XML 데이터 공유를 위한 리스트 잠금 프로토콜

이 은 정†

요 약

XML 트리를 공유하면서 여러 사용자가 동시에 수정할 수 있게 하기 위해서는 부분 트리의 삽입/삭제가 공유될 수 있어야 한다. 이를 위해서는 트리 데이터의 구조 변경 행위에 대한 동시성 제어가 가능해야 한다. 본 논문에서는 DTD 문서 타입 정보를 이용하여 XML 트리의 반복부에 대해서만 부분 트리의 삽입/삭제가 가능한 *리스트 데이터 공유 모델*을 제안한다. 제안된 리스트 데이터 공유 모델은 구조 변경 행위의 적용 결과가 항상 유효하며, 여러 사용자가 동시에 접근하는 경우에도 문서의 유효성을 보장할 수 있다. 리스트 데이터 공유 모델에서 반복부 자식 노드 리스트를 잠금의 대상으로 하는 *리스트 잠금 프로토콜*을 제안하였다. 이 잠금 프로토콜은 기존의 방법들과 비교하여 훨씬 간단하면서 적은 수의 잠금 객체만을 다루며, 높은 접근성을 가능하게 한다. 일반적으로 공유되는 XML 트리의 삽입 및 삭제는 흔히 반복부 데이터에 대해서 적용되는 경우가 많으므로 제안된 모델은 터미널 노드 데이터 값에 대한 기존의 접근제어 방법과 함께 사용하였을 때 효과적인 데이터 공유 방법을 제공할 수 있을 것으로 기대된다.

List Locking Protocol for XML Data Sharing

Eunjung Lee†

ABSTRACT

For sharing XML data by many users, a way of concurrency and access control is required for isolating update actions such as inserting and deleting subtrees. Exisiting locking mechanisms as 2PL or MGL suffer low concurrency when applied to tree structures. In this paper, *list data sharing model* is proposed based on the semantics expressed in DTD. In this model, tree updating actions such as inserting and deleting subtrees are considered only for the repetitive parts of XML trees. The proposed model guarantees that the result XML tree after applying a tree updating action is always valid, even when multiple users access the tree at the same time. Also, a new locking mechanism called *list locking protocol* is proposed. The new locking protocol is expected to show better accessiility with less number of locking objects compared to the Helmer's OO2PL model. Since update actions on a shared XML tree usually applied to the repetitive parts of the tree, the proposed model is expected to provide a useful way for efficient data sharing when combined with previous locking methods on terminal node data.

키워드 : XML, DTD, 공유 데이터(Shared Data), 잠금 프로토콜(Locking Protocol), 동시성 제어(Concurrency Control)

1. 서 론

인터넷 기반의 많은 응용이 XML을 사용하면서 XML 데이터를 공유하는 어플리케이션이 점차 많아질 것으로 기대된다. 여러 사용자들이 XML 문서를 동시에 접근하고 수정하는 경우 동시성 제어 방법이 필요하다.

잠금은 데이터베이스에서 동시 접근 제어를 위한 기본적인 장치이다. 트랜잭션이 요청할 수 있는 잠금의 단위에 여러 단계를 두어 접근성을 향상시키려는 다단계 잠금 방식(Multiple Granularity Locking ; 이하 MGL)은 관계형 데이터베이스는 물론 객체지향 데이터베이스에서도 많이 적용되고 있다. 그러나 XML 트리에 MGL을 적용하는 경우 중간 노드에 대한 잠금은 하부 모든 노드에 대한 잠금으로 처리되어 접근성이 떨어진다[4].

최근, XML 문서를 대상으로 동기화에 대한 연구가 다시 관심을 끌고 있다. 몇가지 잠금 기반의 XML 데이터 접근 제어 방법이 제안되었는다[8, 10, 13]. 그러나 이들 연구는 대부분 공유 데이터에 대한 질의 처리의 효율성을 목표로 하여 여러 노드에 대한 읽기 접근과 쓰기 및 트리 구조 변경 행위의 동시 처리에 주로 관심을 가진다. 그 결과 트리 구조 변경 행위에 대한 효율성은 크게 고려되지 못하였다. 또한 기존 연구에서는 DTD와 같은 XML 문서의 구조 정보를 잠금 모델에 이용하지 않고 있는데, 본 연구는 구조 변경 행위의 동시성 제어를 위하여 DTD 구조 정보와 문서의 유효성을 고려하고자 하는 동기에서 출발하였다.

본 논문에서는 공유 구조 변경 행위의 동시성 제어를 위하여 리스트 데이터 공유 모델을 제안한다. 이 모델은 DTD에 표현된 구조 정보를 이용하여 구조 변경 행위가 XML

※ 경기대학교 교내연구과제(2004 021)의 지원을 받았음.
† 정 회 원 : 경기대학교 정보과학부 교수
논문접수 : 2004년 7월 1일, 심사완료 : 2004년 10월 1일

트리에서 반복부에 대해서만 허용되도록 제한한다. 그 결과 이 모델은 여러 사용자에 의한 구조 변경 행위 요청의 경우에도 결과 트리가 항상 유효함을 보장한다.

제안된 리스트 데이터 공유 모델에 대한 새로운 다단계 잠금 방식을 제안한다. 이 프로토콜은 반복부 자식 노드의 리스트를 잠금의 단위로 사용하여, 부분트리의 삽입과 삭제 시에도 다른 자손 노드에 대한 접근을 허용할 수 있게 한다. 반복부만을 공유 대상으로 삼았으므로 이 프로토콜은 기존에 제안된 XML 트리의 잠금 방식과 비교하여 더 적은 수의 잠금 객체를 이용하여 나은 접근성을 보장할 수 있다.

기존에 문헌에서 소개된 XML 데이터 공유 유형을 조사한 결과 공유 XML 데이터에 대한 부분트리 삽입 삭제를 반복부에 한정되도록 모델링하는 것이 가능한 것으로 보인다. 또한 데이터에 대한 읽기/쓰기는 터미널 노드에 대해서만 허용되므로 본 공유 모델이 기존의 데이터 읽기 쓰기를 위한 MGL과 결합되어 사용된다면 구조 변경 행위에 대한 효율적인 동시성 제어를 가능하게 할 것으로 기대된다.

이 논문의 구조는 다음과 같다. 우선 아래 1절에서는 본 논문의 연구와 관련된 기존 연구를 살펴보고 2장에서는 본 논문에서 다루고자 하는 데이터 공유 사례를 소개하고 3장에서는 반복부 리스트 데이터에만 구조 변경 행위를 허용하는 리스트 공유 모델을 제시한다. 4장에서 리스트 잠금 프로토콜을 정의하고 다른 잠금 프로토콜과 비교한다. 그리고 5장에서 결론을 맺는다.

1.1 관련 연구

다단계 잠금 프로토콜은 RDB 뿐 아니라 OODB에서도 많이 연구된 주제이다. 특히, OODB에서는 DAG를 이용하여 객체 그룹을 잠금 단계로 이용할 수 있는 방법이 제안되었다. 리 등은 복합 객체에 대한 다단계 잠금 방법을 제안하였는데, 이들의 방법은 객체들이 서로 겹칠 수 있는 임의의 집합을 처리할 수 있다[17].

XML 트리는 DAG이기보다는 계층적 구조를 가지는 트리이다. XML 문서에서 부모 자식 관계 뿐 아니라 ID와 IDREF 속성을 노드 간의 관계로 DAG로 모델링하여야 할 것이다. 그러나 본 논문에서는 XML 트리를 트리 구조로 취급하여 있다. 트리 잠금에 대한 기존 연구에서는 구조나 데이터가 변할 수 있는 공유 트리에 대한 동시 접근 제어를 다루었다. 그러나 그들은 부분트리의 삭제를 허용하지 않는다. 또한 이들 방법은 다단계 접근 제어를 이용하지 않고 노드만을 잠금의 단위로 삼아 부분트리를 취급할 수 없었다.

트리의 구조 변경 행위를 허용하기 위한 잠금 기반 접근 제어 방법으로 최근 제안된 OO2PL[13, 14]과 경로 잠금 방법[8]이 있다.

Helmer 등은 트리 구조 변경에 대한 잠금 프로토콜로 OO2PL 방법을 제안하였다[13, 14]. 트리 방문 및 구조 변경을 노드 간의 포인터에 대한 접근으로 보고 OO2PL은 이들 포인터에 잠금을 적용한다. 이들은 문서 구조 정보를 잠금과 공유 모델에 이용하지 않았다. OO2PL은 트리에 대한 임의의 접근과 구조 변경을 허용할 수 있는 강력한 모델인데, 그 이유는 이들이 물리적인 포인터 수준에서 잠금을 취급하므로서 임의의 자료 구조에 대해 적용가능하기 때문이다. 그러나 XML 트리의 구조 변경이 유효성을 보장하기 위해서는 임의의 구조 변경 행위가 허용되어서는 안 된다. 그러므로 OO2PL처럼 임의의 행위를 모두 허용하는 너무 일반화된 모델은 가능한 성능상의 향상을 얻지 못할 것으로 보인다. 그들은 문서 구조 정보를 활용하여 성능을 향상하는 것에 대해 언급하였으나 반복부에 대한 가상 노드를 이용한다는 언급만 있을 뿐 이를 위한 정형적인 모델이나 알고리즘은 제시하지 않았다.

한편 Dekeyser와 Hidders는 XPath와 같은 경로를 잠금의 단위로 사용하는 다단계 잠금 방식으로 경로 잠금을 제안하였다[8]. 이들의 방법은 //A//B처럼 XML 데이터를 접근하기 위한 일반적인 경로 형태를 지원할 수 있다. 그들은 질의 경로에 대한 상향 또는 하향식 계산이 가능한 알고리즘을 제안하여 성능을 향상시킬 수 있다고 한다. 그러나 이들 방법도 부분트리의 삭제는 지원하지 못하며, 터미널 노드의 삭제만 가능하다.

Grab 등이 제안한 DGLOCK 방법은 데이터 가이드라는 추상적인 데이터 표현 방식에 대하여 Predicate를 검사하는 접근 제어 방법을 제안하였다[10].

한편 XML 문서의 타입 정보를 이용하여 변환에 대한 유효성을 검사하는 방법은 많이 연구되었으며[18], 최근 Bouchou 등에 의해 트리 구조 변경 행위에 대해 점증적인 유효성 검사 방법이 제안되었다[5]. Bouchou 등의 점증적인 유효성 검사 방법은 트리 오토매타가 추가 또는 삭제된 노드의 부모 노드의 생성 규칙을 검사하는 방법으로 트리 구조 변경 행위에 대한 일반화된 유효성 검사 방법이라 할 수 있다.

2. 반복부 중심의 XML 데이터 공유 예

XML 문서 중에서 반복부에 대한 삽입 삭제 만을 허용하는 것은 XML 문서를 공유하는 응용의 일반적인 유형을 잘 반영한다. 이 절에서는 반복부 데이터에 대한 구조 변경 행위를 공유하는 응용 사례를 통해 이 논문에서 제시하는 모델을 소개한다.

2.1 배송 정보 XML 문서의 예

실제 XML 문서를 공유하는 데이터 사용 유형을 살펴보기 위해 아래의 배송 데이터 예를 살펴보자.

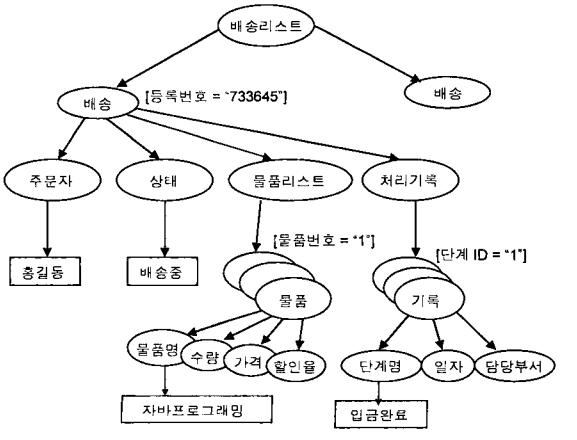
이 XML 문서는 배송할 물건의 리스트를 가지는데, 여러 개의 배송 정보를 포함하며, 각 배송 정보는 기본 정보와 함

〈표 1〉 배송리스트 문서 인스턴스

```
< 배송리스트 >
  < 배송 등록번호 = "733654" >
    < 주문자 > 홍길동 </ 주문자 >
    < 상태 > 배송중 </ 상태 >
    < 주문일 > 2001-08-01 </ 주문일 >
    < 결재액 > 43600 </ 결재액 >
    < 주문내역 >
      < 물품 물품번호 = "1" >
      < 물품명 > 자바 프로그래밍 </ 상품명 >
      < 수량 > 1 </ 수량 >
      < 가격 > 25600 </ 가격 >
      < 할인율 > 20 </ 할인율 >
      </ 물품 >
      < 물품 물품번호 = "2" >
      < 물품명 > JSP 따라하기 </ 상품명 >
      < 수량 > 1 </ 수량 >
      < 가격 > 18000 </ 가격 >
      < 할인율 > 10 </ 할인율 >
      </ 물품 >
    </ 주문내역 >
    < 처리기록 >
      < 기록 단계ID = "1" >
      < 단계명 > 입금완료 </ 단계명 >
      < 일자 > 2001-08-01 </ 일자 >
      < 담당부서 > 온라인관리팀 </ 담당부서 >
      </ 기록 >
      < 기록 단계ID = "2" >
      < 단계명 > 발송 </ 단계명 >
      < 일자 > 2001-08-02 </ 일자 >
      < 담당부서 > 본부물류팀 </ 담당부서 >
      </ 기록 >
    </ 처리기록 >
</ 배송 >
< 배송 >
  - 이하 생략 -
```

〈표 2〉 배송리스트 문서 DTD

```
< 배송리스트 > := (< 배송 >)*
< 배송 > := < 등록번호 >< 주문자 >< 상태 >< 주문일 >< 결재액 >
            < 주문내역 >< 처리기록 >
< 주문내역 > := (< 물품 >)*
< 처리기록 > := (< 기록 >)*
< 물품 > := < 물품명 >< 수량 >< 가격 >< 할인율 >
< 기록 > := < 단계명 >< 일자 >< 담당부서 >
  - 이하 생략 -
```



(그림 1) 〈표 1〉의 배송리스트 문서에 대응하는 트리 인스턴스

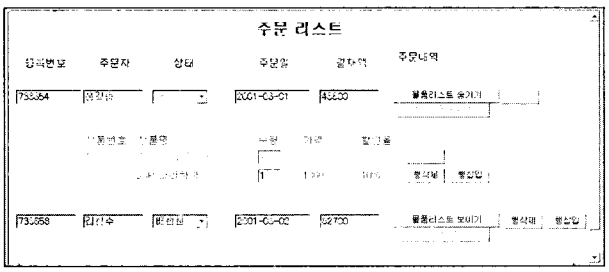
께 배송할 물품의 리스트와 처리 기록 리스트를 가진다. 이 문서에서 반복부는 <배송리스트>, <주문내역>, 그리고 <처리기록> 부분이다. 위 (그림 1)은 이 문서의 트리 구조를 보여주고 있다. 반복부에 속하는 데이터들은 형제 노드와 구분이 가능한 키 값을 속성으로 가진다.

이러한 데이터 모델에 대해 공유 행위는 주로 데이터 값을 동시에 접근하고 수정하는 것과 새로운 항목을 추가 또는 삭제하는 것이 대부분이다. 즉 새로운 배송 요청을 리스트에 추가하거나 이미 등록된 주문에 대해 주문내역을 변경(추가 또는 삭제)하는 경우, 그리고 처리 기록을 추가하는 것이 주로 이 문서에 대한 공유 행위 유형이 된다. 이는 다른 사례에서도 많이 발견되는데, 문헌에서 흔히 사례로 등장하는 경매 시스템의 경우, 대표적인 행위는 경매 항목 추가/삭제, bidding 항목 추가, 경매 항목의 세부 사항(데이터 값) 수정 등이다. 또한 공유 쇼핑 리스트의 예에서도 새로운 쇼핑 항목의 추가나 기존에 있던 쇼핑 항목의 삭제, 쇼핑 항목의 개수나 속성을 바꾸는 행위가 가능해야 한다. 이상의 예에서 볼 수 있듯이 공유 문서에 대한 부분 트리의 삽입/삭제 행위는 대부분 반복 가능한 데이터에 대해서 적용됨을 알 수 있다.

아래 (그림 2)는 반복부에 대한 구조 변경 행위 만을 허용하는 XML 데이터 공유 인터페이스의 예를 보여준다. 사용자에게 제공되는 기능은 다음과 같다.

① 반복부에 대해서 보이기/숨기기 버튼을 두고
② 반복부 각 행은 삭제 버튼을 가지며,
③ 반복부 마지막 행은 삽입 버튼을 가진다.

데이터 중에서 수정될 수 있는 값은 편집창으로 표시된다. 배송리스트의 등록번호나 상태 등의 값은 수정될 수 있는 값이다. 배송리스트의 각 행은 하부 리스트인 주문 내역 및 처리 내역의 리스트 보기 또는 숨기기 버튼을 가지고, 행 삭제 버튼을 가진다. 또한 리스트의 마지막 행은 아래에 추가로 행을 삽입할 수 있는 행삽입 버튼을 가진다.



(그림 2) 배송리스트의 공유 인터페이스 예

편집창이나 선택창은 입력/수정 가능한 값을 표현하는데, 데이터 공유를 위해 값의 수정은 잠금을 획득하여야 가능하다. 첫 번째 주문 리스트(주문번호 733654)의 "상태" 값은

현재 잠금 상태임을 표현한다. 또한 그 "주문"의 삭제 버튼은 비활성화 되어 있는데, 이것은 이 행이 현재 하부 값을 수정 중이므로 삭제할 수 없음을 나타낸다. 또한 주문 내역에서 첫 번째 상품(상품명 자바 프로그래밍)은 수량 데이터를 수정 중이므로 행 삭제를 할 수 없는 경우이다. 그러나 두 번째 상품의 행삭제는 가능하다. 이 인터페이스 화면은 반복부 데이터에 대한 공유의 사례로 본 논문에서 사용된다.

3. 반복부 정규화 XML 트리

본 연구에서는 XML 트리에 대한 구조 변경 행위를 공유하는 한 형태로 반복부 데이터의 삽입과 삭제를 허용하는 트리 공유 모델을 제시하고자 한다. 한번 이상의 삽입 삭제가 반복적으로 적용 가능한 대상이 DTD 상의 * 반복부 부분임에 착안하여 반복부를 다른 부분과 구별하여 독립적인 구조를 가지게 하는 반복부 정규화 방법을 제안한다.

3.1 반복부-factored DTD

XML 트리에서 각 노드가 가지는 자식노드의 형식은 DTD 규칙에 의해 정의되는데, DTD 규칙에서 자식 노드들은 필수, 생략 가능부, 선택부, 그리고 반복부로 나눌 수 있다. 본 논문에서는 DTD 규칙을 표현하기 위하여 사용되는 심볼의 집합을 V, 심볼들을 이용한 정규 표현을 RE(V)라고 표현한다. 그리고 심볼을 나타내는 기호로 A, B, C, $\cdots \in$V를 사용하고 규칙의 우변에 나타나는 정규표현을 나타내는 기호로 $\alpha, \beta, \gamma, \cdots \in$ RE(V)를 사용한다. 정규표현이 *를 포함하지 않은 경우 *-free RE(V)라고 한다. 예를 들어 a, b g가 각각 *-free RE(V)이라면 (a | b) g는 *-free RE(V)이다.

DTD에서 반복되는 부분이 독립된 부분 트리로 묶여지고 반복부를 묶어주는 부모 노드가 다른 노드들과 분리되어 있는 DTD를 *반복부 factored DTD*라고 정의한다.

[정의 3.1] 주어진 DTD 규칙 A ::= δ가 아래와 같은 성질을 만족하면 그 규칙을 반복부factored라고 한다.

① δ가 *-free RE(V)이거나
② $\delta = \alpha(\beta)\times\gamma$의 형태를 가질 때 $\alpha=\varepsilon$, $\gamma=\varepsilon$, $\beta \in V$를 만족한다. 여기서

$$A, B, C, \cdots \in V,$$
$$\alpha, \beta, \gamma, \cdots \in RE(V).$$

DTD의 모든 규칙이 반복부 factored라면 반복부 factored DTD라고 한다.

```
<경매리스트> ::== <분류> (<경매>)* (<취소사유><경매>*)*
                  (<종료일><경매>*))*
```

반복부 factored DTD에서는 생성 규칙의 우측에 반복부가 나타나는 경우 A::= B*와 같은 형식으로 단일 심볼의 반복부만 존재한다. 위 (그림 1)의 예에서 <배송리스트> ::= <배송>*이고 <주문내역> ::= <물품>*, <처리 기록> ::= <처리>*와 같이 반복부는 모두 독립 생성 규칙을 가지고 생성규칙의 우측이 단일 심볼의 반복으로 표시된다. 아래 규칙은 반복부 factored되어 있지 않은 예이다.

이 규칙에 대응하는 반복부 factored 된 DTD 규칙은 다음과 같다.

```
<경매리스트> ::== <분류> <경매리스트> <취소경매리스트>
                  <종료경매리스트>
<경매리스트> ::= <경매>*
<취소경매리스트> ::= <취소경매>*
<취소경매> ::= <취소사유><경매리스트>
<종료경매리스트> ::= <종료경매>*
<종료경매> ::= <종료일><경매리스트>
```

위의 예에서 살펴보았듯이 반복부 factored 되어 있지 않은 DTD 규칙을 새로운 심볼을 도입하여 반복부 factored로 변환할 수 있다. 아래 [알고리즘 3.1]은 위에서 정의된 반복부 factored DTD로의 변환 알고리즘을 소개한다.

[알고리즘 3.1] 반복부 factoring 알고리즘
우변이 *-free RE(V)가 아닌 모든 생성규칙 $A ::= \delta \in DTD$에 대해 다음과 같이 변환한다.

$\delta = \alpha_0(\beta_1)\,\alpha_1(\beta_2)\cdots\alpha_{k-1}(\beta_k)\,\alpha_k$, $\alpha_0, \cdots, \alpha_k \in$ $*-free\ RE(V)$라 하자.

① $A ::= \delta$를 DTD 규칙에서 제외한다.
② 각 i에 대해 $\beta_i \notin V$이면 $C_i ::= \beta_i$와 $B_i ::= C_i*$를 DTD 규칙에 추가한다. $1 \le i \le k$.
③ 각 i에 대해 $\beta_i \in V$이면 $B_i ::= \beta_i*$를 추가한다.
④ $A ::= \alpha_0 B_1 \alpha_1 B_2 \cdots \alpha_{k-1} B_k \alpha_k$를 생성규칙에 추가한다.

이상의 과정을 모든 생성규칙이 반복부 factored일 때까지 반복한다.

반복부 factoring 알고리즘은 반복과정에서 β_i가 반복부를 포함하는 경우 다시 그 규칙을 변환하여 * 반복부를 factoring한다. 즉 한 번의 변환 단계에서 제일 바깥 수준의 반복부를 factoring하는 효과를 얻게 된다. [알고리즘 3.1]은 DTD 내의 * 부호의 개수를 n이라 했을 때 위의 변환 과정을 $O(n)$번 반복하면 변환할 수 있으며 새로 추가되는 심볼의 수는 $2n$개이므로 역시 $O(n)$이다.

이러한 변형은 주어진 XML 트리에 대해서도 가능한데, [13]에서는 가상 노드의 추가로 이와 비슷한 성질을 만족하는 XML 트리를 만들 수 있음을 언급하였다. 본 논문에서는 반복부 factored DTD 문서에 대해 유효한 XML 트리는 반복부 factored라고 부른다. 본 논문에서는 위와 같은 성질을 만족하는 반복부 factored DTD 및 XML 트리만을 다루므

로서 이후의 논의를 간소화하고자 한다.

[정의 3.2] 반복부 factored XML 트리에서 C ::= B*와 같은 유형의 생성규칙에서 C에 대응하는 노드를 리스트 부모 노드라 하고 B에 대응하는 노드를 리스트 (반복부) 노드라고 한다.

예를 들어 (그림 1)에서 <기록>과 <배송> 노드는 리스트 노드이고 <상태> 노드는 리스트 노드가 아니다.

3.2 반복부 정규형

반복부 factored 트리에서 형제 리스트 노드들은 동일한 이름(레이블)을 가지고, 같은 부모를 가진다.

각 노드를 고유하게 표시할 수 있도록 리스트 노드가 형제 노드들 사이에서 구분 가능한 키 값을 가지는 XML 트리의 성격을 *키로 구분가능한 트리*라고 정의한다. 그 경우 각 리스트 노드가 name[key]의 쌍으로 표시되고 여기서 key는 동일한 부모를 가지는 형제 노드들 사이에서 유일한 값을 가진다. 리스트 노드가 아닌 경우 키는 생략될 수 있다. 반복부 리스트 노드의 키 속성은 기존의 ID 형의 속성으로 대신할 수도 있으나 실제로는 같은 부모 노드를 가지는 형제 노드들 사이의 구분만 가능하면 되므로 훨씬 간소화된 모델이다.

예를 들어 (그림 1)에서 각 <배송> 노드는 "배송번호"라는 유일한 키 값을 가져 형제 노드들 간에 구분이 가능하다. 실제 구현에서는 형제 노드들이 시스템에 의해 일련번호를 할당받는 방법을 사용할 수도 있다.

[정의 3.3] 주어진 DTD가 반복부 factored이면서 모든 리스트 반복부 노드들이 키로 구분 가능한 경우 이를 *반복부 정규형*을 만족한다고 한다.

편의상 이하에서의 논의는 모두 반복부 정규형인 DTD와 그에 대해 유효한 XML 트리라고 가정한다.

XML 트리에서 **경로**는 루트 노드에서부터 해당 노드에 도달하기까지 거쳐오는 노드들을 나열하여 하나의 노드를 지정하는 방식이다. 반복부 정규화된 XML 트리에서, 경로 p는 루트 노드에서 시작해서 해당 노드까지 거쳐오는 노드들의 name 또는 name[key] 값의 연속으로 표시된다. 위 (그림 1)에서 <배송리스트><배송>[등록번호 = "733645"]<주문내역><물품>[물품번호 = "2"]는 "JSP 따라하기" <물품> 노드를 가리킨다.

4. XML 데이터의 리스트 데이터 공유 모델

4.1 유효한 반복부 구조 변경 행위

XML 데이터의 공유 행위는 데이터 읽기, 쓰기와 내부 노드에 대한 방문, 삽입, 삭제로 나누어진다. 데이터 읽기와 쓰기는 터미널 노드에 대해 적용되며 나머지는 내부 노드에 대해 적용된다. 각 행위는 다음과 같이 정의된다. 아래에서 w는 터미널 노드이고 v는 내부 노드를 나타낸다.

- Read(w) : 터미널 노드 w의 데이터를 읽는다.
- Write(w, $value_{new}$) : 터미널 노드 w의 데이터를 $value_{new}$로 바꾼다.
- Traverse(v) : 노드 v가 가진 자식 노드 테이블을 읽는다.
- Insert(v, T_{new}) : 노드 v의 자식 트리로 T_{new}를 삽입한다.
- Delete(v, $A[k]$) : 노드 v의 자식 트리 중 키가 k인 $A[k]$ 이하의 자식 트리를 삭제한다.

XML 문서의 공유 행위를 고려할 때 Read/Write는 기존의 데이터에 대한 공유 방법을 그대로 적용할 수 있으므로 본 논문에서는 구조변경 행위인 부분트리의 삽입 및 삭제에 관심을 가진다.

XML 트리에 구조 변경 행위인 삽입 및 삭제를 적용하였을 때 그 결과 트리가 유효한가는 응용 분야에 따라 매우 중요한 문제가 될 수 있다. 여기서는 그러한 행위를 임의의 공유 행위와 구분하여 정의한다. 유효한 XML 트리 T와 T에 적용가능한 행위 a가 있을 때 트리 T에 행위 a를 적용한 결과 트리 T'이 여전히 유효하다면 행위 a를 *유효한 행위*라고 한다.

본 논문에서는 구조 변경 행위의 적용대상을 반복부에 제한하므로써 유효한 행위만을 허용하고자 한다. 위에서 제시한 Insert/Delete의 구조 변경 행위를 다음과 같이 제한하여 이를 *반복부 구조 변경 행위*라고 한다.

[정의 4.1] XML 트리에 대한 부분트리의 삽입/삭제 행위 중에서 *반복부 구조 변경 행위*를 다음과 같이 정의한다. 여기서 v_L은 리스트 부모 노드이다.

- ListInsert(v_L, T_{new}) : v_L의 자식 트리로 T_{new}를 삽입한다. 이 때 $Tnew$의 루트 노드가 가지는 키 값 k는 다른 형제 노드들이 가진 키 값과 구별되어야 한다.
- ListDelete(v_L, $A[k]$) : v_L의 자식 트리 $A[k]$를 삭제한다.

위와 같이 정의된 ListInsert/ListDelete 행위는 항상 유효한 행위임을 아래와 같이 증명할 수 있다.

[정리 4.1] [정의 4.1]에서 정의된 반복부 구조 변경 행위는 항상 유효하다.

[증명] 트리 T가 반복부 정규화된 DTD에 대해 유효한 XML 트리이고 v_L은 *리스트 부모 노드이고* $B ::= A*$*에 대응하는 노드로 자식들이* $w_1 w_2 \ldots w_n$*라 하자.*

행위 a가 ListInsert(v_L, T)이고 T가 v_L의 자식으로 삽입 가능한 트리라면 T의 루트노드는 $A[k]$로 표현가능하고 k는 다른 자식 노드들과 구분가능한 키 값이 되어야 한다. 삽입한 후의 결과 트리는 $n+1$개의 자식을 가지며 반복부 DTD 생성규칙 $B ::= A*$에 대해 유효하다.

한편 ListDelete(v_L, $A[k]$)이고 $wi = A[k]$라 하자. 그러면 $n-1$개의 자식을 가지는 v_L, 노드는 삭제 후에도 $B ::= A*$ 규칙에 대해 유효하다. □

4.2 리스트 잠금 프로토콜

본 연구에서는 MGL(Multiple Granularity Locking)[3]을 변형한 반복부 중심의 리스트 잠금 방법을 제안한다. 대상이 되는 공유 행위는 앞 절에서 정의된 Traverse, ListInsert, 그리고 ListDelete의 세 가지이며, 다음의 다섯 가지 잠금 유형을 가진다.

① 방문 잠금(T) : 자식 노드나 자식 리스트 반복부의 방문을 위해 요청하는 잠금으로, 리스트 부모 노드의 인덱스 테이블에 대해 방문 잠금을 요청하게 된다.
② 리스트 잠금(L) : 리스트 부모 노드의 자식 트리를 삭제하거나 삽입하기 위해 요청하는 잠금으로 리스트 부모 노드가 가진 인덱스 테이블에 대해 쓰기 잠금을 요청하게 된다.
③ 삭제 잠금(D) : 실제 삭제하고자 하는 노드에 요청하는 잠금으로 그 노드 이하의 부분트리를 삭제하게 된다.
④ 하부 방문 잠금(IT) : 하부의 노드를 방문하고자 할 때 요청하는 잠금이다.
⑤ 하부 리스트 잠금(IL) : 하부의 리스트 부모 노드에 리스트 삽입 또는 삭제를 하고자 할때 요청하는 잠금이다.

이상의 잠금 프로토콜 간에 충돌관계가 <표 3>에 나타나 있다. 아래 <표 3>에서 ○는 동시 처리가 허용되는 경우이고 ×는 순차적으로 처리되어야 하는 경우, 그리고 On은 구데이터 읽기(dirty read)가 발생할 수 있는 경우로 알림 방식에 의해 변경을 반영하여야 하는 관계를 나타낸다.

〈표 3〉 리스트 잠금 프로토콜

	IT	IL	T	L	D
IT	○	○	○	○	×
IL	○	○	○	○	×
T	○	○	○	×	×
L	○	○	On	×	×
D	×	×	×	×	×

[알고리즘 4.1] 리스트 잠금 프로토콜

(1) 잠금 설정 단계

① 루트 노드에 제일 먼저 잠금 요청을 한다.
② 해당 공유 행위가 노드 v에 T(v) 또는 IT(v)를 요청하려면 v의 부모 노드 w에 IT(w)를 얻어야 한다.
③ 해당 공유 행위가 노드 v에 대해 L(v) 또는 IL(v)를 요청하려면 v의 부모 노드 w에 IL(w)를 얻어야 한다.
④ 반복부 리스트 방문을 위해서는 해당 리스트 노드 v에 T(v)를 얻어야 한다.
⑤ 리스트 노드 v의 자손을 삭제하기 위해서는 L(v)를 얻은 후 해당 자손 노드 x에 대해 D(x)를 요청해야 한다.
⑥ 리스트 노드 v의 자손을 삽입하기 위해서는 L(v)를 얻은 후 삽입한다. 단, 이 때 키의 충돌이 있는 경우는 실패한다.

(2) 잠금 해지 단계

① 해당 공유 행위가 노드 v의 잠금을 해지하려면 자식 노드에 대한 잠금이 모두 해지되어야 한다.
② 구데이터 읽기가 발생한 경우 L(v)를 해지하기 전에 T(v)를 가지고 있는 다른 모든 클라이언트들에게 변경 사실을 알린다.
③ 해당 공유 행위가 노드의 잠금을 해지하면 루트까지 모든 잠금이 해지되기 전에는 다른 잠금을 얻을 수 없다.

리스트 잠금 방법은 XML 트리의 루트에서부터 하향식으로 필요한 노드의 잠금을 모두 획득하여야 접근이 허용되므로, 삭제 잠금의 경우 하부의 삭제할 노드들에는 잠금을 요청할 필요가 없으며, IT 및 IL 잠금의 경우 루트로부터 해당 노드까지의 경로 상에 있는 노드들에 모두 이 잠금을 얻어야 한다. [보조정리 4.1]은 [알고리즘 4.1]의 리스트 잠금 프로토콜이 리스트 삽입과 삭제 행위의 순차성을 보장함을 증명한다.

[보조정리 4.1] 리스트 잠금 프로토콜에서 동시에 하나의 리스트에 두개 이상의 삽입과 삭제 행위가 동시에 허용되는 경우는 발생하지 않는다.

[증명] 동일한 리스트 부모 노드 v에 대해 삽입 또는 삭제 행위가 허용되었다면 이 노드는 L(v) 잠금을 가지고 있다. 같은 리스트에 대한 삽입 또는 삭제 행위가 또 요청된 경우 v에 대한 L(v) 잠금 요청은 허용되지 않으므로 동시에 허용되는 경우는 발생하지 않는다. □

한편 이 잠금 프로토콜은 터미널 노드의 데이터에 대한 읽기 및 쓰기에 대해서는 다루지 않으므로 이것은 기존의 MGL 잠금이나 2단계 잠금 방식을 사용하여야 할 것이다. 아래 [보조정리 4.2]는 MGL과 본 리스트 잠금 프로토콜이 같이 사용될 수 있음을 보여준다.

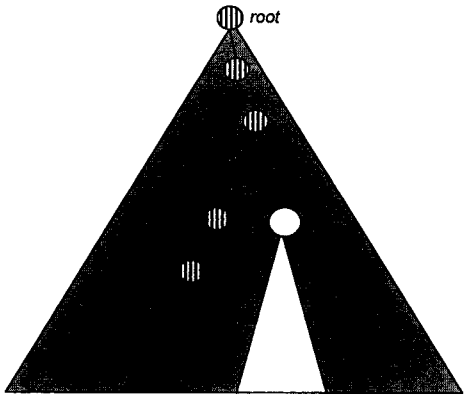
[보조정리 4.2] 리스트 잠금 프로토콜은 터미널 노드의 데이터에 대한 기존의 MGL 방식의 잠금과 같이 사용하여 순차성을 보장할 수 있다.

[증명] MGL의 쓰기 및 읽기 잠금은 터미널 노드에 대해서만 적용되며 내부 노드는 읽기 및 쓰기를 위한 의도 잠금이 적용된다. 이것은 트리 삭제 잠금과 충돌을 일으키고 다른 잠금과는 충돌을 일으키지 않는다. 하부 노드가 다른 트랜잭션에 의해 잠금상태이면 삭제 잠금이 허용될 수 없으므로

이를 반영한 확대 리스트 잠금 프로토콜이 아래 <표 4>와 같다.□

〈표 4〉 터미널 노드에 대한 MGL을 반영한 확대 리스트 잠금 프로토콜

	IT	IL	T	L	D	IR	IW
IT	O	O	O	O	×	O	O
IL	O	O	O	O	×	O	O
T	O	O	O	×	×	O	O
L	O	O	On	×	×	O	O
D	×	×	×	×	×	×	×
IR	O	O	O	O	×	O	O
IW	O	O	O	O	×	O	O



(그림 3) 삽입과 삭제의 동시 적용 예

IL 잠금은 트리를 탑다운으로 방문하면서 수정할 리스트를 가진 부모 노드에 이르기까지 거쳐가는 노드들에 요청하는 것으로 자손 노드에 대한 수정을 위한 잠금이다. 이것은 삭제를 허용하기 전에 하부 노드에 대한 방문 또는 리스트 수정 잠금이 없어야 됨을 반영하기 위해 사용된다. 수정 중인 리스트 노드에 대해 IL은 허용되는데(L → IL), 이것은 현재 삭제 중인 노드에 대한 리스트 수정 요청이 아니라면 (D ↛ IL) 다른 자손 노드의 수정은 허용됨을 의미한다. 한편 방문 잠금 노드에 대한 리스트 수정 잠금 요청(T → L)은 알림 방식에 의해 변경을 반영하는 조건으로 On으로 허용된다. 한편 L 잠금이 설정된 노드에 대해 방문 요청은(L ↛ T) 데이터의 불일치를 막기 위해 허용하지 않아야 한다. 또한 같은 리스트 부모 노드의 자식 노드를 동시에 삽입하거나 삽입/삭제하는 경우 동일한 키 값에 대해 동시에 수정이 허용될 수 있는 가능성이 있으므로 리스트 수정 잠금 요청은 순차적으로 처리되어야 한다(L ↛ L).

이를 Helmer 등의 모델[13]에 대응하면 부모 노드가 자식 노드들에 대한 포인터 리스트를 일종의 인덱스 테이블의 형태로 가지고 이 테이블이 잠금의 단위가 되는 경우로 볼 수 있다. 한편 MGL 방식[11]과 비교하면 묵시적인 잠금 하부의 노드에 서로 다른 행위에 의한 삭제나 삽입이 동시에 허용될 수 있다는 점이 차이다. (그림 5)는 리스트 잠금 프로토콜에서 v_a 노드의 삽입 행위와 하부 자손인 v_d의 삭제가 동시에 허용되는 예를 보여준다.

5. 결 론

XML 문서의 공유를 위하여 여러 사용자가 트리의 구조를 동시에 접근하면서, 부분 트리의 삽입 및 삭제를 요청하는 경우에 대하여 살펴보았다. 본 논문에서는 구조 변경 행위의 유효성을 보장하기 위하여 XML 문서의 반복부에 대해서만 삽입 삭제가 가능하도록 제한한 *리스트 데이터 공유 모델*을 제안하였다. 이 모델은 XML 데이터 공유 응용에서 일반적으로 반복부가 부분 트리의 삽입과 삭제의 대상이 되는 경우가 많음을 반영한다.

리스트 데이터 공유 모델의 정형화된 모델로 XML 문서의 반복부를 독립시킨 반복부 정규화 DTD를 정의하고 임의의 DTD를 반복부 정규화하는 변환 알고리즘을 제시하였다. 그리고 반복부 형제 노드를 대상으로 리스트의 방문 잠금(T)과 리스트 구조 변경 잠금(L)을 가지는 MGL 기반의 *리스트 잠금 프로토콜*을 제안하였다. 본 논문에서 제안된 동시성 제어 방법은 임의의 구조 변경 행위를 허용하는 기존의 연구에 비하여 다음과 같은 장점을 가진다.

첫째, 구조 변경 행위의 적용 대상을 반복부로 한정하므로써 구조 변경 행위의 적용 후에 결과 트리가 반드시 유효함을 보장한다. XML 문서를 공유 데이터로 이용할 때 중간 상태 트리가 DTD에 대해 유효하여야 데이터의 파싱과 처리가 가능하다. 그러나 기존의 연구에서는 구조 변경 행위의 적용에서 DTD 기반의 유효성을 고려하지 않았는데, 본 연구에서는 특히 반복적인 구조 변경 행위의 적용 가능성을 고려하였을 때에도 유효성을 보장한다.

두 번째로 적용 대상을 반복부에 한정한 결과 잠금 프로토콜의 효율성을 높일 수 있었다. 구조 변경 행위에 대한 대상 노드가 반복부 리스트 부모 노드이므로 잠금 대상 객체의 수가 훨씬 적어진다. 또한 계층적인 리스트 잠금 방식에 의해 중간의 노드가 자식 트리의 삽입 또는 삭제를 위해 잠금 상태라 하더라도 다른 자손의 리스트 노드에 대한 방문이나 삽입, 삭제가 허용될 수 있다.

본 연구의 결과는 Helmer 등이 제안한 잠금 프로토콜인 OO2PL과 비교하여 훨씬 적은 수의 잠금 만으로 동시성 제어가 가능하고, 포인터 수준의 자료 구조에 의존하는 OO2PL 모델에 비하여 높은 수준에서 잠금 모델을 정의할 수 있었다. 이러한 차이는 본 모델에서는 DTD의 구조정보를 이용하면서 키 기반의 리스트 잠금 방식을 사용하여 반복부에 대한 구조 변경 행위의 특성을 반영하였기 때문이다.

반복부 정규화 XML 문서에 대한 공유 모델은 실제 XML 문서를 공유하는 응용 형태에 잘 적용될 뿐 아니라 매우 효

율적인 구현이 가능하여 실용적인 방법으로 쓰일 수 있을 것으로 기대된다. 본 논문에서 제안된 공유 모델은 인터넷 상에서 XForms 기반으로 구현 중이며[2], 반복부에 대한 구조 변경 만을 허용하는 XML 문서 변경을 통합하는 협상 방법에도 적용될 수 있을 것으로 기대된다.

참 고 문 헌

[1] 송하주, 김형주, "XML 데이터의 효과적인 검색을 위한 다중 경로", *한국정보과학회논문지 : 컴퓨팅의 실제*, 제7권 제1호, pp.12-23, 2001.

[2] 서원일 외, "ThruDoc : XForms를 이용한 액티브 문서 플랫폼 개발 ThruDoc : Development of an Active Document System Based on XForms," *한국정보과학회 02 가을 학술발표논문집*(2) pp.184-186, Oct., 2002.

[3] N. S. Barghouti, G. E. Kaiser, "Concurrency control in advanced database applications," *ACM Computing Surveys*, Vol.23, No.3, pp.269-317, 1991.

[4] P. Bernstein, V. Hadzilacos and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Reading, Mass : Addison Wesley, 1987.

[5] B. Bouchou and M. Halfeld, F. Alves, "Updates and Incremental Validation of XML Documents," *The 9th International Workshop on Data Base Programming Languages* (*DBPL* '03), pp.216-232, 2003.

[6] Eun-Hye Choi and Tatsunori Kanai, "XPath-based Concurrency Control for XML Data," In : *Proceedings of the 14th Data Engineering Workshop* (*DEWS* 2003), Kaga city, Ishikawa, Japan, March, 2003. Available at : http://www.ieice.org/iss/de/DEWS/proc/2003/papers/6-C/6-C-04.pdf.

[7] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, "A Fine-Grained Access Control System for XML Documents," *ACM Transactions on Information and System Security* (*TISSEC*), Vol.5, Issue 2, pp.169-202, 2002.

[8] Stijn Dekeyser, Jan Hidders, "Path locks for XML Document collaboration," *Proc. WISE* '02, pp.105-114, 2002.

[9] T. Fiebig, et al., "Anatomy of a Native XML database management system," *VLDB Journal*, Vol.11, No.4, pp. 292-314, 2002.

[10] Torsten Grabs, Klemens Böhm, Hans-Jörg Schek, "XML-TM : Efficient Transaction Management for XML Documents," *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management*, McLean, VA, USA, November, 2002.

[11] J. N. Gray, et al., "Granularity of Locks and Degrees of Consistency in a Shared Data Base," *IBM Research Report* RJ1654, Sept., 1975.

[12] Satoshi Hada and Michiharu Kudo, "XML Access Control Language : Provisional Authorization for XML Documents," *IEICE Trans. Fundamentals*, Vol.E-84A, No.1, 2001.

[13] S. Helmer, C. Kanne, G. Moerkotte, "Lock-based Protocols for Cooperation on XML Documents," *Int. Workshop on DB and Expert Systems Applications Conference* (*DEXA* '03), pp.230-234, 2003.

[14] S. Helmer, C. Kanne, G. Moerkotte, "Evaluating lock-based protocols for coorperation on XML documents," *ACM SIGMOD Record,* Vol.33, Issue 1, pp.58-63, March, 2004.

[15] H. V. Jagadish, et al, "TIMBER : A native XML database," *VLDB Journal,* Vol.11, pp.274-291, 2002.

[16] Kuen-Fang Jea, Shih-Ying Chen and Sheng-Hsien Wang, "Concurrency Control in XML Document Databases : XPath Locking Protocol," P*roceedings of the 9th International Conference on Parallel a6d Distributed Systems* (ICPADS 2002), Taiwan, ROC, IEEE, pp.551-556, December, 2002. Available http://csdl.computer/comp/proceedings/icpads/2002/1760/00/17600551abs.htm.

[17] S.-Y. Lee and R.-L. Liou, "A multi-granularity locking model for concurrency control in object-oriented database systems," *IEEE Trans. On Knowledge and Data Engineering*, Vol.8, No.1, pp.144-156, 1996.

[18] Igor Tatarinov, Zachary G. Ives, Alon Y. Halevy, Daniel S. Weld, "Updating XML," *Proceedings of ACM SIGMOD.*, pp.413-424, 2001.

이 은 정

e-mail : ejlee@kyonggi.ac.kr

1988년 서울대학교 계산통계학과(학사)

1990년 한국과학기술원 전산학과(공학석사)

1994년 한국과학기술원 전산학과(공학박사)

1994년~2000년 한국전자통신연구원 선임연구원

2001년~현재 경기대학교 정보과학부 조교수

관심분야 : 컴파일러 이론, XML 처리 기술 등