{i-1} . [(1 . i) . (1 . 3)]
N1 : L_{i-1} . [(1 . i) . (2 . 5)]
N2 : L_{i-1} . [(1 . i) . (3 . 8)]
S1,S2,S3 : 초기노드, N1,N2 : 신규 노드 삽입순서

그림 5 형제노드 사이에 새노드 삽입

2.1.4 부모-자식 관계인 두 노드 사이에 새 노드를 삽입

부모와 자식 관계인 두 노드 사이에 새 노드를 넣는 경우에는 기존에 존재하는 동일 부모하의 형제 노드간의 형제관계 속성을 유지하기 위하여 불가피하게 해당 자식노드 1개의 레이블을 재작성 해야만 한다.

따라서, 부모 노드 P1 와 자식 노드 C1 사이에 새 노드 N1 를 추가한다면 기존의 자식 노드인 Ci 의 깊이 순서쌍을 2배 하여 레이블을 재작성 한다. 새 노드 N1 는 부모 노드 레이블값을 그대로 할당하고 추가로 P1 와 C1 의 레이블 마지막 노드 부분의 깊이값 순서쌍을 더하여 N1의 마지막 부분 깊이값으로 할당하는데, 만약 각 순서쌍의 분모 분자값이 서로 소가 아닌경우 해당 순서쌍을 두 수의 최대공약수로 나눈 몫을 이용하여 더해준다, 문서 순서값의 경우 C1의 문서 순서값을 그대로 가져온다.

알고리즘 4 부모-자식관계의 두노드 P_1, C_1 사이에 새 노드 N_1 삽입

Input : $P_1 = L_{i-1} + \text{“[(} u_1 \text{ . } u_2 \text{) . (} v_1 \text{ . } v_2 \text{)]”}$

$C_1 = L_{i-1} + \text{“.[(} u_1 \text{ . } u_2 \text{) . (} v_1 \text{ . } v_2 \text{)]”} + \text{“[(} u'_1 \text{ . } u'_2 \text{) . (} v'_1 \text{ . } v'_2 \text{)]”}$

Output : $N_3 = L_{i-1} + \text{“[(} u_1 \text{ . } u_2 \text{) . (} v_1 \text{ . } v_2 \text{)]”} + \text{ “.[(} u''_1 \text{ . } u''_2 \text{) . (} v''_1 \text{ . } v''_2 \text{)]”}$

Operation : If (GCD(u_1, u_2) != 1) or (GCD(u'_1, u'_2) != 1)

Then

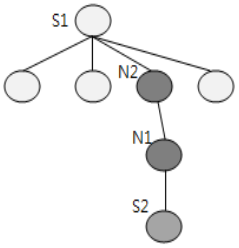
$u_1 = u_1 / \text{GCD}(u_1, u_2)$, $u_2 = u_2 / \text{GCD}(u_1, u_2)$,

$u'_1 = u'_1 / \text{GCD}(u'_1, u'_2)$, $u'_2 = u'_2 / \text{GCD}(u'_1, u'_2)$

End If

$u''_1 = u'_1 + u_1$, $u''_2 = u'_2 + u_2$, $v''_1 = v'_1$, $v''_2 = v'_2$

$u'_1 = 2^* u'_1$, $u'_2 = 2^* u'_2$



S1 : L_1 . [(1 . 2) . (1 . 4)] = L_2
S2 : L_2 . [(2 . 6) . (1 . 3)]
N1 : L_1 . [(4 . 10) . (1 . 3)]
N2 : L_1 . [(3 . 7) . (1 . 3)]
S1,S2 : 초기노드
N1,N2 : 신규 노드 삽입순서
S2 와 N1은 각각 N1과 N2 삽입 시 재작성.

그림 6 부모-자식 관계의 노드 사이에 새 노드 삽입

2.2 레이블 속성을 이용한 노드 간의 관계 파악

두개의 노드간의 관계는 크게 3가지로 분류 할 수 있으며, 이에 따른 관계파악은 정적 레이블링 기법인 Dewey 레이블링 기법[12][7][1]에 기반한다. 다만 Farey labeling 의 경우 부모-자식 사이에 노드가 입력될 경우 입력된 노드의 바로 하위의 자식노드가 자신의 노드 깊이값을 2배 하여 저장하므로 증조부-자손 관계와 부모 자식 파악시에는 각 노드의 레이블 값 중 깊이값에 해당하는 부분의 두 숫자값의 최대공약수를 각 수로 나눈 몫의 값을 실제 레벨값으로 이용한다. **그림.4**를 기반으로 다음 3가지 경우에 대하여 설명한다.

2.2.1 증조부 - 자손 관계 파악

두 개의 노드 레이블 N1 , N2 에 대하여 N2의 레이블안에 N1의 레이블이 포함되어 있고, N2레이블이 N1 레이블의 후위에 2개 이상의 단일 노드 레이블을 추가로 가지고 있으면 이 두 노드는 증조부 - 자손 관계이다. 이 때 , N1 이 N2의 증조부 이며 N2는 자손인 관계가 성립한다.

2.2.2 부모 - 자식 관계 파악

두 개의 노드 레이블 N1 , N2 에 대하여 N2의 레이블안에 N1의 레이블이 포함되어 있고, N2레이블이 N1레이블의 후위에 한 개의 단일 노드 레이블을 추가로 가지고 있으면 이 두 노드는 증조부 - 자손 관계이다. 이 때 , N1 이 N2의 부모 이며 N2는 자식인 관계가 성립한다..

2.2.3 형제 관계 파악

두 개의 노드 레이블 N1, N2 가 레이블 후위의 한 개의 단일 노드 레이블 값만 다르고 나머지 상위 경로가 동일한 경우에 이 두 노드는 형제 관계가 성립한다.

XML 문서의 특성상 형제관계일 경우 두 노드간에 순서가 존재하게 되는데 **그림 4** 와 같은 구조에서

$$N1.V2 * N2.V1 - N1.V1 * N2.V2 > 0$$

일 경우 N1 이 문서상의 순서에서 N2 보다 앞에 위치한다. 만약 위의 값이 0보다 작을 경우에는 N2가 N1보다 문서상의 순서에서 앞에 위치하게 된다.

3. Encoding scheme

Farey labeling 인코딩은 각각의 숫자값을 저장하기 위하여 ordpath에서 쓰인 호프만 알고리즘 기반의 인코딩 기법[3]을 사용한다. 하지만 Farey의 경우 **그림 4** 와같이 초기 삽입시 기본값이 들어가는 데이터가 있기 때문에 해당 노드의 레이블 인코딩값 앞에 상황 별로 4가지 타입을 구분할 수 있도록 헤더 형식의 2bit를 할당하여 **그림 7** 의 형태로 분류하여 인코딩하도록 하였다.

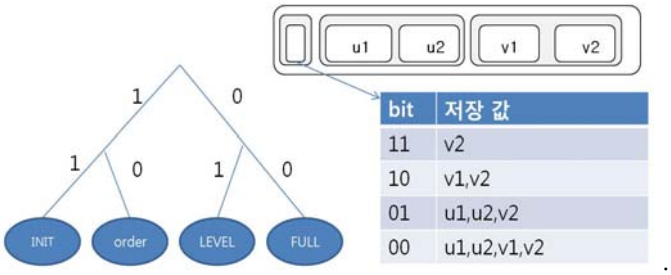


그림 7 인코딩시 데이터형식에 따른 저장 값의 분류

4. 실험

실험 환경은 **표 1** 과 같으며, hid[6]의 기반이 되는 기법인 ordpath[3]와의 XML 데이터집합의 초기 레이블링 저장 사이즈를 비교 하였다. 또한, 3가지 타입의 노드 삽입형태 중 부모와 자식 사이에 새 노드를 삽입하는 경우 다른 기존 노드의 리레이블링에 걸리는 시간을 측정하였는데, 말단 노드에 입력하거나 형제노드의 왼쪽 , 오른쪽 또는 사이에 넣는 경우는 ordpath labeling 과 farey labeling 둘다 추가적인 레이블링이 필요하지 않으므로 실험에서 제외하였다.

실험에 쓰인 XML 문서는 **표 2** 의 데이터집합을 이용하였다.

표 1 실험에 쓰인 기자재

명칭	기종	비고
운영체제	Windows Server 2008 Standard SP2	
기기	AMD 64 3500+ 2.20GHZ	3G 32bit-OS
toolkit	Visual Studio 2008 C#	
DBMS	MSSQL Server 2008	

표 2 실험에 쓰인 XML 데이터집합[13][14]

	Size(MB)	Tot.num	Max-fan	Avg-fan	Max-dep	Avg-dep
Xmark	113	1666315	25500	3242	12	6
Nasa	23.8	476646	2435	255	10	7
Treebank	85.4	2437666	56384	1623	36	8

4.1 초기 레이블링 사이즈 비교

초기 레이블링 사이즈는 XML 문서와 비교하여 해당 계층데이터를 테이블로 저장할 때 드는 저장 비용을 가지고 비교 하였다. 저장하는 방식에 따라 부모/자식으로 저장하여 연결하는 방식과 ordpath 기법을 이용한 방식 그리고 본 논문의 Farey 레이블링 기법을 비교하였는데, 실험 결과는 **그림 8** 과 같다. 초기 레이블링의 경우 Farey 레이블링이 부모/자식으로 저장하는 방법보다는 사이즈를 적게 차지하며 ordpath 보다는 약 5%정도의 추가 공간이 더 필요한 것으로 나타났다.

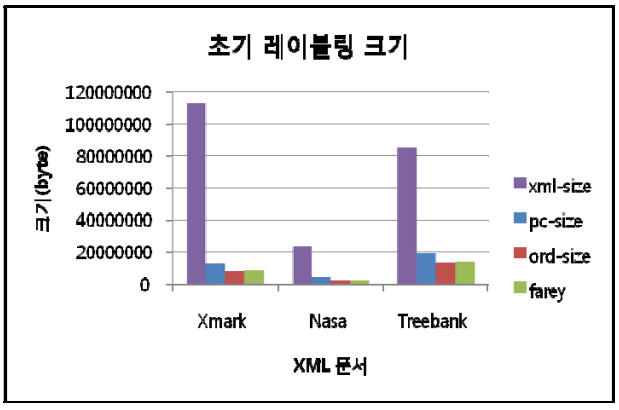


그림 8 초기 레이블링 크기 비교

4.2 부모- 자식 사이에 노드 입력시 레이블링 사이즈 비교

부모와 자식 사이에 신규 노드가 입력되는 경우에는 ordpath 레이블링 기법과 Farey 레이블링 기법의 리레이블링에 걸리는 시간을 재서 평가하였다. 단일노드가 XML문서상의 랜덤한 위치에 삽입되는 경우 그 횟수에

따른 평균값을 가지고 리레이블링에 드는 소요시간을 계산하였는데, 문서의 형태에 따라 Farey가 단일 하위 노드의 리레이블링만을 요구한데 비하여 ordpath의 경우 하위의 모든 노드를 리레이블링 해야 하므로 시간간격에서 크게 차이가 남는 것을 **그림.9** 를 통하여 알 수 있다.

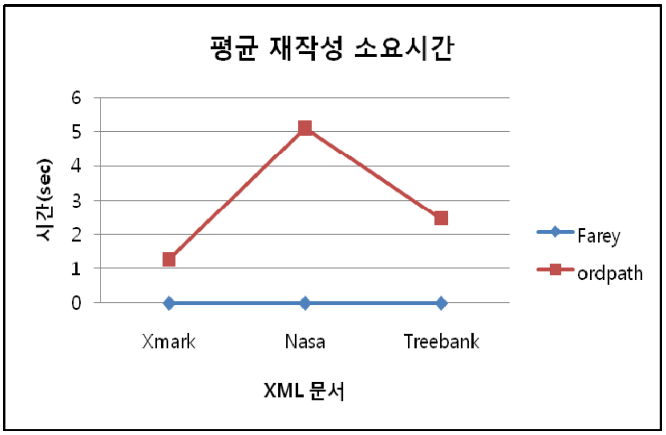


그림 9 부모-자식 사이에 노드입력 평균 재작성 소요시간

5. 결론과 향후 과제

기존의 동적 레이블링 기법의 경우 계층적 데이터를 저장하는 수단으로 사용하는 데에는 다양한 계층적 데이터의 계층구조 형성방법에 유연하게 대응하기 어려운 단점이 있었다. 본 논문에서는 XML 레이블링을 계층적 구조의 데이터를 RDBMS 상에서 저장할 때 필요한 요구사항을 충족시키면서 레이블링 기법의 장점을 그대로 유지할 수 있는 farey 레이블링을 제안하여 XML 레이블링을 계층적 구조의 데이터 관리에 좀 더 유연하게 적용시키는 방법을 보였다. 물론 기존의 레이블링 기법에 비하여 5%정도의 추가적인 초기 저장공간이 필요하고 반복적인 삽입 연산시 데이터의 저장공간이 노드의 리레이블링에 필요한 영역보다 늘어날것으로 예상되지만, 그럼에도 불구하고 이 방법을 사용함으로써 쿼리연산에 필요한 성능을 기대할 수 있을 것으로 보인다. 앞으로는 접근제어 관리나 물류 수송 관리 등 계층적인 구조를 가진 데이터를 사용할 수 있는 환경에서 빠르게 질의처리가 필요한 경우 이 기법을 응용하여 기존 방법과 비교하고 성능향상을 도모할 계획이다.

참고문헌

[1] E. Cohen, H. Kaplan, and T. Milo, "Labeling Dynamic XML Trees," In SPDS, 2002

[2] I. Tatarinov, S. Viglas, K. S. Beyer,J. Shanmugasundaram, E. J. Shekita, and C. Zhang, "Storing and Querying Ordered XML Using a Relational Database System," In SIGMOD, 2002.

[3] O'Neil, P.E., O'Neil, E.J., Pal, S., Cseri, I., Schaller, G.,Westbury, N." ORDPATHs:Insert-friendly XML node labels," 2004 ACM SIGMOD Conference on the Management of Data, pp. 903–908, 2004.

[4] Liang Xu, Tok Wang Ling, Huayu Wu, Zhifeng Bao, "DDE: From Dewey to a Fully Dynamic XML Labeling Scheme," SIGMOD'09, 719-730, 2009.

[5] Sans, V., Laurent, D. "Prefix based numbering schemes for XML: techniques, applications and performances," VLDB '08, pp. 23–28, 2008.

[6]http://technet.microsoft.com/ko-kr/library/bb677173.aspx

[7] Tatarinov, I., Viglas, S., Beyer, K.S., Shanmugasundaram, J., Shekita, E.J., Zhang, C. "Storing and querying ordered XML using a relational database system," SIGMOD '02, pp. 204–215, 2002.

[8] Khaing, A., Thein, N.L, "A Persistent Labeling Scheme for Dynamic Ordered XML Trees," Conf. on Web Intelligence, pp. 498–501, 2006.

[9] Gabillon, A., Fansi, M, "A persistent labelling scheme for XML and tree databases," SITIS Conf., pp. 110–115, 2005.

[10]Augustin-Louis Cauchy,Exercises de mathématique Vol. I, 114 -116, 1840.

[11]http://www.emis.de/journals/AUA/acta5/survey3.ps_pages1-20.pdf

[12] S. Abiteboul, S. Alstrup, H. Kaplan, T. Milo, and T. Rauhe. "Compact Labeling Scheme for Ancestor Queries," SIAM J. Comput., 2006.

[13] XMark - An XML Benchmark Project. http://monetdb.cwi.nl/xml/downloads.html.

[14] University of Washington XML Repository. http://www.cs.washington.edu/research/xmldatasets/.